

PhD Dissertation



International Doctorate School in Information and
Communication Technologies

DISI - University of Trento

INFORMATION QUALITY REQUIREMENTS ENGINEERING: A
GOAL-BASED MODELING AND REASONING APPROACH

Mohamad Gharib

Advisor:

Prof. Paolo Giorgini

Università degli Studi di Trento

April 2015

Abstract

Information Quality (IQ) has been always a growing concern for most organizations, since they depend on information for managing their daily tasks, delivering their services to their costumers, making important decisions, etc., and relying on low-quality information may negatively influence their overall performance, or even disasters in the case of critical systems (e.g., air traffic management systems, healthcare systems, etc.). Although there exist several techniques for dealing with IQ related problems in the literature (e.g., checksum, integrity constraints, etc.), but most of them propose solutions that are able to address the technical aspects of IQ, and seem to be limited in addressing social and organizational aspects. In other words, these techniques do not satisfy the needs of current complex systems, such as socio-technical systems, where humans and their interactions are considered as an integral part of the system along with the technical elements (e.g., healthcare systems, smart cities, etc.). This introduces the need of analyzing the social and organizational context where the system will eventually operates, since IQ related problems might manifest themselves in the actors' interactions and dependencies. Moreover, considering IQ requirements since the early phase of the system development (the requirements phase) can prevent revising the system to accommodate such needs after the system deployment, which might be too costly. Despite this, most of the Requirements Engineering (RE) frameworks and approaches either loosely define, or simply ignore IQ requirements. To this end, we propose a goal-oriented framework for modeling and reasoning about IQ requirements since the early phases of the system development. The proposed framework consists of (i) a modeling language that provides concepts and constructs for modeling IQ requirements; (ii) a set of analysis techniques that support system designers while performing the required analysis to verify the correctness and consistency of the IQ requirements model; (iii) an engineering methodology to assist designers in using the framework for capturing IQ requirements; and (iv) an automated tool-support, namely ST-IQ Tool. In addition, we empirically evaluated the framework to demonstrate its applicability, usefulness, and the scalability of its reasoning techniques by successfully applying it to a case study concerning a stock market system.

Keywords[Requirements Engineering, Information Quality Requirements, Critical systems, Socio-technical Systems]

Acknowledgements

I would like to thank all the people who contributed to the work described in this thesis. First and foremost, I would like to express my gratitude to my supervisor Prof. Paolo Giorgini for his guidance, advice, and support. I would like also to thank Prof. Oscar Pastor, Prof. Haris Mouratidis, and Prof. Luca Spalazzi for accepting the invitation to participate in my thesis defense committee, and for their valuable advices for improving my work. Big thanks to Prof. John Mylopoulos for his valuable feedback. Special thanks to Dr. Fabiano Dalpiaz and Dr. Raian Ali for their help and interesting discussions. I thank all my colleagues at Trento University, for their help, support and valuable feedback. Last but not least, I would like to thank my family; this thesis would not have been possible without your help.

Mohamad

Contents

1	Introduction	1
1.1	Objectives of our research	4
1.2	Contributions of the Thesis	5
1.3	Structure of the Thesis	6
1.4	Published Work	7
2	State of the art	9
2.1	Organizational Engineering	9
2.1.1	Enterprise Engineering	10
2.1.2	Enterprise and Organization Ontologies	11
2.1.3	Organizational Aspects in AOSE and MAS	12
2.1.4	Organizational aspects in Requirements Engineering community . .	13
2.2	Trust Management	14
2.3	Information and Information Quality	16
2.3.1	Information	16
2.3.2	Information Quality	18
2.3.3	Information Quality in the Literature	22
2.4	Chapter summary	28
3	The Modeling Language	29
3.1	The modeling language objectives	30
3.2	The modeling language principles	31
3.3	The essential modeling concepts	31
3.3.1	Actors, roles, and agents	32
3.3.2	Goals, information, and their relations	33
3.3.3	Actors' objectives and entitlements	34
3.3.4	Actors' social interactions	35
3.4	Social trust concepts	37
3.4.1	Analyzing trust based on the actors' internal structure	38

3.4.2	Trust and Distrust Analysis	41
3.5	Information Quality concepts	43
3.5.1	Multi-dimensional Model for Analyzing Information Quality	43
3.5.2	Information Quality constructs	50
3.6	Workflow net with actors (WFA-net)	54
3.6.1	Approach for Modeling and Reasoning about IQ Requirements in Business Process	56
3.6.2	Analysis Phase	62
3.7	Dealing with IQ requirements	62
3.8	Chapter summary	66
4	Automated Reasoning Support	67
4.1	Concepts and Relations	67
4.1.1	Extensional predicates	68
4.1.2	Intensional predicates	71
4.2	Reasoning rules (axioms)	76
4.2.1	Actors' objectives, entitlements and capabilities axioms	76
4.2.2	Information Quality analysis axioms	78
4.2.3	Actors' social interactions	83
4.2.4	Trust analysis axioms	84
4.2.5	Goals achievement axioms	84
4.3	Verifying the requirements model (properties of the design)	85
4.4	Chapter summary	88
5	Automated Information Quality Policy Specification	89
5.1	Information Quality Policy Specification Language	90
5.2	Rules for Automated Derivation of Information Quality Policy Specifications	92
5.3	Chapter summary	96
6	The Methodological Process & Computer-Aided Support: ST-IQ Tool	97
6.1	The Methodological Process	97
6.1.1	A detailed description of the methodological process	99
6.2	Computer-Aided Support: ST-IQ Tool	103
6.2.1	ST-IQ Tool Architecture	103
6.2.2	Modeling component	104
6.2.3	Model-To-Text (M2T) component	105
6.2.4	Reasoning component	106

6.3	Chapter summary	107
7	Case study: the Flash crash	109
7.1	The stock market system structure	110
7.1.1	The Flash Crash: chronology of events	111
7.1.2	Main reasons of the Flash Crash	111
7.1.3	Applying the framework to the Flash Crash case study	112
7.1.4	Requirements Modeling	112
7.1.5	Analyzing the requirements model	122
7.1.6	Deriving the final IQ Specifications	124
7.2	Scalability experiments	125
7.2.1	Design of the study	125
7.2.2	Experiment results	125
7.3	Chapter summary	125
8	Conclusions and future work	127
8.1	Summary of the Thesis	127
8.2	Limitations of the framework	129
8.3	Ongoing and Future Work	129
	Bibliography	131

List of Tables

2.1	IQ dimensions in literature	18
3.1	Analyzing trust based on the actors' internal structure	42
3.2	IQ softgoal classification & approximation into IQC	65
4.1	Extensional Predicates	69
4.2	Intensional Predicates	72
4.3	Actors' objectives, entitlements and capabilities axioms	77
4.4	Information Quality analysis axioms	78
4.5	Social Relations Axioms	83
4.6	Trust analysis axioms	84
4.7	Goals Achievement Axioms	85
4.8	Properties of the design	87
5.1	Rules for Automated Derivation of IQ Specifications	93

List of Figures

3.1	Graphical representation of actors, agents, roles, and their interrelations . . .	32
3.2	Graphical representation of goal, and its and/or-decompositions	33
3.3	Graphical representation of goal-information relations	34
3.4	Graphical representation of actor's objective and information ownership . .	35
3.5	Graphical representation of goal delegation & information provision	36
3.6	Graphical representation of permissions delegation	36
3.7	A partial goal model concerning the stock market structure	37
3.8	Graphical representation of trust/distrust of goals & permissions delegation	38
3.9	Graphical representation of goal/ information corresponding threat	39
3.10	Graphical representation of roles' capability/incapability toward achieving a threat	40
3.11	Graphical representation of goals positive and negative contributions . . .	40
3.12	Graphical representation of threat-goal positive and negative contributions	41
3.13	A partial goal model concerning trust analysis based on the actors' internal structure	41
3.14	Graphical representation of goal/information monitoring	43
3.15	A multi-dimensional model for analyzing IQ from social perspective	45
3.16	A meta-model of the main concepts of the modeling language	54
3.17	Overview of the approach for modeling and analyzing IQ in BP	55
3.18	A partial goal model concerning the stock market structure	57
3.19	A WFA-net of a stock investor for trading securities	59
3.20	Process: from informal to formal IQ requirements	64
5.1	A partial goal model concerning the stock market structure	94
6.1	The Methodological Process	98
6.2	ST-IQ Tool architecture	104
6.3	ST-IQ Tool: Modeling Interface	105
6.4	ST-IQ Tool: Model-2-text Component	106

6.5	Disjunctive Datalog formal specifications	106
7.1	Actor Diagram	116
7.2	Partial Goal Diagram	118
7.3	Partial Information Diagram	119
7.4	Partial Social Interaction Diagram	120
7.5	Partial Threat Diagram	121
7.6	Partial Trust Diagram	122
7.7	Scalability results with increasing the number of modeling elements	126

Chapter 1

Introduction

Whether you think that you can, or that you can't, you are usually right.

Henry Ford

Most organizations depend on information to perform their everyday tasks, e.g., deliver their services to customers, support their business processes, assist their decision making, etc. In this context, the efficient performance of these tasks is highly influenced by the quality of information they depend on. More specifically, Information Quality (IQ) is a key success factor for organizations, since depending on low-quality information may result in undesirable outcome [Red95], or even disasters in the case of critical systems [GG13b] (e.g., Air Traffic Management, health care systems, etc.).

Generally speaking, quality can be defined as “fitness for use” [JGB79], or as in [RB94] the conformance to specifications, i.e., meeting or exceeding consumer expectations. Similarly, IQ can be defined based on the “fitness for use”, i.e., determining whether IQ is high or low depends on its “fitness for use”. In other words, the same information might be considered as high quality for one user and low quality for another one. For example, a stock market investor who uses his laptop to trade some securities, the level of IQ required by him concerning his trades is not the same as the IQ level required by a main stock market (e.g., NYSE, NASDAQ) that is responsible of managing thousands of trades in milliseconds simultaneously. In the first case, low-quality information can be accepted to a certain level, while in the second case it may result in a financial disaster (e.g., stock market mini crash, full crash, loses of millions of dollars, etc.).

In addition, IQ is a hierarchical multi-dimensional concept that can be characterized by different dimensions, including: accessibility, accuracy, completeness, timeliness, consistency, trustworthiness, etc. [Red95; PLW02; BSM03; Jia10]. Thus, determining whether IQ “fitness for use” or not depends on analyzing its different dimensions. That is why

deciding whether IQ is high or low is not a trivial task. For instance, a stock investor can evaluate the quality of its trading order (information) at the intended trading market by analyzing the order accuracy, completeness, timeliness (validity), etc.

Although there exist several models for analyzing IQ based on its different dimensions [LC02; PLW02; WRK95; BSM03], yet most of them share several common limitations, including: (1) ambiguity [Jia10], (2) subjectivity: there is no clear theoretical bases to justify why a certain IQ dimension is considered or not for analyzing IQ [LC02]; (3) inconsistency among the dimensions they consider (e.g., completeness is a sub-dimension of believability in [WRK95], while it is a sub-dimension of integrity in [BSM03]). Moreover, most of them propose holistic methods for analyzing IQ (one size fits all), i.e., they consider a user-centric view [WS96] without taking into consideration the different relations between information and its purposes of use. In particular, they do not explicitly capture the “fitness for use” for “what” and the “fitness for use” of “who”, which is very important when information has several stakeholders, who may require different (might be conflicting) quality needs. More specifically, existing approaches miss the clear semantics that enables for capturing IQ needs. Without having such semantics, it is hard to determine whether IQ “fits for use” or not.

Consider for example, a stock investor who wants to send a trading order to a stock market through a stock trader. This simple scenario raises several questions: How can we define the IQ about the order? Does its quality have the same meaning for the different stakeholders (e.g., investor, trader, market)? Do all stakeholders have the same purpose of information usage? How we can define the quality of the order based on the different purposes of use? Finally, do these stakeholders have the same IQ needs? If not, how do they differ? Actually, the previous questions cannot be properly answered without defining a clear semantics between information, its quality, and the stakeholders’ intended purposes of information usage.

Moreover, not only how IQ can be analyzed based on its different dimensions is problematic, but also how each of these dimensions can be measured is problematic as well. In particular, IQ requirements used to be represented as generic non-functional properties of the system-to-be, without specific methods for their analysis [CdPL09]. However, it is not possible to properly deal with IQ requirements without providing a clear-cut criteria for their satisfaction, i.e., removing any ambiguity related to their verification among the different stakeholders of the system by providing a general consensus among the stakeholders on how IQ dimensions can be represented and measured.

At the other hand, several approaches and techniques for dealing with IQ related concerns have been proposed in the literature (e.g., checksum [Coh87], integrity constraints [Mot89], etc.), but most of them focus on the technical aspects of IQ, and do not con-

sider the social or organizational aspects. More specifically, they do not satisfy the needs of current complex systems, such as socio-technical systems [ET60], where humans and their interactions are considered as an integral part of the system along with the technical elements (e.g., healthcare systems, smart cities, etc.). Fisher and Kingma [FK01] highlighted the limitation of existing approaches for addressing IQ concerns that might arise at the social and organizational levels. They showed how different kinds of vulnerabilities might manifest themselves in the actors' interactions and dependencies, and how such approaches are not able to address such vulnerabilities.

For instance, the Flash Crash (a main stock market crash) is an example where the problem was not caused by a mere technical failure, but it was due to several socio-technical related vulnerabilities of the system [Cli11; SCC⁺12]. In particular, several reasons contributed to the Flash Crash were caused by social and organizational issues. For example, some traders exploit vulnerabilities in the system design, and intentionally provide fraud/ falsified information in order to destabilized the trading environment and influence the prices of the securities they aim to buy/ sell before their real trades [KKST11]. Others continue trading during the crash by forwarding their orders to the markets that did not halt their trading activities due to lack of coordination among the markets[GHLZ12; Sub13], where the lack of coordination resulted also from IQ vulnerabilities. This introduces the need of analyzing IQ needs in their social and organizational context where the system will eventually operates.

In addition, most of these approaches provide ad-hoc techniques to deal with IQ related vulnerabilities, instead of solving the main reason for such vulnerabilities by considering them during the early phase of the system development (e.g., requirements level). This is particularly important since considering IQ during the early development phase of the system can prevent revising the entire system to accommodate such needs after the system deployment, which might be too costly. Despite this, most of the Requirements Engineering (RE) frameworks and approaches either loosely define, or simply ignore IQ requirements (e.g., UMLsec [Jür05], *i** [Yu95], etc.).

In summary, a RE framework for capturing IQ requirements is still missing. To this end, we propose a novel RE framework for modeling and reasoning about IQ requirements from the early phases of the system development. The framework offers: (i) a modeling language that provides concepts and constructs for modeling IQ requirements; (ii) a set of analysis techniques that help in verifying the correctness and consistency of the IQ requirements model; (iii) an engineering methodology to assist system designers during the different phases of the system design; and (iv) an automated tool-support (ST-IQ Tool).

1.1 Objectives of our research

This thesis aims to propose a RE framework that is able to model and analyze IQ requirements since the early phases of the system development. In order to achieve that, we need to answer the following questions:

RQ1: How can we analyze IQ in socio-technical systems? in other words, which IQ dimensions should be considered for analyzing IQ, and how each of the considered dimensions can be analyzed taking into consideration the intentional, social and organizational aspects that might underlie some of them. We address this question by proposing a multi-dimensional model for analyzing IQ from socio-technical perspectives in Section 3.

RQ2: How can we deal with IQ requirements? most RE approaches deal with IQ requirements as generic non-functional requirements, without specific methods for their analysis [CdPL09]. Usually, non-functional requirements (e.g., IQ requirements) are more difficult to be expressed in a measurable way [NE00], since they do not always have a clear-cut criteria for their satisfaction [MCN92; CdPL09]. However, it is not possible to properly deal with IQ requirements without removing any ambiguity related to their verification, i.e., providing a clear-cut criteria for their satisfaction. We address this question in Section 3 by proposing a systematic process that starts by identifying top-level IQ requirements as softgoals (informal), and then gradually refining them until reaching their formal operational specification.

RQ3: How can we model the IQ requirements? it is generally accepted in the RE community that system development requires models that represent the system-to-be [NE00]. One main benefit of such model is representing the system requirements in a simple way, which enables the system designer along with the different stakeholders of the system to communicate and understand each other perfectly. However, existing RE frameworks and approaches (e.g., i^* [Yu95], Secure Tropos [MG07], SI* [Zan06], etc.) do not propose concepts or constructs for modeling IQ needs. We address this question in Section 3 by proposing the required concepts and constructs for modeling the IQ requirements of the system-to-be.

RQ4: How can we verify the IQ requirements model? in other words, how can we verify that the requirements model is correct and consistent, i.e., it meets the stakeholders' requirements, and there is no inconsistencies among such requirements. We address this question in Section 4 by proposing: (i) a formalization of the concepts that our framework offers; (ii) a set of reasoning axioms; and (iii) a set of

properties of the design that enables for detecting incorrectness and/or inconsistency in the requirements model, which helps in resolving them.

RQ5: How the final specifications of the IQ requirements can be defined? as highlighted in [CCRC13], specifying IQ requirements is not a trivial task, since we need to guarantee that the defined specifications are consistent with one another, and they are also consistent with the social and organizational aspects, where the system will eventually operates. We address this question in Section 5 by proposing a mechanism to support the automatic derivation of IQ specifications from the IQ requirements model, and represent them in clearly defined IQ specification language.

RQ6: How can we support system designers in constructing the system-to-be? we address this question in Section 6 by suggesting a detailed methodological process to be followed by designers during the different phases of the system design, i.e., starting from IQ requirements modeling until deriving the final operational specification from such requirements.

RQ7: How well does the framework performs when applied to realistic settings? validating the proposed framework provide an evidence about its usefulness in terms of its applicability and performance. We address this question in Section 7 by performing a set of experiments to verify whether the framework can efficiently perform the tasks it has been developed to perform. In particular, we need to answer questions like: **RQ7.1:** do the modeling concepts and constructs cover all the required aspects of the system?; **RQ7.2:** is the methodological process easy to be understood and used by designers?; **RQ7.3:** do the proposed analysis techniques provide all the required reasoning we need?; and **RQ7.4:** how well the CASE tool (ST-IQ Tool) performs, can it deal with large models?.

1.2 Contributions of the Thesis

This thesis provides the following contributions:

A modeling language for capturing IQ requirements: we propose a modeling language that introduces concepts and constructs for modeling the stakeholders of the system along with their objectives, entitlements, capabilities, and their social interactions (e.g., information provision, goals and permissions delegation). In addition, it proposes more refined constructs for modeling trust relations among actors, and it introduces specialized concepts and constructs for modeling IQ related aspects.

A modeling language for capturing IQ requirements in Business Processes:

we propose Workflow net with Actors (WFA-net) that is a modeling extension of our framework, which enables for modeling and analyzing IQ requirements in Business Processes.

IQ policy specification language: that is used to represent IQ specifications in terms of IQ policies that clearly define the permitted, forbidden and obligated actors' activities toward information.

A formal framework: that supports the automated reasoning about the IQ requirements model. In particular, the framework introduces formalization for all the concepts that or framework supports, and it proposes the required reasoning axioms along with a set of properties of the design that enables for verifying the correctness and consistency of the IQ requirements model.

A RE methodology specialized for capturing IQ requirements: the proposed methodology assist system designers during the different phases of the system design. In Particular, it supports designers during the requirements modeling phase, and it helps them during the analysis phase to perform the required analysis to verify the correctness and consistency of the IQ requirements model against some properties of the design. Finally, it supports designers while deriving the final operational specification from the IQ requirements model.

A CASE tool (ST-IQ Tool): the prototype tool has been developed to assist system designers and analysts during the system design process. ST-IQ Tool enables for modeling the IQ requirements model, and transforming it into formal specification that allows for performing the required analysis to verify the correctness and consistency of the model. Moreover, it supports deriving the final operational specification from the IQ requirements model in terms of IQ policies.

1.3 Structure of the Thesis

This thesis is organized as follows:

Chapter 2 presents an overview of the state-of-the-art related to our work. We start by reviewing the main works in Organization and Enterprise Engineering, and then we discuss several related work in trust management. Finally, we overview how different communities have dealt with IQ related issues.

Chapter 3 propose our modeling language. The chapter starts by introducing the objectives that underlie our modeling language, and then we discuss several principles that have been taken into account while developing the language. Finally, we introduce our modeling language that proposes concepts and constructs for modeling IQ requirements in their social and organizational context. The work partially appeared in [GG13a; GG15d; GG15a; GG15e].

Chapter 4 discusses the formal framework that underlies our approach, which enables for performing the required analysis to verify the correctness and consistency of the IQ requirements model. The work partially appeared in [GG15d; GG15a; GG15e].

Chapter 5 introduces our proposed IQ specification language that clearly defines the permitted, forbidden and obligated actors' activities toward information. Moreover, it discusses the mechanisms that we rely on for the automatic derivation of IQ policies from the IQ requirements model. The work partially appeared in [GG15c].

Chapter 6 describes our proposed methodology that supports system designers/ analysts during the different phases of the system design. In addition, it presents our CASE tool (ST-IQ Tool) that support our framework. The tool provides a GUI for drawing IQ requirements models, supports translating the graphical models into disjunctive Datalog formal specifications, and helps in verifying the correctness and consistency of the requirements models. The work partially appeared in [GG15b; GG15d; GG15a; GG15e].

Chapter 7 illustrates the efficiency of our proposed framework for modeling and analyzing IQ requirements of a case study concerning a stock market crash (the Flash Crash). In this section, we empirically evaluate the usefulness of the proposed framework. The work partially appeared in [GG14; GG15d; GG15a; GG15e].

Chapter 8 concludes the thesis with a summary of our contributions, its limitations and a brief discussion of future work.

1.4 Published Work

- Mohamad Gharib and Paolo Giorgini. "Modeling and analyzing Information Quality Requirements for Socio-technical Systems: Experience Report", 1st International Workshop on Socio-Technical Perspective in IS development(STPIS 2015). To appear.

- Mohamad Gharib and Paolo Giorgini. “Dealing with Information Quality Requirements”, *Enterprise, Business-Process and Information Systems Modeling (EMM-SAD’15)*, Springer, 2015. To appear.
- Mohamad Gharib and Paolo Giorgini. “Modeling and Reasoning about Information Quality Requirements in Business Processes”, *Enterprise, Business-Process and Information Systems Modeling (BPMDS’15)*, Springer, 2015. To appear.
- Mohamad Gharib and Paolo Giorgini. “A Goal-based Approach for Automated Specification of Information Quality Policies”, In *Proceedings of Research Challenges in Information Science (RCIS), 2015 IEEE Ninth International Conference*, pages 177-188.
- Mohamad Gharib and Paolo Giorgini. “Modeling and reasoning about information quality requirements”, *Requirements Engineering: Foundation for Software Quality (REFSQ’15)*, LNCS 9013, Springer 2015, pages 49-64.
- Mohamad Gharib and Paolo Giorgini. “A Framework for Information Quality Requirements Engineering”. In *Proceedings of REFSQ-2015 Workshops, Research Method Track, and Poster Track, Essen, Germany, March 23, 2015. Proceedings*, pages 218-219.
- Mohamad Gharib and Paolo Giorgini. “Analysing information integrity requirements in safety critical systems”. In *Proceedings of the 6th International i star Workshop 2013. Proceedings*, pages 85-90.
- Mohamad Gharib and Paolo Giorgini. “Modeling and analyzing information integrity in safety critical systems”. In *Proceedings of the 3rd International Workshop on Information Systems Security Engineering WISSE’13*, pages 524–529.

Chapter 2

State of the art

This chapter aims to review the most relevant research efforts on capturing IQ requirements in their social and organizational context. As previously mentioned, socio-technical systems are not pure technical systems, but they are also composed of humans who operate these systems along with their interrelations and dependencies. Thus, considering only the technical aspects of the socio-technical system leaves the social and organizational aspects outside the system's boundary, which leaves the system open to different kinds of vulnerabilities that may arise at business, social and organizational levels. This introduces the need of analyzing the social and organizational environment where the system will operate. In particular, the development of such system requires adopting different ideas and techniques from several research areas, such as (1) Organization and Enterprise Engineering areas 2.1, which propose concepts enable for modeling the social and organizational context where the system will eventually operates; (2) Trust Management area in 2.2 that proposes concepts enable for modeling the social interactions and dependencies among the actors of the system. Finally, (3) IQ area in 2.3 that discusses concepts related to information, IQ along with its dimensions, metrics and measurements, and then we review how different communities have dealt with IQ needs.

2.1 Organizational Engineering

Organizational Engineering aggregates multi-disciplinary concepts, methods and technology to model, develop and analyze various aspects of organizations [BWHW05]. Different research communities have considered the problem of modeling and analyzing organizational setting (e.g., Enterprise Engineering, Agent Oriented Software Engineering (AOSE), Multi-agent Systems (MAS), etc.). In this section, we review several proposals for modeling organizational aspects in Enterprise Engineering area 2.1.1, and then we present several organizational concepts that have been introduced in organization and enterprise

ontologies 2.1.2. Moreover, we discuss organizational concepts that have been proposed by Agent Oriented Software Engineering (AOSE) and Multi-Agent Systems (MAS) communities 2.1.3. Finally, we introduce the main organizational concepts that have been proposed by the RE community 2.1.4.

2.1.1 Enterprise Engineering

Organizations are becoming increasingly complex and competitive, since they need to adapt to the rapidly changing markets [Sta96]. Usually, the organizational modeling of an enterprise is often dealt with by Enterprise Engineering methodologies [Zan06], where Enterprise Engineering is a sub-discipline of Systems Engineering. In particular, Enterprise Engineering is defined as the body of knowledge, principles, and practices related to analyzing, designing, and implementing of an enterprise [LJMU95]. In other word, it focuses on the design of the enterprise as a whole, which enables the enterprise to more effectively achieve its goals and objectives. In what follows, we discuss the most influential frameworks for enterprise and organizations modeling.

Computer-Integrated Manufacturing Open-System Architecture (CIMOSA) [Kos95] is an enterprise modeling framework, which aims to support the enterprise integration of machines, computers, and people. CIMOSA supports complete life cycle of enterprise modeling (e.g., requirements definition, design specification and implementation description). The sequence of modeling is optional, and modeling may start at any of the enterprise system life cycle phases, and it might be iterative as well. CIMOSA defines four different modeling views, namely: function, information, resource and organization. The functional view describes the functionality and behavior of the enterprise operations, and it can be described by processes, events and enterprise activities. The information view describes the inputs and outputs of enterprise, and information that is required by each function. The resource view describes all the resources needed by functions. Finally, the organization view describes the enterprise organizational structure.

The Open Group Architecture Framework (TOGAF) [HV07] is a framework for enterprise architecture modeling, which can be used for designing, planning, implementing, and governing enterprise information architecture. TOGAF provides a detailed, step-by-step method on how to build, maintain, and implement enterprise architecture. The core components of TOGAF are (1) Architecture Development Method (ADM); and (2) TOGAF Foundation Architecture, where the first defines a process for developing and maintaining the organization's enterprise architecture, and for implementing the architecture through a planned program of work. While the last describes an enterprise continuum through which the development of architecture progresses from the general (foundation) to the organization-specific ones.

Generic Enterprise Reference Architecture and Methodology (GERAM) [BN96] is an enterprise architecture framework that can be used for designing and maintaining enterprises during their entire life-cycle. GERAM introduces the required concepts for describing the structure, content, and behavior of enterprises. GERAM can be described as a generic architecture framework because it applies to all types of enterprises.

2.1.2 Enterprise and Organization Ontologies

In computer science area, an ontology is a formal specification of shared conceptualization of a common area of interest among a community of people [Die06]. In this section, we investigate most influential ontologies in both enterprise and organizational areas.

Enterprise Ontology

The ENTERPRISE ontology introduced by the University of Edinburgh [UKMZ98] proposes five top-level classes for integrating the various aspects of an enterprise: (1) meta-ontology: entity, relationship, role, actor, and state of affairs; (2) activities and processes: activity, resource, plan, and capability; (3) organization: organizational unit, legal entity, management, and ownership; (4) strategy: purpose, strategy, help to achieve, and assumption; and (5) marketing: sale, product, vendor, customer, and market.

Organization Ontology

Fox et al. [FBGL98] propose an organizational ontology for enterprise modeling. In their ontology, an organization consists of a set of *divisions*, a set of *subdivisions*, a set of *organization-agents*, a set of *roles* that the agents play in the organization, and an *organization-goal* tree that specifies the goals (and their decomposition into subgoals) that agents try to achieve. Moreover, an *organization-agent* plays one or more *roles*, where each role is defined along with a set of *goals* that the *role* is created to fulfill, and it is allocated with proper *authority* for fulfilling such goals. Moreover, *roles* might be *generalized* or *specialized* from one another.

Agents *perform* activities in the organization, and they may *consume resources* (e.g. materials, labors, tools, etc.). An agent can be a member of a *team* that have been set up in response to a special task. Moreover, agents have *skills*, and they have a set of *communication-links* that define their communications with other agents in the organization. *Communication-with-Authority* links can be used when communication is intended to create obligations, specifies the two agents, one in the authority position (called supervisor) and the other is the controlled position (called *supervisee*), among which communication takes place. While *authority relationship* is a control relationship between two

organizational agents, and it can be defined among agents in given organization roles.

2.1.3 Organizational Aspects in AOSE and MAS

Driven by organizations need, software agents and multi-agent systems received greater attentions lately. This attention was, mainly, because of the “autonomous agent” concept that both AOSE and MAS adopt. Jennings [Jen00] argued that agent-oriented approaches can significantly enhance our ability to model, design and build complex software systems. In this section, we review how organizational aspects were considered by AOSE and MAS communities.

Organizational Aspects in Agent-Oriented Software Engineering

In GAIA [ZJW03], organizations are composed of a set of roles, which are defined by four attributes: (1) responsibilities: define the functionality of the role; (2) permissions: are the rights which allow the role to perform its responsibilities. i.e., permissions identify the resources that can be used legitimately by a role; (3) activities: are computations that can be executed by the role; and (4) protocols: define the interaction between roles. However, GAIA does not make a separation between responsibilities and capabilities, in general a role might be responsible of some activity, but it does not have the capabilities to perform it. In such case, it depends or delegates the activities to another role. The notion of a role in GAIA gives an agent a well-defined position in the organization, with an associated set of expected behavior.

The main aim of the design process in GAIA is to transform the abstract models derived during the analysis stage into models at a sufficiently low level of abstraction that can be easily implemented. In particular, the design process in GAIA involves generating three models: (1) the agent model that identifies the agent types and the agent instances that will be instantiated from these types; (2) the services model that identifies the main services that are required to realize the agent’s role; and (3) the acquaintance model that documents the lines of communication between the different agents.

MESSAGE [CCG⁺02] is an agent-oriented software engineering methodology that extends UML with agent knowledge level concepts, and diagrams with notations for representing them. Most of the MESSAGE knowledge level entity concepts fall into 3 main categories: (1) ConcreteEntity (e.g., agent, organization, role and resource); (2) Activity (e.g., task, Interaction and InteractionProtocol); and (3) MentalStateEntity (e.g., Goal, InformationEntity, Message). MESSAGE can be used for the analysis and design of multi-agent systems (MAS). Moreover, MESSAGE distinguishes between high-level design phase and detailed design phase. In the high-level design phase, agent architecture is

defined, and roles are assigned to agents. Detailed design deals with computational representations of the entities modeled in high-level design. During this phase, the refinement process continues to determine how entities can be implemented.

Organizational Aspects in Multi-Agent Systems

Multi-agent Systems Engineering (MaSE) [DWS01] is a general purpose methodology for designing, analyzing, and developing heterogeneous multi-agent systems. In MaSE, each role is defined in terms of its interactions [Ken98], and it is responsible of achieving, or help in achieving specific system goals or sub-goals, and roles can be played by actors. Moreover, a single role may have multiple executing tasks that define the required role behavior. However, MaSE suffers from a main drawback that is the permission concept is missing, the authors consider that permissions are implicitly included along with the role's functionality, i.e., a role has the required permissions to perform the activities assigned to them, which is not true in general.

MOISE+ [HSB02] is an organizational model for Multi-Agent Systems that is based on three main concepts roles, role relations, and groups. *MOISE+* organizational model follow the organizational centered point of view, which can be composed by two core notions: an Organizational Specification (OS) and an Organizational Entity (OE). An Organizational Specification (OS) is formed by, (1) a Structural Specification (SS): that is defined at a three levels 1- individual level; 2- social level; and 3- collective level. Based on those definitions, the SS of a MAS organization is formed by a set of roles, a set of root group specifications, and inheritance relation; (2) a Functional Specification (FS): that is based on the missions and global plans concepts; and (3) a Deontic Specification (DS): that links the structural specification to functional specification. While an OE is a population of agents functioning under an OS, i.e., agents play roles defined in the OS, aggregated in groups instantiated from the OS groups, and behaving as normalized in the OS.

2.1.4 Organizational aspects in Requirements Engineering community

Three decades ago, the traditional view of RE focuses mainly on the target system. A different view start to emerge by the early 80's, which pays more attention to the environment of the system as to the system itself [GMB82; DHL⁺86], i.e., modeling the environment where the system operates has been recognized as an important part of the system design process. For example, RML [GMB82] was able to model the organizational environment where the system will eventually operates.

Since then, several RE researchers have proposed languages that are able to capture

the organizational aspects of the system-to-be. For example, in [YM94] an organization is viewed as being made up of social actors who are intentional entities, and they have motivations and beliefs, etc. Moreover, i^* [Yu95] and Tropos [BPG⁺01] offer concepts such as actors, roles, agents, positions, goals, tasks, resources, and social dependencies, which enables to capture several social and organizational aspects of the system-to-be.

2.2 Trust Management

The importance of trust in human societies is out of discussion [SS05], so it is in Socio-Technical Systems (STS) [ET60], where organizations and humans are an integral part of the system. Vulnerabilities of a STS are not only originated by technical issues, but they can be directly related to the social interactions of its components (both social and technical actors). Trust is complex and multidimensional concept that has been studied within many disciplines, including philosophy, psychology, sociology, transaction economics, and organization theory [CW03]. In particular, trust is a relation that is frequently defined in terms of the trustor's attitude toward the trustee, such as confidence [Luh88], credibility [DC97; Bai86], competence [Blo97], goodwill [Blo97; RSB⁺98], expectation [Blo97], belief [Blo97], positive intentions [LMB98], or future actions [Gam00; ZMP98].

There is a very rich literature concerning trust, in which we can find a wide variety of trust definitions. For example, Jøsang [Jøs97] defines trust in agents community as the belief that the agent will behave without malicious intent. Rousseau et al. [RSB⁺98] define trust as the psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another. Tseng and Fogg [TF99] define trust as a property of the recipient, such as dependability or reliability. Blomqvist and Stahle [BS00] define trust as an actor's expectation of another actor's competence, goodwill and behavior. While, Chopra and Wallace [CW03] define trust as the willingness to rely on a specific other actor, based on confidence that this trust will lead to positive outcomes. Finally, Jonker and Treur [JT99] define trust as the attitude an agent has, with respect to the dependability/capabilities of some other agent. In summary, an integrated definition of trust recognizes it as the union of three elements: the **trustor** that is the source of trust relation, where trust represents its expectations about the **trustee's** behavior and willingness to act on that expectation to fulfill the **trustum** that is the object of the trust relationship.

At the other hand, distrust has been characterized as a psychological state in which perceivers actively entertain multiple, possibly rival, hypotheses about the motives or genuineness of a person's behavior [FH94]. Lewicki et al. [LMB98] assert that both trust and distrust involve movements toward certainty, i.e., trust concerning expectations of

things hoped for and distrust concerning expectations of things feared of. Traditionally, scholars use to see trust and distrust as mutually exclusive concepts. However, this is not true, since trust and distrust are not totally separated concepts [LMB98]. In other words, you may have reasons to trust a trustee for a trustum, and you may have other reasons to distrust the same trustee for the same trustum. Lewicki et al. [LMB98] identify four different interrelations between trust and distrust: 1- low trust/low distrust, in which an individual or actor has neither reasons to be confident nor reasons to be wary and watchful, 2- high trust/low distrust, an actor has reason to be confident in another and no reason to suspect the other, 3- low trust/high distrust, in which an actor has no reason for confidence in another and ample reason for wariness and watchfulness, and 4- high trust/high distrust, in which an actor has reason to be highly confident in another in certain respects, but also has reason to be strongly wary and suspicious in other respects.

Considerable effort has been spend for identifying the bases of trust/ distrusts, i.e., how trust/ distrusts can be constructed. For example, Kramer [Kra99] stated that the bases of trust/ distrusts can be classified under: 1-dispositional trust; 2- history-based trust; 3- category-based trust; 4- role-based trust; and 5- rule-based trust. Moreover, Jones and George [JG98] stated that positive moods; emotions; and positive experience can enhance trust/ distrusts building. While Blomqvist and Stahle [BS00] propose a model for trust/ distrusts building, in which trust can be built based on 3 dimensions: competence; goodwill; and behavior. Finally, several researchers focused at building trust based on some related belief and assumptions. For example, Castelfranchi and Falcone [CF98] define seven kinds of beliefs for building trust: 1- competence; 2-disposition; 3-dependence; 4- fulfillment; 5- willingness; 6- persistence and 7- self-confidence belief.

Furthermore, a large body of literature has focused on computational trust. For instance, Marsh [Ste94] proposes one of the earliest trust models that consider direct interaction as a main source of trust related beliefs. Schillo et al. [SFR00] introduce a trust model based on probability theory, which can be used when trust among agents has a Boolean value (good or bad). While in Abdul-Rahman and Hailes [ARH00], trust can be build based on agents' belief in one another concerning their experience and reputational, and the degrees of trust range from complete distrust to complete trust. Esfandiari and Chandrasekharan [EC01] propose a cognitive based trust and reputation model. In which, trust can be built based on observation and interaction. Finally, Castelfranchi et al [CFP03] propose a cognitive trust model, in which different types of belief can be used to build trust.

At the other hand, trust is still a new research thread within the RE community. However, several RE approaches suggested concepts for capturing trust. For instance, in Giorgini et al. [GMMZ04] they focus on capturing trust at the level of roles. While in

Giorgini et al. [GMMZ05], they capture trust at two different levels (roles and agents), and they highlighted the problem that may arise when a trusted role is played by an untrusted individual (agent). In [Zan06] trust was introduced as a fundamental aspect for making decisions on security. While Asnar et al. [AGMZ07] refine the Goal-Risk framework by introducing the notion of trust for assessing risk. Finally, Chopra et al. [CPG11] introduce architectural trust that can be used to assist the trustworthiness of the overall STS.

2.3 Information and Information Quality

The first section starts by discussing information 2.3.1, clarifying its related concept, and then discussing information sources and information provenance. While in the second section 2.3.2, we start by discussing IQ along with its different dimensions. Moreover, we introduce IQ metrics and measurements that have been proposed in the literature, and we list and discuss several principles to be followed while developing IQ measurements. Finally, in section 2.3.3, we review how different communities have dealt with IQ needs. In particular, we review several proposals for improving IQ by design, and how different RE approaches have dealt with IQ requirements, and then we review the several technologies and mechanisms that have been proposed for dealing with IQ related issues.

2.3.1 Information

Information Quality (IQ) is a key success factor for organizations, since depending on low-quality information may cause severe consequences [Red95], or even disasters in the case of critical systems. However, to get better understanding of IQ, first we need to clarify what information is, and how it is related to other concepts such as “data” and “knowledge”, and then we briefly discuss information sources and information provenance.

Data, Information and Knowledge

The literature is rich with different attempts to define data, information, and knowledge along with their inter-relationships. Despite the lack of an agreed upon definitions of these terms, a general consensus exists that data, information, and knowledge are not the same thing [AN95; Ste01; DP01; CEH⁺09]. According to [Ack89] data can be defined as raw entity; and information is data that has been given a meaning. While knowledge is an appropriate collection of information to be used when needed. Aamodt and Nygard [AN95] define data as entities with no meaning, and they define information as interpreted data, i.e., information is data with meaning. Moreover, they define knowledge as learned

information, i.e., information existing in an agent's reasoning resources (it is the output of a learning process).

Not only the definitions of data, information and knowledge are arguable, but their interrelations are not always clear as well. Yet several researchers (e.g., [AN95; Ack89; DP01; BCM04]) have described this relationships as a hierarchical relation consisting of data at the bottom, followed by information, and knowledge on top, which implies that data can be transformed (interpreted) into information, which in turn can be transformed into knowledge (learning). However, Stenmark [Ste01] argued that the relation between data, information and knowledge is not linear in one direction, since we all on several occasions have used our knowledge to derive information, and created data out of information we have.

Information Sources

Generally speaking, information is produced by information sources. Buckland [Buc91] classify information based on its source under: (1) created internally: actors are able to produce information based on their knowledge, or they can elaborate it from information they have (e.g., a doctor set his daily schedule (created information) based on the activities he supposed to perform); (2) acquired from an objects: actors are able to obtain information that describes an object or one of its properties. (e.g., a doctor is able to acquire the patient temperature by different means); and (3) acquired by communications: information provided from one actor to another one (e.g., a doctor provides the patient with his date and time appointment).

At the other hand, humans (social actors) can serve as information sources, i.e., they are able to produce information based on information they have (elaboration), or derive it from their knowledge. Davenport and Prusak [DP01] stated that knowledge is embedded in our minds. Thus, we can instantiate some of this knowledge as explicit information [Ste01], i.e., such knowledge can be made tangible and represented as objects outside of the human mind [Ste01]. More specifically, Ackoff [Ack89] stated that the content of human mind can include: (1) data: symbols; (2) information that provides answers to "who", "what", "where", and "when" questions; and (3) knowledge: that is used to answers "how" questions.

Information Provenance (Lineage)

Information might not be used where it has been produced, i.e., it might be transferred, stored, etc., which might affect its quality. Thus, estimating its quality requires understanding the information chain (from origin to destination), and the rules that have been

Table 2.1: IQ dimensions in literature

	Accuracy	Completeness	Timeliness	Consistency	Believability	Accessibility	Trustworthiness
Ballou and Pazer [BP85]	X	X	X	X			
DeLone and McLean [DM92]	X	X	X				
Wand and Wang [WW96]	X	X	X	X			
Wang and Strong [WS96]	X	X	X		X	X	
Bovee et al. [BSM03]	X	X		X			
Kovac et al. [KLP97]	X		X			X	
Cykana et al. [CPS96]	X	X	X	X			
Liu and Chi [LC02]	X	X		X		X	X

applied to it as it moves along the chain, since different rules might have different effects over the quality of information [SC12]. In particular, to get better understanding of IQ, we should rely on what is called information provenance (also called “lineage”), which can be defined as any information that helps in determining the history of information product, starting from its original sources and the process by which it has been delivered to its destination [BKWC01; SPG05; SC12]. More specifically, information provenances is particularly important for analyzing IQ [dSDMM03], since the level of details included in the provenance enables to estimate IQ, and it has been studied in both database and data warehouse systems (e.g., Buneman et al. [BKWC01], Simmhan et al. [SPG05]).

2.3.2 Information Quality

In this section, we start by discussing IQ along with its different dimensions. Moreover, we introduce several IQ metrics and measurements that have been proposed in the literature, and then we list and discuss several principles to be followed while developing IQ metrics and measurements.

Information Quality Dimensions

Generally speaking, quality has been defined as “fitness for use” [JGB79], or as in [RB94] the conformance to specifications, i.e., meeting or exceeding consumer expectations. There is a general consensus that IQ is a hierarchical multi-dimensional concept [BP85; Red95; WW96] that can be characterized by different dimensions/ sub-dimensions including: accessibility, accuracy, completeness, timeliness, consistency, etc. [PLW02; BSM03; Jia10]). Table 2.1 lists different IQ dimensions that have been considered in the literature; in what follows we define each of these dimensions.

Accuracy: means that information should be true or error free with respect to some known or measured value [BSM03]. Some researchers use the term correctness in-

stead of accuracy. However, according to [WW96] these two terms seems to be equivalent;

Completeness: means that all parts of information should be available, and information should be complete for performing a task at hand [BSM03; WW96];

Timeliness: means to which extent information is valid in term of time (e.g., sufficiently up-to-date) [WW96; PLW02];

Consistency: means that multiple records of the same information should be the same across time and space [BSM03];

Believability: the extent to which information is accepted or regarded as true [WKM93; PLW02; BSM03];

Accessibility: means to which extent information is available, or easily and quickly retrieved [PLW02];

Trustworthiness: means to which extent information is credible [LC02].

Information Quality Metrics and Measurements

IQ dimensions enable to capture the quality of information with respect to stakeholders' requirements. Yet such dimensions must be associated with specific measures to evaluate IQ with respect to user requirements [FP04]. In what follows, we discuss several proposed measurements for IQ dimensions including: accuracy, correctness, completeness, timeliness and consistency. For instance, Redman [Red05] measure accuracy in a databases at two different levels: at the "field" (attribute) level accuracy is measured as: field level accuracy = number of fields judged "correct" / number of fields tested; while the "record" level accuracy = number of records judged "completely correct" / number of records tested. At the other hand, Parssian et al. [PSJ04] measure information accuracy in databases by considering a relation S that contains tuples captured for a predefined real world entity type. Each tuple in S is either accurate, inaccurate, or a mismember, which can be formally defined as:

- A tuple is *accurate* if all of its attribute values are *accurate*;
- A tuple is *inaccurate* if it has one or more *inaccurate* values for its non-identifier attributes, and no *inaccurate* values for its identifier attribute(s);
- A tuple is a mismember if it should not have been captured into S , i.e., a mismembership could be a tuple mistakenly included in the relation.

To understand the relationship between tuples in S and the underlying entity instances in the real world, they use the notion of a conceptual relation T , where T consists of tuples as they should have been captured in S if there were no errors of any kind (i.e., in an ideal world). Moreover, tuples in T belong to three categories that can be defined as follows: T_A , a set of instances in T that are correctly captured into S and thus remain accurate; T_I , a set of instances in T that are captured into S , and one or more of their non-identifying attribute values are inaccurate or null; and T_C , a set of instances in T that have not been captured into S and therefore form the incomplete dataset for S . Each instance in T_A corresponds to an instance in S_A . Similarly, instances in T_I and T_C correspond to instances in S_I and S_C , respectively. Based on the above definitions, they define the following quality metrics for a relation S .

- Accuracy of S , measured as $\alpha_S = |S_A| / |S|$, is the probability that a tuple in S accurately represents an entity in the real world.
- Inaccuracy of S , measured $\beta_S = |S_I| / |S|$, is the probability that a tuple in S is inaccurate.
- Mismembership of S , measured as $\mu_S = |S_M| / |S|$, is the probability that a tuple in S is a mismember.

At the other hand, Heinrich et al. [HKK07] measure the correctness of information by comparing the values of an information item in the Information System (IS) with its corresponding value in the real world. More specifically, consider w_L be a value of an information item within a database (IS), and w_R as the corresponding value of information in the real world. Moreover, they define $d(w_L, w_R)$ as a domain-specific distance function quantify the difference between w_L and w_R , which has two forms d_1 for character based values and d_2 for numerical values, both of them are defined as follows:

$$d_1(w_l, w_r) := \begin{cases} 0 & \text{if } w_l = w_r \\ \infty & \text{otherwise} \end{cases}$$

$$d_2(w_l, w_r) := |w_l - w_r|$$

And information correctness can be defined as follows:

$$Q_{corr.}(w_l, w_r) := \frac{1}{d(w_l, w_r) + 1}$$

Several measurements for information completeness have been proposed in the literature. For example, Pipino et al. [PWKR05] measure the degree of completeness of an

information item based on information items exists / information items that should exist. Moreover, Ballou and Pazer [BP03] measure both the structural and content completeness of information as follows: (1) Structural Completeness = values that are recorded / values that could have been recorded; (2) Content Completeness = content that is conveyed / content that could have been conveyed.

Considering the measurements of IQ time-related dimensions, Ballou et al. [BWPT98] propose an equation to measure information currency in information system (database), where currency is the time interval between information creation (or updated) to its usage time [WS96; PLW02]. They consider: (1) information delivery time: the time that an information item needs to be delivered to the customer; (2) information input time: represent the time when information was obtained in the system; and (3) age: how old information item was when it was received:

$$Currency = (DeliveryTime - InputTime) + Age$$

At the other hand, information timeliness can be defined as the extent to which information is sufficiently up-to-date for a task at hand [PLW02; WW96]. According to Ballou et al. [BWPT98], the timeliness of information can be measured depending on both its *currency* and *volatility* (how long the item remains valid):

$$Timeliness = \{max[(1 - \frac{currency}{volatility}), 0]\}^s$$

Where s is a parameter allows to control the sensitivity of timeliness to the currency /volatility ratio. Moreover, Heinrich and Helfert [HH03] measure information timeliness by considering both information mean update time along with information age:

$$Timeliness = 1/(mean\ update\ time).(age) + 1$$

Information Quality Measurement Requirements

To ensure consistency and usefulness assessment of IQ dimensions, Even and Shankaranarayanan [ES05; ES07] propose several principles to be followed while developing IQ dimensions measurements:

Representation Consistency : the measurement outcome should be easy to interpret by business users;

Interpretation Consistency : the measurement should have consistent semantic interpretation;

Aggregation Consistency : the calculation should be subject to some rules that guarantee its consistency, i.e., the aggregation cannot be higher than the highest quality level nor lower than the lowest;

Impartial-Contextual Consistency : the measurement should consider the contextual perception of the metrics (if any).

Moreover, Heinrich et al. [HKK07] refine the quality metrics requirements proposed in Even and Shankaranarayanan [ES05; ES07] by refining the representation consistency to requirements (R1) - (R3), while (R4) integrates the consistency principles (interpretation consistency and aggregation consistency). (R5) refers to the adaptively requirement metric that represent impartial-contextual consistency [ES07]. Finally, they propose one more property to capture the measurement procedure (R6). Each of these requirements is defined as follows:

R1. Normalization : an adequate normalization is necessary to assure that the values of the metrics are comparable;

R2. Interval scale : the difference between two levels of IQ values must be meaningful;

R3. Interpretability : demand the measurement being “easy to interpret by business users”, i.e., the IQ metric has to be easily understandable;

R4. Aggregation : stated that the aggregation consistency can be guaranteed only, if both of the interpretation consistency and aggregation consistency are guaranteed;

R5. Adaptivity : it is necessary that the metrics can be adapted to the context of a particular application;

R6. Feasibility : the metrics should be based on input parameters that are determinable.

2.3.3 Information Quality in the Literature

In this section, we review several proposals for improving IQ by design, we discuss how different RE approaches have dealt with IQ requirements, and then we review the relevant technologies and mechanisms for improving IQ that have been proposed in literature.

Improving Information Quality by Design

Several approaches for maintaining high quality information have been proposed in the literature, Jiang [Jia10] classify these approaches under two main types: (1) *Curative*

approaches: focus on detecting and correcting existing errors in information that might affect its quality [BS06]; and (2) *Preventive approaches* focus at preventing errors from occurring in the first place, or at least reducing the chance of their occurrence (e.g., information quality by design [Wan98]). Even no pure curative or preventive approach is able to address all IQ related problems [BS06], but preventing or reducing errors occurrence is more desirable, since the consequences of depending on low quality information might be too costly or their effects might be non-reversible (e.g., human causalities).

Improving IQ by design is not new; several researchers have proposed approaches that can be used for improving IQ by design. For example, Wang [Wan98] propose the Total Data Quality Management (TDQM) methodology, where the main purpose of TDQM is to deliver high quality information products (IP) to information consumers. TDQM was build based on the same idea of Wang proposed in his previous work [WSF95]. In their work, the concept of Information Product (IP) is introduced to emphasize the fact that the information output from an information manufacturing system has value that is transferable to the consumer. By depending on TDQM, organizations can develop a new information manufacturing system for the IP that allows for capturing many IQ requirements.

Moreover, Ballou et al. [BWPT98] presented an information manufacturing system that can be used to determine the data quality in terms of timeliness, quality, cost, and value of information products. In their work the information quality is a customer driven, i.e., the quality of the information products manufactured by the system is determined by the customer of information products. Shankaranarayanan et al. [SWZ00] extend the information manufacturing system model proposed by [BWPT98] to develop a formal modeling method for creating an IP-MAP. This model permits checking every raw input data item for data quality problems before it can be used in the production of an IP. While Ballou et al. [BWPT98] model allows raw input data items to be used without checking their quality first, and uses a quality block only when the raw input data items. Scannapieco et al. [SPP02] propose IP-UML, which is a software engineering approach developed to improve data quality in a single organization. The proposed approach relies on the IP-MAP framework [SWZ00], and IP-UML uses the Data Quality profile as the modeling language; which consists of three different models, namely: the Data Analysis Model, the Quality Analysis Model, and the Quality Design Model.

Information Quality in Requirements Engineering Area

Information Quality (IQ) modeling is an extension of traditional information modeling, where information modeling captures the structure and semantics of information, and IQ modeling captures the structural and semantic issues underlying IQ [WKM93]. Require-

ment Engineering community [BPG⁺04; SPP06; MG07] suggests concepts for modeling and analyzing information, yet they did not appropriately support modeling or analyzing IQ requirements, i.e., they either loosely define such requirements, or simply ignore them. For instance, SecureUML [BDL06] is a UML-based modeling language with a security modeling language for formalizing access control requirements. Yet it does not consider IQ related concerns, since it has been developed with a main reason to model access control policies.

Moreover, UMLsec [Jür05] that is an extension to UML modeling language, which allows for integrating security requirements modeling and analysis within the system development process. UMLsec is able to model security related features such as secrecy, integrity (IQ related aspect), access control, etc. It represents security feature on UML diagrams by providing several extension mechanisms, namely: (1) stereotypes: a new types of modeling elements that extends the semantics of existing types in the UML meta-model; (2) tagged values: that is used to associate data with model elements and (3) constraints: that are used to define criteria to determine whether requirements are met or not by the system design. In UMLsec, integrity is modeled as a constraint, which can restrict unwanted modification (e.g., insert), but information Quality can be affected in several other ways that cannot be captured by this approach.

Lin et al. [LNI⁺03b; LNI⁺03a] propose abuse frames based on Jackson's Problem Frames [Jac01]. In their approach, they introduce the notion of anti-requirement as the requirement of a malicious user, which can threaten an existing requirement of the system. A problem frame defines an identifiable problem class in terms of its context and the characteristics of its domains, interfaces, and requirements. Each frame describes a class of security violation that can be classified under: *interception*: that is used to address information confidentiality, *modification*: that is used to address information integrity related issues (e.g., unwanted information change), and *denial of access*: that is used to address situations where an attacker wants to make information unavailable to a user. Even abuse frame addresses the integrity problem (modification) by preventing unauthorized actors from modifying the data or prevent authorized actors from doing unauthorized modifications, yet information still can be modified in several other ways, i.e., abuse frames does not fully address integrity related issues.

The KAOS methodology [VLDM95] adopts the concept of *goal* to model the desired behaviors of the system, and the use *obstacles* to represent a set of undesirable behaviors. Therefore, *obstacles* are used as preconditions a designer has to overcome in order to achieve the goal. Furthermore, KAOS provides *obstacle prevention* and *obstacle mitigation* for resolving *obstacles*. Obstacle analysis is a recursive process since it may produce new goals for which new obstacles may be generated and resolved. However, obstacles seem

to be sufficient for modeling non-intentional obstacles to goals, but they appear limited for modeling and resolving intentional obstacles.

At the other hand, Van Lamsweerde et al. [VLBDLJ03] extends KAOS framework by introducing anti-requirement concept to represent the requirements of malicious attackers, and they introduce anti-goals that can be defined as the intentional obstacles to security goals. Moreover, they introduce the active attacker concept that can be modeled along with their own goals, capabilities, and the vulnerabilities that they can monitor or control. The anti-goal analysis starts by obtaining root anti-goals by negation of *confidentiality*, *privacy*, *integrity* or *availability* goals. Then for each of these anti-goals, we identify potential attacker. Finally, anti-requirements are defined in terms of the capabilities of the corresponding attacker. The preliminary elicitation of security-related goals is driven by generic specification patterns associated with each specialization of this goal class, namely, confidentiality, integrity, availability and privacy goals. Then these patterns are expressed in terms of abstractions from the language meta-model and security-specific language constructs. For each subclass, the instantiation of the corresponding specification pattern to “sensitive” objects found in the object model yields corresponding candidates for application-specific security goals (to be refined if necessary). For example, the specification pattern for integrity goals refers to meta-model elements such as Agent and Object to capture the definition of integrity.

Finally, Secure Tropos methodology [MG07] seems to be sufficient to capture the functional, security, privacy and trust requirements of systems. It is also able to capture the social and organizational context where the system will operate, but offers no concepts to capture IQ requirements of the system. In summary, all the previously mentioned languages, approaches and frameworks either ignore IQ requirements, or they do not properly capture them. Moreover, beside Secure Tropos, all of these approaches do not consider the social or the organizational context of the system-to-be, which is very important in our work.

Technical Solutions for Improving Information Quality

IQ can be improved at the technical level by ensuring information integrity. In what follows, we presents several technical solutions that have been proposed to avoid, detect, and correct information integrity violations in both storage and network transmission systems. For example, some systems use avoidance techniques that provide a certain level of integrity preservation for information they store, such as Read only Storage that is one of the simplest techniques to preserve the integrity of information in storage devices. However, these systems enforce read-only limitation that can prevent integrity violations due to user errors (e.g., Venti [QD02]), but information is still vulnerable to

hardware and software errors. Another example is the journaling file system [PADAD05] that can recover from a system crash by examining its log, where any pending changes are stored and then replaying any operations it finds there. This means that even after an unexpected shutdown, it is not necessary to scan through the entire contents of the disk looking for inconsistencies, i.e., the system just needs to figure out whatever has been added to the journal but not marked as done. However, journaling systems cannot protect information from malicious modifications or hardware bit errors, but it can ensure file system consistency without performing any explicit integrity checks for each file.

Beside information integrity avoidance techniques, some systems implement information integrity violation detection techniques. For examples, Checksumming [Coh87] is a well-known method for performing integrity checks, which can be computed for disk data and can be stored persistently. A checksum is a fixed-size datum computed from an arbitrary block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage. The integrity of information can be checked at any later time by re-computing the checksum and comparing it with the stored one. If the checksums match, information is likely to be unaltered. Moreover, mirroring [Bak85] is one of the simplest and oldest ways to implement integrity verification, which can be achieved by maintaining two or more copies of the same information in the storage device. Any integrity violation in one of the copies can be easily detected by comparing the copies. Usually mirroring can detect integrity violations, but it cannot always help in recovering the error, since it is not easy to determine which of the copies is correct.

Redundant Array of Independent Disks (RAID) is a storage technology that combines multiple disk drive components into a logical unit. Information is distributed across the drives in one of several ways called “RAID levels”, starting from RAID-3 then RAID-4, and RAID-5 [PGK88] parity bit (check bit) is used to detect errors in the stored information, which can verify information integrity across the multiple disks to a certain level. At the other hand, Cyclic Redundancy Check (CRC) [Bom08] is a powerful technique for error detection in information communications that can be easily implemented to obtain information integrity in network transmissions.

At the other hand, several integrity violation detection and correction techniques have been proposed. For instance, Error Correction Codes (ECCs) [Ham50] is an advanced form of parity detection that is used in servers and critical information applications. ECC modules use multiple parity bits per byte to detect double-bit errors. Some systems that support ECC can use a regular parity module by using the parity bits to make up the ECC code. Several storage disks today employ error correcting codes to detect and correct bit errors at the hardware level. Another example is Forward Error Correction (FEC) [Bie92], in which, the correction is performed at the receiver end using the check

bits sent by the transmitter. FEC uses the Reed-Solomon algorithm [MS81] to perform correction. Usually, FEC is used in network file systems. While, RAID-2 [PGK88] is able to detect and recover information integrity violation in storage devices. In RAID-2 the lost information (integrity violation) is recovered by reading the other components in a subset, including the parity component, and setting the missing bit to 0 or 1 to get the proper parity value for that subset.

It is well known that information integrity is a main issue in the databases area, where preserving information integrity in the database field means ensuring that information in the database is accurate, valid, and consistent [GA93]. Usually, the integrity of information in databases can be violated either by unwarranted changes to the database, or as a result of changes in the real world (inconsistency) [Mot89]. Generally speaking, databases are updated through *transactions*, when applying a *transaction* a database consistency might be falsified. [SV84] define four disciplines in database technology that are used to prevent such kinds of errors:

Security control that prevent users from accessing and modifying information in a database by unauthorized ways;

Concurrency control that prevent inconsistencies resulted from concurrent access of multiple users or applications to the database;

Reliability control that prevent errors due to the malfunctioning of system hardware or software;

Integrity control that prevent semantic errors made by users due to their carelessness or lack of knowledge; the integrity control use some integrity rules to verify the database and operations on the database.

Furthermore, databases incorporate *integrity constraints* (integrity controls) to ensure that the integrity of information is preserved. Grefen and Apers [GA93] classify *integrity constraints* based on their types under:

Domain and non-null constraints are used to specify a certain range of values for the attribute;

Referential integrity constraints are used to specify semantic links between the various relations in a database that should hold. Such constraints are a central issue in the relational databases;

Transition constraints constraints that are used to correctly evaluate a pre-transaction and post-transaction states of a database.

Temporal constraints are used to specify the temporal qualifiers for information in the database;

Fuzzy constraints that is used in fuzzy relational database systems to deal with fuzzy or incomplete data.

Moreover, Motro [Mot89] define two other types of *integrity constraints*, a *validity constraint* and *completeness constraint*, where both of them are used to ensure the validity of information in the database. However, integrity constraints are able to enhance information integrity in a database, but they cannot ensure it.

2.4 Chapter summary

In this chapter, we have reviewed the state of the art of several research areas related to this thesis. In 2.1, we have reviewed the main organizational and enterprise concepts that have been proposed by different research communities. In particular, we discussed several concepts in Enterprise Engineering area 2.1.1, in organization and enterprise ontologies 2.1.2, and then we discussed organizational concepts that have been proposed by Agent Oriented Software Engineering (AOSE) and Multi-Agent Systems (MAS) communities 2.1.3. Finally, we introduce the main organizational concepts that have been proposed by the RE community 2.1.4. Moreover, we reviewed the related work in Trust Management area in 2.2. While in section 2.3, we discussed information, IQ along with its dimensions, metrics and measurements, and then we reviewed how different communities have dealt with IQ needs.

Chapter 3

The Modeling Language

All models are wrong, but some are useful.

George Box

It is generally accepted in RE area that a system development requires models that represent the system-to-be [NE00], i.e., a modeling language is needed to describe the conceptual construct underlying the system [POB00], where a modeling language can be defined as the language that is used to construct and specify a software system [RJB04]. In other words, a modeling language can be used to describe concepts and constructs in the problem domain [Fow97], and it might be more useful for a modeling language to be graphical [Fow97; RJB04]. To this end, we propose a modeling language that adopts concepts from both Secure Tropos [MG07] and SI* [Zan06] modeling languages, and extends them with concepts and constructs for modeling IQ requirements of the system since the early phases of the system development.

In the remainder of this chapter, we introduce the objectives that underlie our modeling language in Section 3.1, and then we propose the principles that have been taken into consideration while developing the language in Section 3.2. In Section 3.3, we present our modeling language, we start by discussing the essential modeling constructs, and then we introduce more refined concepts for modeling trust. Finally, we propose concepts and constructs for modeling IQ requirements. Section 3.6 proposes workflow net with actors (WFA-net) that is a modeling extension, we propose, for modeling and analyzing IQ requirements in business processes. In particular, we introduce the semantics of the language, and we discuss how IQ requirements can be captured in their social and organizational context, and then how these requirements can be mapped into WFA-net. Finally in 3.7, we present a systematic process for dealing with IQ requirements.

3.1 The modeling language objectives

Our modeling language has been built with the following objectives in mind:

- O1:** the language should provide the required concepts and constructs for modeling the stockholders' IQ requirements in their social and organizational context. As previously discussed, leaving the social and organizational aspects outside the system's boundary leaves the system open to different kinds of vulnerabilities that might manifest themselves in the actor' interactions and dependencies.
- O2:** the language should provide the required concepts and constructs to refine top-level requirements into more detailed ones. In other word, if a requirement is too coarse to be satisfied, the language should provide the required constructs to refine them.
- O3:** the language should provide the required concepts and constructs to capture the relation between information, its stakeholders and its intended purpose of use at one hand, and required IQ dimensions at the other hand. In other words, the modeling language should provide clear semantics among information, its stakeholders, and its IQ dimensions.
- O4:** the language should provide the required concepts and constructs to capture the information structure taking into consideration its "fitness for use" for a particular purpose. The structure of information should be defined in a way that guarantee its completeness based on its purpose of usage, i.e., the extent to which information is applicable and helpful for the task at hand.
- O5:** the language should provide the required concepts and constructs for capturing IQ related aspects in actors' interactions. Usually, information might not be used where it is produced, i.e., it might be transferred, stored, etc., which might influence its quality. Thus, to get better understanding of IQ, the language should provide the required concepts to capture any information that can helps in determining the history of information, starting from its original sources and the process by which it has been delivered to its destination.
- O6:** the language should provide the required concepts and constructs to capture the Means-end relations between the different modeling constructs in order to identify whether a certain concept (e.g., goal, etc.) is achieved/satisfied or not.

3.2 The modeling language principles

The modeling language has been built based on several principles proposed by Paige et al [POB00]. In what follows, we list and define each of these principles:

Reliability: the language should be able to design a reliable system, i.e., it should be able to capture all the aspects of the system that the language has been developed to capture.

Uniqueness: the language should provide single construct to express every concept of interest, and it should avoid providing more than one. In particular, there should be no redundancy or overlapping among the concepts that the language proposes. The main aim of uniqueness principle is having a simple language that have a small but expressive number of clear concepts. Note that uniqueness does not mean minimizing the language concepts, but avoiding the use of unnecessary concepts. More specifically, a construct can be included in the language, only if, it is necessary for modeling a required concept, and if there is no other way of modeling such concept using an already existing construct.

Seamlessness: can be defined as the ability of mapping abstractions in the problem space to implementations in the solution space without changing notation [PO99a], i.e., deriving specifications from requirements should be consistent. Seamlessness provides several benefits, including: avoiding semantic gaps, mismatches between analysis and design, design and implementation, etc. [PO99b].

Consistency: the language constructs should meet the language design goals. In particular, any construct that is included (added) in the language should be justified by its contribution to the language purpose, and any construct that does not support the language purpose must be removed.

Simplicity: means that no unnecessary complexity should be included in the language; and it can be achieved by satisfying the previous three principles (e.g., uniqueness, seamlessness, and consistency).

3.3 The essential modeling concepts

Our modeling language adopts concepts from both Secure Tropos [MG07] and SI* [Zan06] modeling languages, and extends them with concepts and constructs for modeling IQ requirements in their social and organizational context. In the rest of this section, we present actors related constructs in 3.3.1, goals and information along with their different

relations in 3.3.2. Subsection 3.3.3, presents actors objectives and entitlements related constructs, and 3.3.4 introduces the different social relations among the actors, while in 3.4 we present their trust relations constructs. Finally, in 3.5, we propose IQ related concepts and constructs.

3.3.1 Actors, roles, and agents

An actor¹ represents an autonomous entity that has intentionality and strategic goals within the system. Basically, an actor covers two concepts, namely: a role and an agent, where the first can be defined as an abstract characterization of an actor in terms of a set of behaviors and functionalities within some specialized context, and the last can be defined as an autonomous entity that has a specific manifestation in the system. Agents can be a social agent as well as a software agent. An agent can play a role or more within the system [Yu95]. Moreover, our language adopts the modeling of role hierarchies based on the concept of *specialization*, i.e., a role can be a *specialization* of another one, and this relation can be represented by *is_a* relation. Figure 3.1 shows the graphical representation of actors, roles, and agents along with their interrelations.

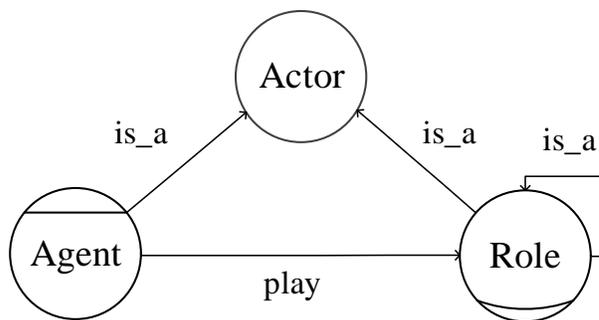


Figure 3.1: Graphical representation of actors, agents, roles, and their interrelations

Example 1. *in Figure 3.7, a stock trader (a role) is a person or company involved in trading securities in the stock market, and it is able to employ different strategies for making profit out of its trading activities. Moreover, we have different kinds of traders (e.g., fundamental trader; small trader, etc.), where they can be considered as a specialized (is_a) form of the stock trader role. Also we can identify Pro Trading (agent) that plays the role of stock trader.*

¹The notion of actor here is adopted from the Artificial Intelligence (AI) notion of software agent [GMP03]

3.3.2 Goals, information, and their relations

A goal can be defined as a state of affairs that an actor intends to achieved, and it is used to represent actors' strategic interests [BPG⁺04]. When a goal is too coarse to be achieved, it can be refined through and/or-decompositions of a root goal into finer sub-goals². Refining a root-goal into finer sub-goals through and-decomposition (shown in 3.2(a)) implies that the achievement of the root-goal requires the achievement of all its sub-goals. While or-decomposition (shown in 3.2(b)) is used to provide different alternatives to achieve the root goal, since or-decomposition allows for different alternatives for achieving the root-goal, i.e., achieving any of the sub-goals allows for achieving the root-goal.

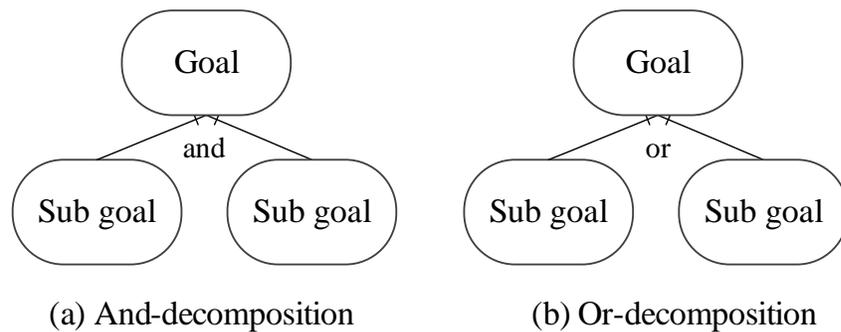


Figure 3.2: Graphical representation of goal, and its and/or-decompositions

Example 2. *in Figure 3.7, we can identify the top-level goal “Make profit by facilitating the trades among stock traders” that is and_decomposed into “Manage orders matching among traders” and “Ensure stable trading environment”. While the stock investor goal “Analyze the trading environment” is or_decomposed into “Trade by itself” and “Delegate trading activities to a trader”.*

Information represents any informational entity without intentionality. In Secure Tropos, they use the term resource to cover both physical and informational entities. However, in this thesis, we only consider informational entities (information), since our main concern is capturing IQ requirements. Goals may produce, read, modify, and send information, where each of these relations can be defined as follows:

Produce : indicates that an information item can be created by achieving the goal that is responsible of its creation process;

Read : indicates that a goal consumes an information item, and it can be strictly classified under:

²And/or refinement have been first introduced in AI domain [Nil71]

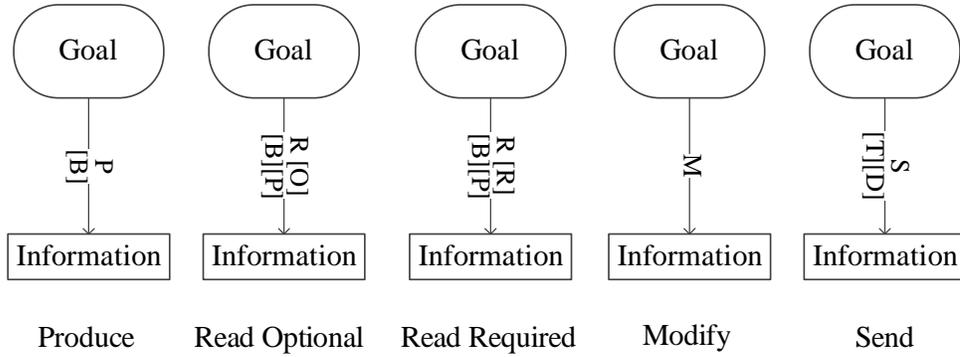


Figure 3.3: Graphical representation of goal-information relations

Optional : indicates that information is not required for the goal achievement, i.e., the goal can be achieved even without reading such information;

Required : indicates that information is required for the goal achievement, i.e., the goal cannot be achieved without consuming such information.

Modify : indicates that the goal achievement depends on modifying a particular information item;

Send : indicates that the goal achievement depends on transferring a particular information item to a specific destination under predefined criteria.

Example 3. *in Figure 3.7, NYSE (a stock market) produces “Trade information” by achieving the goal “Perform trades”, where such information describes the performed trades in the market, and the same information is read (required) $[R][R]$ by the goal “Analyze the trading environment”. Moreover, the goal “Receive orders from traders” reads (optional) $[R][O]$ “Trading orders”. While the stock investor’s goal “Produce and send trading orders” needs to send “Investor’s orders” to destination $[D]$ within a defined time period $[T]$. The different relations between goals and information are shown in Figure 3.3 as edges labeled with P (produce), R (read) that can be $R[R]$ (read required) and $R[O]$ (read optional), M (modify), and S (send) that has $[D]$ destination and $[T]$ time attributes³.*

3.3.3 Actors’ objectives and entitlements

Actors may have different relations with the previously mentioned modeling constructs; in what follows we discuss two main relations:

³We describe the other properties of these relations later in this section

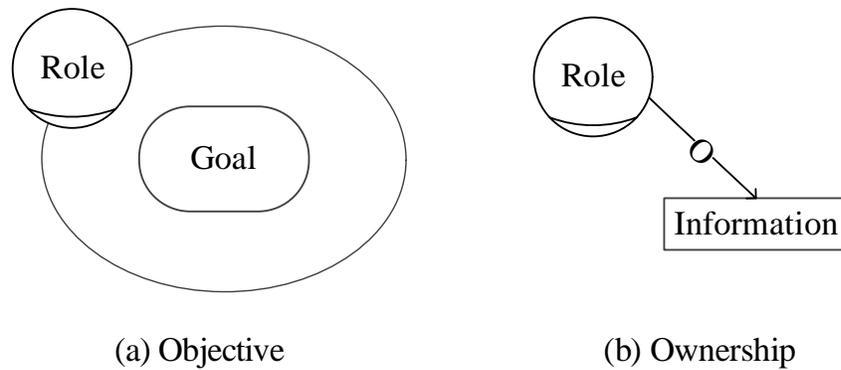


Figure 3.4: Graphical representation of actor's objective and information ownership

Objective : indicates that an actor aims to satisfy a goal, if the goal is represented within the actor's scope. Scope is represented as oval in Figure 3.4(a);

Ownership : indicates that an actor is the legitimate owner of an information item, where information owner has full control over the use of information it owns, i.e., it has the authority to control the delegated permissions over information it owns. The "ownership" relationship between an actor and information it owns is represented as edges labeled with O in Figure 3.4(b).

Example 4. *in Figure 3.7, a stock investor owns its own orders ("Investor orders"), and any goal appears within its scope belongs to its objectives (e.g., "Produce and send trading orders").*

3.3.4 Actors' social interactions

Actors may not have the required capabilities to achieve their own objectives by themselves (e.g., achieve a goal, provide an information item, etc.). Thus, they depend on one another. In what follows, we discuss the different actors' social interactions and dependencies.

Goal delegation

Our modeling language adopts the notion of goal delegation that has been proposed in SI* to model the transfer of objectives from one actor to another. In particular, delegation is a ternary relation among two actors concerning the delegatum (e.g., a goal), where the source of delegation called the delegator and the destination is called delegatee. Figure 3.5(a) shows the graphical representation of a goal delegation between two roles.

Information provision

Our modeling language adopts and extends the notion of information provision proposed in SI*, which indicates that an actor has the capability to deliver an information item to another one, where the source of the provision relation is the provider and the destination is the requester. However, the quality of the provided information might be affected by the provision process, which we discuss later in this section. Figure 3.5(b) shows the graphical representation of information provision between two actors.

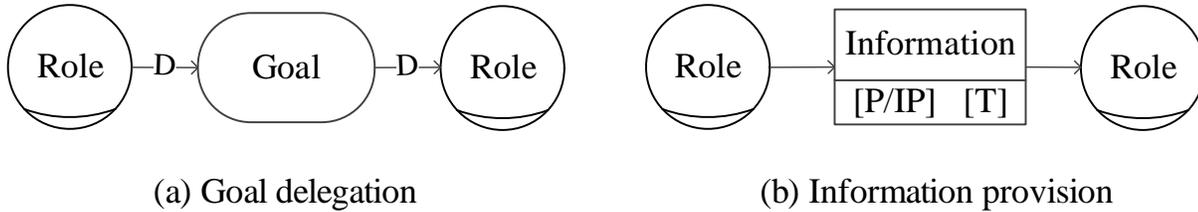


Figure 3.5: Graphical representation of goal delegation & information provision

Information permissions and permissions delegation

Generally speaking, a permission is a consent of a particular use of a particular object in a system [SCFY96], i.e., the holder of the permission is allowed to perform some action(s) in the system. In particular, permissions are used to restrict information usage to authorized actors only. In our work, information permission are classified under (P)roduce, (R)ead, (M)odify, and (S)end permissions, which covers the four relations between goals and information that our framework support. Moreover, permissions can be delegated among actors, where permissions delegation indicates that an actor delegates to another actor the right to produce, read, modify, and/or send permissions over a specific information item. Note that permission delegation should be under the control of the legitimate owner of information, since it has full control over information it owns. Figure 3.6 shows permissions delegation over information between two roles.

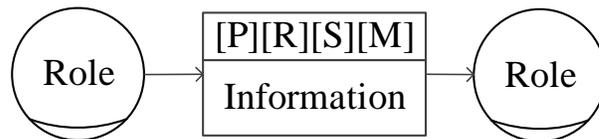


Figure 3.6: Graphical representation of permissions delegation

Example 5. in Figure 3.7, a stock investor delegates its goal “Make profit from trading securities” to a stock trader, and it provides “investor’s orders” to the trader as well, and

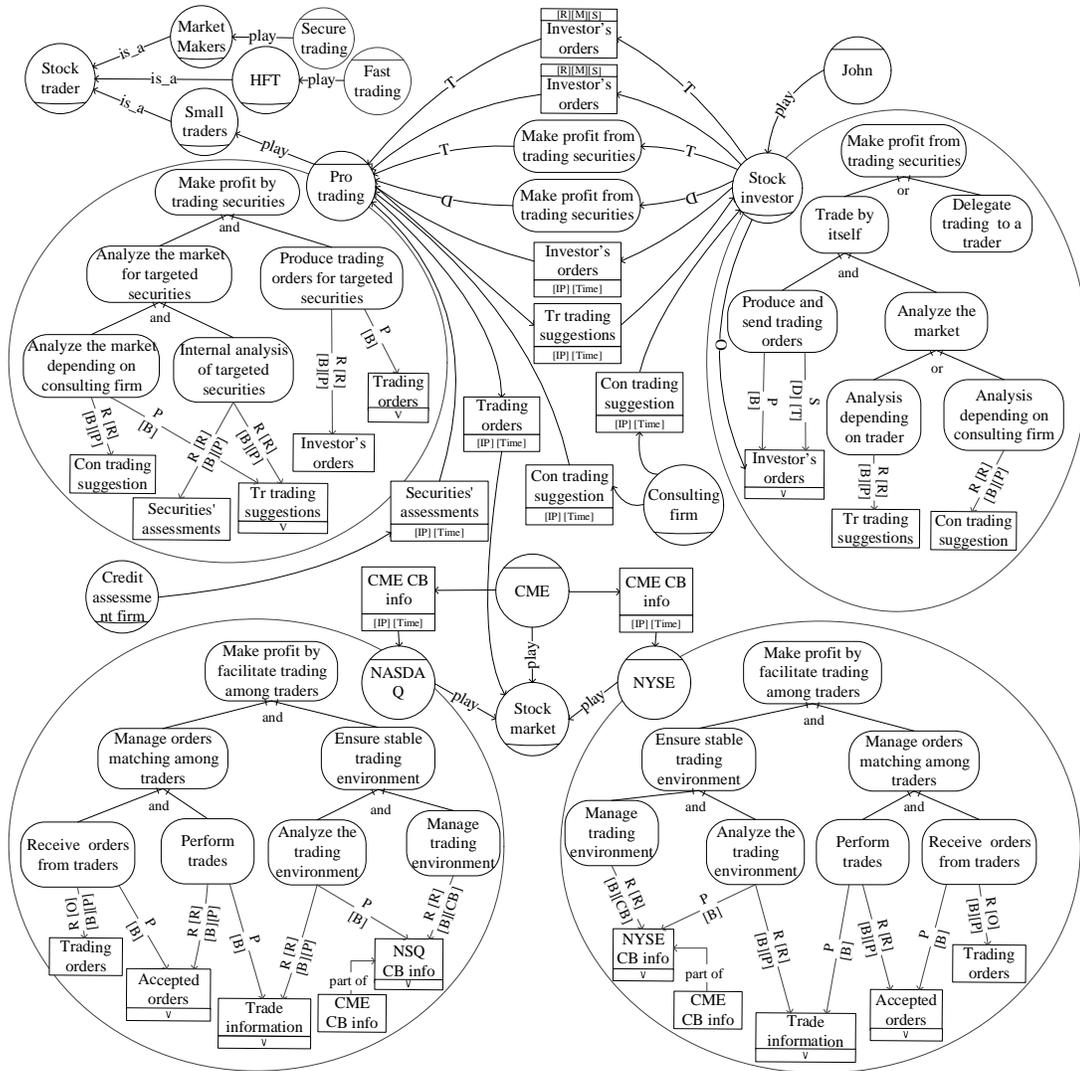


Figure 3.7: A partial goal model concerning the stock market structure

it also delegates read, modify and send permissions to the trader.

3.4 Social trust concepts

Trust is very important in human societies [SS05], so it is in Socio-Technical Systems (STS) [ET60], where organizations and humans are an integral part of the system. Vulnerabilities of a STS are not only originated by technical issues, but they can be directly related to the social interactions of its components (both social and technical actors). Our language adopts the notion of trust and distrust proposed in SI* to capture the actors' expectations

of one another concerning their delegated entitlements and authorities⁴, which can be defined as follows:

Trust : indicates the expectation of trustor that the trustee will behave as expected considering the trustum (e.g., trustee will achieve the delegated goal, or it will not misuse the delegated permission);

Distrust : indicates the expectation of trustor that the trustee will not behave as expected considering the trustum (e.g., trustee will not achieve the delegated goal, or it will misuse the delegated permission).

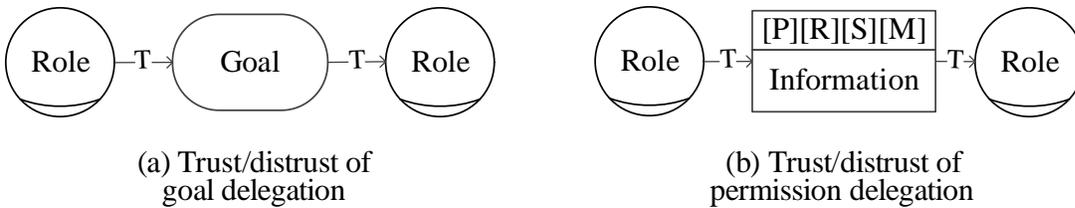


Figure 3.8: Graphical representation of trust/distrust of goals & permissions delegation

Trust/distrust relations between two roles concerning a goal and permission delegation are shown in Figure Figure 3.8(a) and 3.8(b) respectively.

Example 6. *in Figure 3.7, a stock investor trusts a stock trader for its delegated goal “Make profit from trading securities”, and it also trusts it for the delegation of read, modify and send permissions.*

3.4.1 Analyzing trust based on the actors’ internal structure

As previously discussed, several RE approaches (e.g., [GMMZ04; GMMZ05; Zan06; MG07]) propose constructs to model trust requirements, but they mainly focus on trust as social relations, without proposing any specific modeling technique to relate them with the internal requirements of the system’s components. In particular, none of these approaches proposes to analyze trust directly from the actors’ internal structure (capabilities and motivations). To this end, we enrich trust requirements analysis by acquiring information about the actors’ internal structure, and using such information to analyze trust, i.e., we propose analysis mechanisms to verify trust requirements consistency with the actors’ internal structure. In order to do that, we rely on actors’ competencies (can do)

⁴In SI*, they differentiate between trust of execution and trust of permissions. In this thesis, we do not differentiate between them

and motivations (will do) toward the trustum to clearly identify “why” an actor trust/distrusts another one.

However, relying only on the trustee’s competencies and motivations toward the trustum might not be enough for analyzing trust, since other factors might influence the trustee’s behavior. In particular, we need to define the different *intentional threats* [VL04] (*threat* for short) that might *threaten* the achievement of the trustum, and identify the trustee’s competencies and motivations toward such *threat(s)*. In other words, actors are intentional entities, and relying only on their competencies and motivation toward the trustum might not be enough for analyzing trust, since some of the defined *threats* might be within their objectives, i.e., the trustee aims to achieve them, which might *threaten* the achievement of the trustum. The graphical representation of thread along with *threaten* relation toward goal and information are shown in Figure 3.9(a) and Figure 3.9(b) respectively.

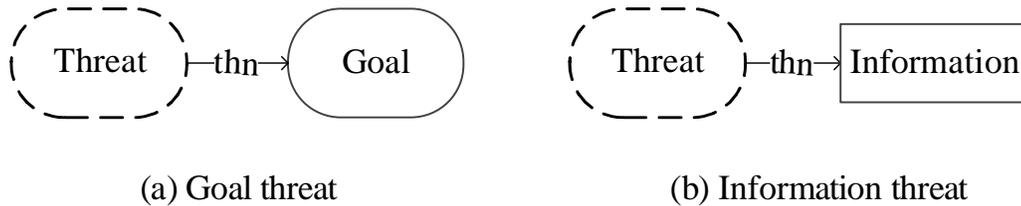


Figure 3.9: Graphical representation of goal/ information corresponding threat

Example 7. *in Figure 3.13, the Stock trader defines “Biasing the security assessment” as a threat to the delegated goal “Assessing the worthiness of companies’ securities”. Similarly, a stock market defines “Provide falsified/fraud orders” as a threat to the received “trader’s orders” information.*

To this end, an integrated analysis of trustee’s competencies and motivations toward the trustum along with its corresponding threat(s), if any, is required to analyze trust. In what follows, we show how role’s (trustee’s) competencies and motivations toward a trustum/threat can be classified and captured.

Roles’ competencies: we classify a role’s competencies toward a trustum/ threat under: (1) *Competence*: when the role has the capability of achieving the trustum/ threat; and (2) *Incompetence*: when the role does not has the capability of achieving the trustum/ threat. A role capability/incapability toward achieving a threat⁵ is modeled as an edges labeled with *cap/incap* respectively (Figure 3.10).

⁵We only model the actors’ capability/incapability toward achieving threats, since capabilities toward goals/ information can be captured by other concepts proposed in Secure Tropos and SI* languages.

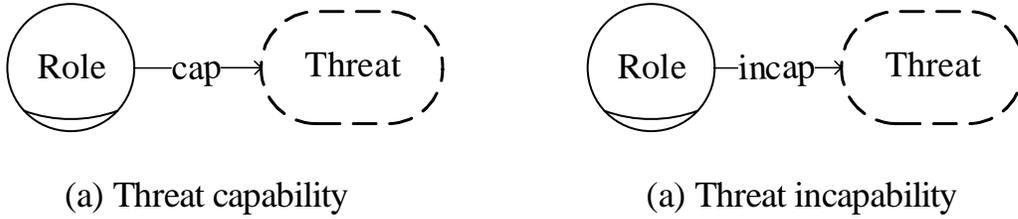


Figure 3.10: Graphical representation of roles' capability/incapability toward achieving a threat

Example 8. *in Figure 3.13, based on the roles they play, both Secure trading and Fast trading have the capability to achieve the threat “Provide falsified/fraud orders”. While Pro trading does not has such capability (incapable). Similarly, Star Co has the capability to achieve the threat “Biasing the security assessment”.*

Roles' motivations: we classify a role's motivations toward achieving a trustum/threat under: (1) *Positively motivated*: when there is evidence(s) that the role has positive intentions toward achieving the trustum/threat; and (2) *Negatively motivated*: when there is evidence(s) that the role has negative intentions toward achieving the trustum/threat. Roles' motivations toward achieving a trustum/threat can be derived from the different interrelations among their own objectives and a trustum/threat, which can be captured by relying on qualitative goal relationships [GMNS03], in which a goal can contributes positively or negatively towards the achievement of another goal. In particular, an actor is positively motivated to achieve a trustum/threat, if such trustum/threat contributes positively to at least one of its own goals. At the other hand, an actor is negatively motivated to achieve a trustum/threat, if such trustum/threat contributes negatively to at least one of its goals.

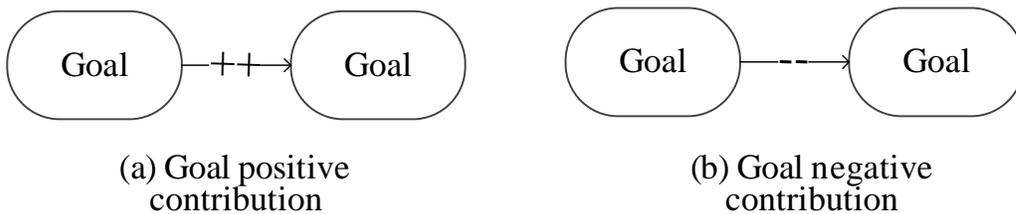


Figure 3.11: Graphical representation of goals positive and negative contributions

Positive and negative contribution among goals are modeled as edges labeled with ++ and -, and they are shown in Figure 3.11(a) and Figure 3.11(b) respectively. Similarly, positive and negative contribution between a threat and a goal are modeled as edges labeled with ++ and -, and they are shown in Figure 3.12(a) and Figure 3.12(b) respectively.

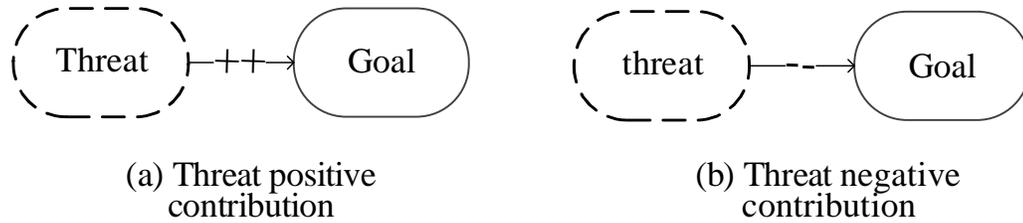


Figure 3.12: Graphical representation of threat-goal positive and negative contributions

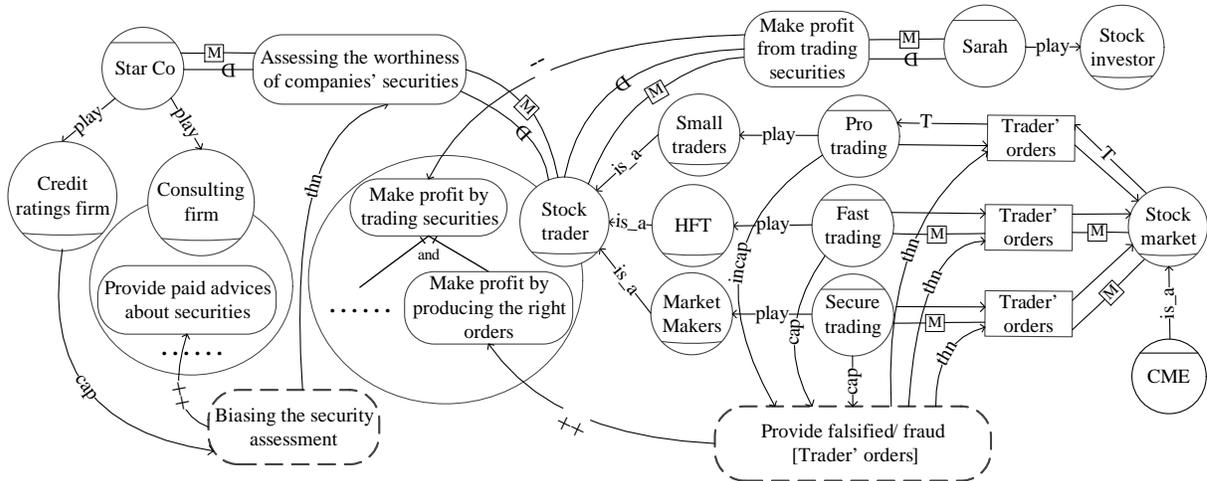


Figure 3.13: A partial goal model concerning trust analysis based on the actors' internal structure

Example 9. in Figure 3.13, Sarah's goal "Make profit from trading securities" contributes negatively to the stock trader's goal "Make profit by trading securities". At the other hand, the threat "Provide falsified/fraud orders" contributes positively to the stock trader's goal "Make profit by producing the right orders". Similarly, the threat "Biasing the security assessment" contributes positively to the consulting firm's goal "Provide paid advices about securities".

3.4.2 Trust and Distrust Analysis

In the previous sections, we discussed how actors' competencies and motivations toward trustum/threats can be derived. In this section, we show how trust relations can be analyzed based on actors' competencies and motivations (shown in Table 3.1).

Trust: an actor *trusts* another one for a specific trustum, if the trustee have the *competency* and *positive motivations* toward achieving the trustum, and there is no evidence(s) about the trustee's *incompetency* neither *negative motivations* toward achieving the trustum. Moreover, the trustee should not have the *competency* toward achieving the

Table 3.1: Analyzing trust based on the actors' internal structure

	Trustum				Threat			
	Competence		Motivation		Competence		Motivation	
	Comp	Incomp	Positive	Negative	Comp	Incomp	Positive	Negative
Trust	✓	✗	✓	✗	✗	✓	-	-
Distrust	-	✓	-	-	-	-	-	-
	-	-	✗	✓	-	-	-	-
	-	-	-	-	✓	✗	✓	✗

trustum related threat (if any).

Example 10. *in Figure 3.13, a stock market can trust small traders for “trader’s orders”, since they have the competencies and positively motivated to provide them, and they are incompetence for achieving the related threat.*

Distrust: an actor *distrusts* another one for a specific trustum, if:

- (1) The trustee does not have the *competency* toward achieving the trustum⁶.
- (2) There is no evidence(s) about the trustee’s *positive motivations* toward achieving the trustum, and there is evidence(s) about its *negative motivations* toward the trustum.

Example 11. *in Figure 3.13, Sarah distrust the stock trader for its goal “Make profit from trading securities”, since such goal contributes negatively to the stock trader’s goal “Make profit by trading securities”.*

- (3) There is evidence(s) about the trustee’s *competency* and *positive motivations* toward achieving the trustum related threat, and there is no evidence(s) about the trustee’s *incompetency* and *negative motivations* toward achieving the threat.

Example 12. *in Figure 3.13, a stock market distrusts both Secure trading and Fast trading for “trader’s orders” information, since they have the capability to achieve the related threat “Provide falsified/fraud orders”, and they are positively motivated to achieve such threat (contributes positively to their goal “Make profit by producing the right orders”). Similarly, stock trader distrusts Star Co for its delegated goal “Assessing the worthiness of companies’ securities”, since Star Co have the capability to achieve the related threat*

⁶This situation is handled by a logical constraint that prevents delegating a goal to actors, who do not have the capability to achieve the trustum

“*Biasing the security assessment*”, and it is positively motivated to achieve such threat (contributes positively to its goal “*Provide paid advices about security*”). *Enron scandal [PS03]* is a famous example about biasing the security assessment.

Monitoring can be defined as the process of observing and analyzing the performance of an actor in order to detect any undesirable performance [GZF04]. According to [GJKL01; Zan06], the lack of trust or distrust can be compensated by monitoring. Monitoring is modeled as an edge labeled with **M** between two roles concerning the subject of monitoring. The graphical representation of monitoring concerning a goal and information are shown in Figure 3.14(a) and Figure 3.14(b) respectively.

Example 13. *in Figure 3.13, a stock market can compensate the lack of trust in both Secure trading and Fast trading for “trader’s orders” information, by monitoring such information. Similarly, stock trader can compensate the lack of trust in Star Co for its delegated goal “Assessing the worthiness of companies’ securities” by monitoring the goal.*

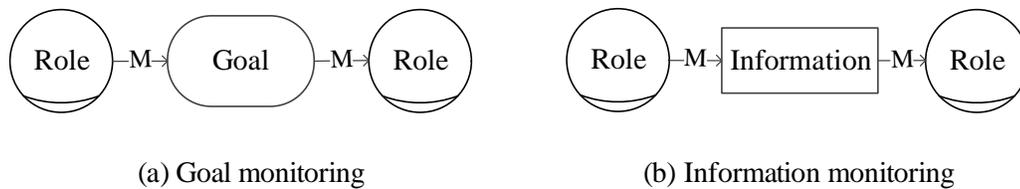


Figure 3.14: Graphical representation of goal/information monitoring

3.5 Information Quality concepts

In this section, we introduce a new model for analyzing IQ based on seven IQ dimensions 3.5.1, and then we introduce our proposed modeling concepts and constructs for modeling IQ requirements 3.5.2.

3.5.1 Multi-dimensional Model for Analyzing Information Quality

As previously mentioned, IQ is a hierarchical multi-dimensional concept [BP85; WW96; BSM03], which can be characterized by different dimensions, including: accessibility, accuracy, completeness, consistency, trustworthiness, etc. [Red95; PLW02; BSM03; Jia10]. Although there exist several models for analyzing IQ based on its different dimensions [LC02; PLW02], yet there is no general consensus on which dimensions should be considered or not for analyzing IQ, i.e., most of these works do not theoretical justify why a certain dimension should be considered or not for analyzing IQ [LC02].

Moreover, several attempts to categories IQ dimensions into different groups have been proposed. For example, Wang et al. [WS96] categories IQ dimensions into four groups: (1) Intrinsic IQ: considers IQ dimensions that is related to the consumer needs (e.g., accuracy, believability); (2) Contextual IQ: considers IQ dimensions related to the contextual aspects of the task at hand (e.g., relevancy, timeliness, completeness); (3) Representation IQ: considers IQ dimensions related to the IQ representation and understood by its consumers (e.g., representational consistency, interpretability); and (4) Accessibility IQ: considers IQ dimensions related to accessible to information consumer (e.g., security). The same four categories were adopted by [SLW97; LC02], but they did not consider the same IQ dimensions as in [WS96]. Similarly, there was no theoretical basis that justifies why those researchers classify IQ dimensions under just only four categories [LC02], why not three or five?

Not only the classification of IQ dimensions is not clear or appropriately justified, but also there are inconsistencies in the interrelations among the IQ dimensions that these works consider. For example, relevance has been considered as an IQ dimension in [BSM03], while it has been considered as a sub-dimension of usefulness in [WRK95]. Further, completeness has been considered as a sub-dimension of believability in [WRK95], while it has been considered as sub-dimension of integrity in [BSM03]. In addition, most of these approaches intend to propose a holistic method for defining IQ along with its sub-dimensions (one size fits all), i.e., they consider the same dimensions for analyzing IQ (user-centric view) without taking into consideration the relation between information and its purpose of use.

Furthermore, most of these approaches were not designed to capture the needs of socio-technical systems. More specifically, they mainly focus on the technical aspects of IQ dimensions, and ignore the intentional, social and organizational aspects that might underlie some of these dimensions. Ignoring such aspects leaves the system open to different kinds of social and organizational vulnerabilities that might manifest themselves in the actors' interactions and dependencies.

To tackle the previously mentioned problems, we need to propose a model for analyzing IQ in terms of its different dimensions based on information purpose of use. Moreover, the model should be able to capture the intentional, social and organizational aspects that might underlie these dimensions. To this end, we propose a multi-dimensional model (Figure 3.15) for analyzing IQ based on seven IQ dimensions: accessibility, accuracy, believability, trustworthiness, completeness, timeliness and consistency. We chose these dimensions based on the following criteria. Although there is no general agreement on the identification of the most important IQ dimensions, it is possible to distinguish several IQ dimensions that have been considered in most of the IQ models; including: accuracy,

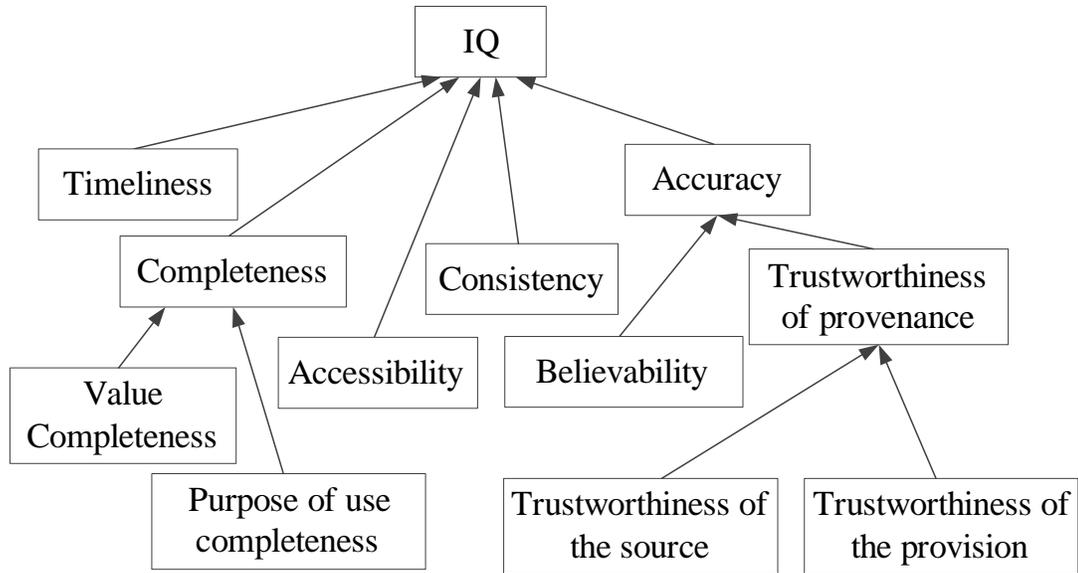


Figure 3.15: A multi-dimensional model for analyzing IQ from social perspective

completeness, timeliness and consistency [BS06; CCRC13]. In addition, we consider both information believability and trustworthiness, since they can be used to analyze information accuracy. Finally, before thinking about information accuracy, completeness, etc., we need to have information and the required permissions over it to perform a task at hand. Thus, we consider information accessibility in our model.

In the rest of this section, we define these dimensions along with their interrelations, and we discuss how each of them can be analyzed from different social perspectives (e.g., producer, reader, modifier, and sender perspectives⁷).

- 1. Accessibility:** the extent to which information is available, or easily and quickly retrieved [PLW02]. In [BS06], they differentiate between accessibility as information available, and accessibility as access security to information that can be restricted by permissions. While accessibility in our work is defined as information availability along with the required permissions over it to perform a task at hand.

Example 14. *A trader might need to reads, modifies, and/or sends trading orders belong to its investors, yet it is not able to perform such activities unless it has the required permissions from investors (information owners). Note that accessibility is analyzed in the same way from all the different perspectives of information users (e.g., producer, reader, modifier, and sender).*

⁷In our framework , produce, read, modify and send are the main relation between a stakeholders and information they use

2. **Accuracy:** means that information should be true or error free with respect to some known, designated or measured value [BSM03]. Accuracy is the most important and studied IQ dimension. Yet without a clear standard, analyzing accuracy is not an easy task, i.e., determining whether information is accurate or not. However, Dai et al. [DLBK08] stated that information accuracy is highly influenced by information *trustworthiness*. While Wang and Strong [WS96] argued that accuracy can be analyzed based on several dimensions including *believability*. Thus, we analyze accuracy based on two sub dimensions, namely: *trustworthiness* and *believability*.
3. **Believability:** to which extent information is accepted or regarded as true [WKM93; PLW02; BSM03].
4. **Trustworthiness :** can be defined as to which extent information is credible [LC02]. We rely on the *trustworthiness of the provenance* to analyze trustworthiness, i.e., trustworthiness of information is analyzed depending on the (*trustworthiness of the source*), and the (*trustworthiness of the provision*) [DLBK08]:

Trustworthiness of the source: the trustworthiness of the source can be analyzed from information reader perspective based on the source capability to produce fraud, falsified, or harmful information that might compromise the system performance. In other words, information source is untrusted if it has the capability to produce such information, otherwise it is trusted;

Trustworthiness of the provision: the trustworthiness of information provision can be analyzed based on the way information arrives to its destination, and the operations that have been applied to it taking into consideration whether such operations were authorized or not (e.g., permissions and trust) [GG15a], and it can be used to guarantee that information has not been corrupted nor modified during its transfer from its source to its destination.

Analyzing accuracy from information producer perspective: information producer requires that the produced information is the same as the intended one, and it analyzes information accuracy based on its believability and trustworthiness dimensions.

Example 15. *consider a stock investor that aims to produce a trading order by itself. In order to analyze the order accuracy, it analyzes the believability and the trustworthiness of such order. More specifically, considering believability for the produced orders enables to avoid producing inaccurate information unintentionally (e.g., fat finger error that is a human error caused by pressing the wrong key when using a computer to input information).*

While considering trustworthiness enables to avoid producing inaccurate information intentionally. For instance, investor analysis accuracy based on its believability only, if it produces orders by itself, since it trusts itself for producing such orders. While if the investor delegate (common in socio-technical system) order producing to a trader, a trust relation between them concerning the produced information should hold, since trader is an intentional actor as well, and it might not have the same intentions as the producer, i.e., trader might produce orders different from the ones intended by the investor.

Analyzing accuracy from information sender perspective: information sender requires that information it sends to be accurate at its destination, and it analyzes information accuracy based on its trustworthiness (trustworthiness of the provision) dimension only.

Example 16. *to guarantee that the trading orders are accurate at their destination (e.g., a trading market), investor (information sender) should guarantee that such orders have been provided by trusted means, which guarantee that its orders have not been intentional/unintentionally compromised. For instance, if the investor provides such orders through its trader, he should be sure that orders will not be modified in unauthorized way, and the trader will not misuse its permissions for compromising the orders, i.e., a trust relation for such permissions between the investor and the trader should hold.*

Analyzing accuracy from information reader perspective: information reader analyzes information accuracy based on its believability and trustworthiness of the provenance (trustworthiness of the source and provision).

Example 17. *in order to fulfill their obligation (facilitate trading), Market Makers (stock traders) provide what is called “stub quotes”, which are orders with prices far away from the current market prices, i.e., such orders can be considered as inaccurate/fraud/ falsified orders. During the Flash Crash, over 98% of the trades were executed at prices within 10% of their values because of such orders [KKST11]. However, such failure could be avoided, if markets apply a mechanism to analyze the believability of the received orders. At the other hand, some HFTs (stock traders) provide inaccurate/fraud/ falsified orders to affect the prices of some securities before starting their real trades (e.g., flickering quotes that are orders last very short time, which make them unavailable for most traders). If markets analyze the trustworthiness of the provenance of the received orders, they will be able to detect such orders and apply the required mechanisms to mitigate their harmful effect. In particular, by considering the trustworthiness of the source, markets can guarantee that the received orders are not provided by harmful traders. Moreover, by considering the trustworthiness of the provision, markets can avoid situation in which traders may deny*

to commit to their performed orders by claiming that such orders either were not issued by them, or they were modified by unauthorized entities or corrupted during their delivery to the market.

5. Completeness: means that all parts of information should be available, and information should be complete for performing a task at hand [BSM03]. Thus, completeness can be analyzed depending on two sub dimensions:

Value Completeness: information is preserved against corruption or lost that might endanger its integrity (e.g., during its storage/ transfer).

Purpose of use completeness: information is complete for performing a task at hand, i.e., all the required information for performing a specific task should be available. Usually, the *purpose of use completeness* is harder to be analyzed, and it requires a domain knowledge.

Information completeness can be analyzed from two different perspectives (sender, and reader⁸):

Analyzing completeness from information sender perspective: information sender analyzes information completeness only based on its value completeness, since send information is complete for the purpose of send.

Example 18. *to guarantee that the trading orders are complete at their destination (e.g., a trading market), investors (information senders) should ensure that their orders will be provided by a trusted means that guarantee their orders have not lose any of their parts during the transfer and became incomplete.*

Analyzing completeness from information reader perspective: information reader analyzes information completeness depending on its two sub dimensions, i.e., value and purpose of use completeness.

Example 19. *a main reason of the Flash Crash was the lack of coordination among the Circuit Breakers (CBs)⁹ of the trading markets. In particular, markets depend only on their own CBs information to stabilize their trading environment. However, such information is complete for each market alone, but when it comes to coordinate the CBs activities among all the markets, it can be considered as incomplete for the purpose of*

⁸We do not analyze it from the producer perspective, since its related issues do not arise in produced information

⁹CBs are techniques that are used by markets to slow down or halt trading for pre-defined period of time in specific cases in order to prevent a potential market failure [GHLZ12]

use. For instance, during the Flash Crash CME market employs its CB, but NYSE did not [Sub13], since each of them depends only on its CB information.

6. Timeliness (validity): means to which extent information is valid in term of time (e.g., sufficiently up-to-date) [PLW02]. According to [BWPT98], information timeliness can be analyzed depending on information *currency (age)* that is the time interval between its creation (or update) to its usage time [WS96; PLW02], and information *volatility* that is the change frequency of information value [WS96], i.e., information is not valid, if its currency is bigger than its volatility interval, otherwise it is valid.

Information timeliness can be analyzed from two different perspectives (sender, and reader¹⁰):

Analyzing timeliness from information reader perspective: information reader analyzes information timeliness based on information *currency*, and information *volatility*, i.e., information is not valid, if its currency is bigger than its volatility interval when it is read, otherwise it is valid.

Example 20. *a stock trader should guarantee the timeliness (validity) of his investors' financial portfolio before accepting to perform any of its orders, i.e., the portfolio currency is bigger than its volatility.*

Analyzing timeliness from information sender perspective: information sender requires that its information to be valid at its destination in terms of time. In particular, it defines the allowed amount of time for information (send time) to reach its final destination, and information is valid from the sender perspective if its currency at destination is less than the defined send time.

Example 21. *to guarantee the validity of its orders at the intended market, investor should ensure that the trader it depends on provides a time to market that fits its needs (less than its send time).*

7. Consistency : means all multiple records of the same information should be the same across time and space [BSM03]. In this thesis, *consistency* is a time related aspect, i.e., the value of information among its different users might became inconsistent due to time related aspects (e.g., currency). While in [WS96] consistency is used to refer to “representational consistency”.

¹⁰Timeliness related issues do not arise in produced information

Analyzing consistency from information reader perspective: information consistency arises only when there are multiple records of the same information that are being used (read) by several actors for *interdependent purposes (goals)*, and we call such actors as *interdependent readers*. While if actors use the same information for independent purposes, inconsistency will not be an issue since the actors' activities are independent. In particular, to ensure consistency among the different *interdependent readers*, we need to ensure that they depend on the same information value in term of time.

Example 22. *the lack of coordination among CB activities of the trading markets will not be resolved unless markets (interdependent readers) depend on consistent information for their CB activities.*

3.5.2 Information Quality constructs

Secure Tropos and SI* modeling languages do not provide specialized concepts or constructs that enables to model IQ related aspects. From this perspective, we propose several new concepts and constructs to model IQ related aspects. In the rest of this section, we discuss each of the proposed concepts.

Information accessibility: can be influenced by the permissions that an actor has over information, which might enables or prevents it from using information as intended. As previously mentioned, we propose four types of permissions concerning the four types of information usage that our framework supports (e.g., (P)roduce, (R)ead, (M)odify and (S)end)¹¹.

Information completeness: means that all parts of information should be available, and information should be complete for performing a task at hand [BSM03]. Thus, it can be subject to:

(1) **Value completeness:** information provision might affect the quality of the transferred information. Thus, we extend information provision concept with a property that define its type, which can be strictly classified under:

Normal provision (P) : does not guarantee the integrity of the provided information against corruption or lost;

Integrity Preserving (IP) provision: guarantee the integrity of the provided information against corruption and lost, i.e., the completeness of information is preserved [GG13a].

¹¹Permissions and permissions delegation have been discussed earlier

Example 23. *in order to guarantee that its information (“CME CB info”) will be preserved against corruption and lost, CME provides it to both NYSE and NASDAQ through IP provision. Provision type (e.g., P/IP) is added as an attribute to the information provision construct as shown in Figure 3.7.*

(2) Purpose of use completeness: we rely on “part of” concept that has been used in several areas (e.g., natural language [Bri95], conceptual modeling [Ode98], etc.) to model the relation between a composite information item and its sub-items. Such relation enables to decide whether information is complete for the purpose of use or not. Note that the completeness of information is subject to task it is intended to be used for, i.e., the structure of information is not always easy to be defined.

Example 24. *a main reason of the Flash Crash was the lack of coordination among the Circuit Breaker (CBs) of the trading markets. More specifically, markets depend only on their own CBs information to stabilize their trading environment. However, such information is complete for each market alone, but when it comes to coordinate the CBs activities among all the markets, such information can be considered as incomplete. In particular, in stock market domain the same security can be traded in several markets, but it will always have only one primary listing market that is the main market for trading the security, such market has full control over the way its listed security can be traded in other markets. Considering the Flash Crash, CME is the primary listing market, and it requires that all CB activities related to its securities to be coordinated with its own activities, i.e., if CME turn to slow trading mode or stop trading, all markets trade the same security should do the same. This can be solved, if we consider that the CB information of any market trading the CME securities is composed of its own CB information along with the CME CB information. Part of relation is shown in Figure 3.7 as edges labeled with *part_of* between “NYSE CB info” and “CME CB info”.*

Information timeliness (validity): to model timeliness, we extend information construct with a *(V)olatility* attribute to represent the change rate of information value [WS96] (shown in Figure 3.7 for produced information). Information might not be used where it is produced (it might be transferred). Thus, we extend *information provision* construct with a time attribute (shown in Figure 3.7) that captures the amount of time an information provision requires (referred to as the transmission time in telecommunication networks [For06]). Since, we already defined two different relations between goals and information that can be affected by time related aspects

(e.g., reads and sends), we extend them to accommodate time related aspects, as follows:

Read timeliness: in order to capture information validity for read, we propose *read time*¹² concept that capture the actual *usage time* of information, and it enables to determine information *currency (age)*, which can be used to analyze the timeliness of read information, i.e., information is valid, if its currency is smaller than its volatility, otherwise it is not invalid.

Send timeliness: in order to capture information validity at its intended destination, we extend *send* relation with two attributes: (1) *(D)estination* that represents the intended send destination of information; and (2) *(S)end time* that represents the allowed amount of time for information to reach its final destination. Both of these attributes are shown in 3.7. The timeliness of send information at its destination can be analyzed depending on the *send time* and destination *read time*, i.e., information is valid, if *read time* is smaller than the *send time*.

Information consistency: as previously mentioned, consistency arises only when there are multiple records of the same information, which are being read by actors for *interdependent purposes (goals)*. In order to capture consistency, we extend the *read* relation between a goal and information with *Purpose Of Use (POU)* attribute that captures the intended purpose of information usage, which enables to identify *interdependent readers* that are actors who read the same information for the same *POU*. To this end, consistency can analyzed among *interdependent readers* based on their *read times*, i.e., information is consistent among its *interdependent readers*, if all of them have the same *read time*, otherwise it is inconsistent.

Example 25. in Figure 3.7, NYSE and NASDAQ are reading “CME CB info” for the same purpose of use (CB), i.e., they are *interdependent readers* for such information. Thus, in order to guarantee that “CME CB info” is consistent among them, both of them should have the same *read time*.

Information believability: is considered in both *read* and *produce*, since only these two relations can be influenced by information believability. Thus, we extend these two concepts with attributes to accommodate a *believability* check for read/produced

¹²Read time can be derived by analyzing the model, i.e., there is no specialized construct or attribute to represent it

information respectively (shown as [B] in Figure 3.7), i.e., produced/read information is believable from the perspective of its producer/ reader, if the produce/ read relations apply a believability check.

Information trustworthiness: can be subject to:

Trustworthiness of the source: as discussed in the trust section, we capture the trustworthiness of information source, by identify and model *threat(s)* that influence the trustworthiness of the produced information (e.g., producing fraud, falsified, or harmful information), and we identify and model actors' capabilities and motivations towards such threat.

Trustworthiness of the provision: can be captured based on the way information arrives to its final destination, taking into consideration its provision type (e.g., P/IP provision), and the operations (e.g., modify) that have been applied to it taking into consideration if such operations were authorized or not (e.g., permissions and trust)¹³.

Information accuracy¹⁴: can be analyzed based on:

Accuracy of produced information: can be analyzed based on its *believability*, which enables to avoid producing unintended information, and its *trustworthiness of the production process* if the producing goal has been delegated;

Accuracy of send information: can be analyzed based on the *trustworthiness of the provision*;

Accuracy of read information: can be analyzed based on *believability* and its *trustworthiness of the provenance*.

All modeling concepts that underlie our language are structured in terms of a meta-model shown in Figure 3.16, where we can identify: an actor covers two concepts (role and agent), an agent can *play* a role or more, and roles can be specialized from one another (*is_a*). Actors can be interdependent readers concerning an information item, they may have a set of goals, they aim to achieve, and they may have the related capabilities for their achievement.

Goals can be and/or-decomposed, and they contribute either positively or negatively to the achievement of one another. Goals may produce, read, modify and send information, where produce concept is enriched with a believability check, and read can be described by its type (e.g., optional or required), purpose of use, and believability check. While send can

¹³No special constructs are added to capture the trustworthiness of the provision

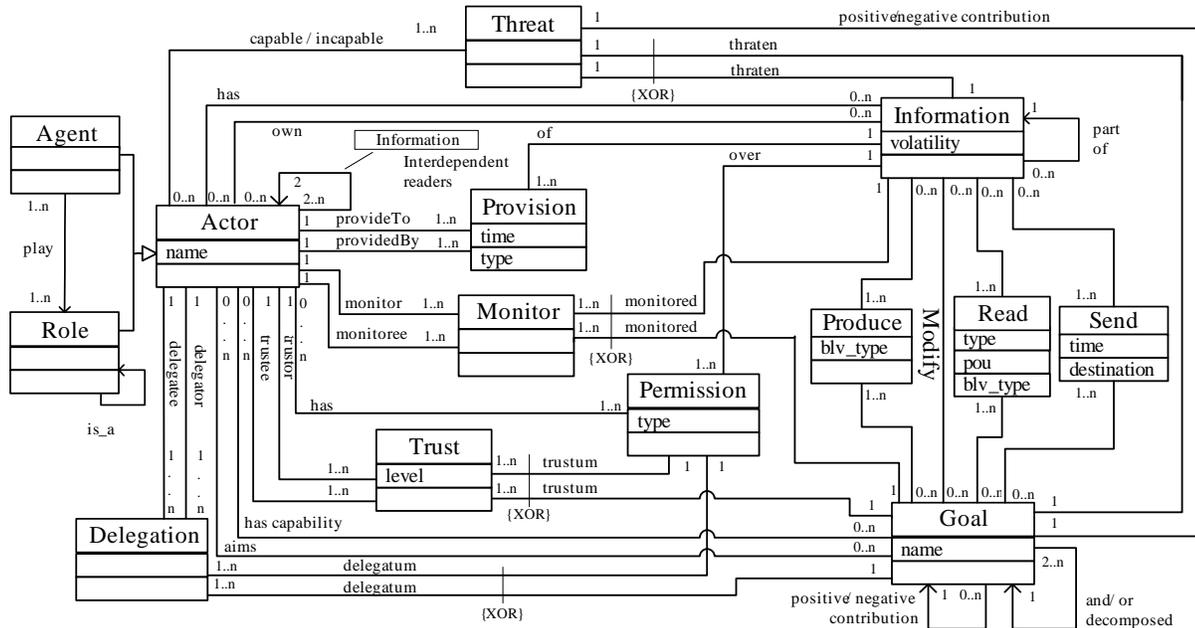


Figure 3.16: A meta-model of the main concepts of the modeling language

be described by a time and destination attributes. Information has volatility attribute that is used to analyze its validity, and it can be composed of several information items (*part of*). An actor can be a legal owner of information item, which gave it a full control over its use. Threat might threaten an information item or goal, and actors might be capable/incapable of achieving such threat. Moreover, threat may contribute either positively or negatively to the achievement of a goal or more. Actors may delegate goals/permissions to one another, and they may have information, and provide them to one another. Finally, actors may trust/distrusts one another for both goal and permissions delegation, and they can monitor one another for goal achievement and information production.

3.6 Workflow net with actors (WFA-net)

A Business Process (BP) can be defined as a set of activities that has a clear structure describing its sequencing order and dependencies [AS04]. Although information related problems can be a main reason of different kinds of errors in BPs [TvdAS09], the BP literature has focused on control-flow (activity flow) perspective of the process with less emphasis on information perspective. However, in recent years some efforts have been devoted to information-aware process design (e.g., Sadiq et al. [SOSF04]; Sidorova et al. [SST10]; Trcka et al. [TvdAS09]). Yet the focus of attention in these works is combining information flow with activity flow, i.e., they are able to detect when an activity in BP

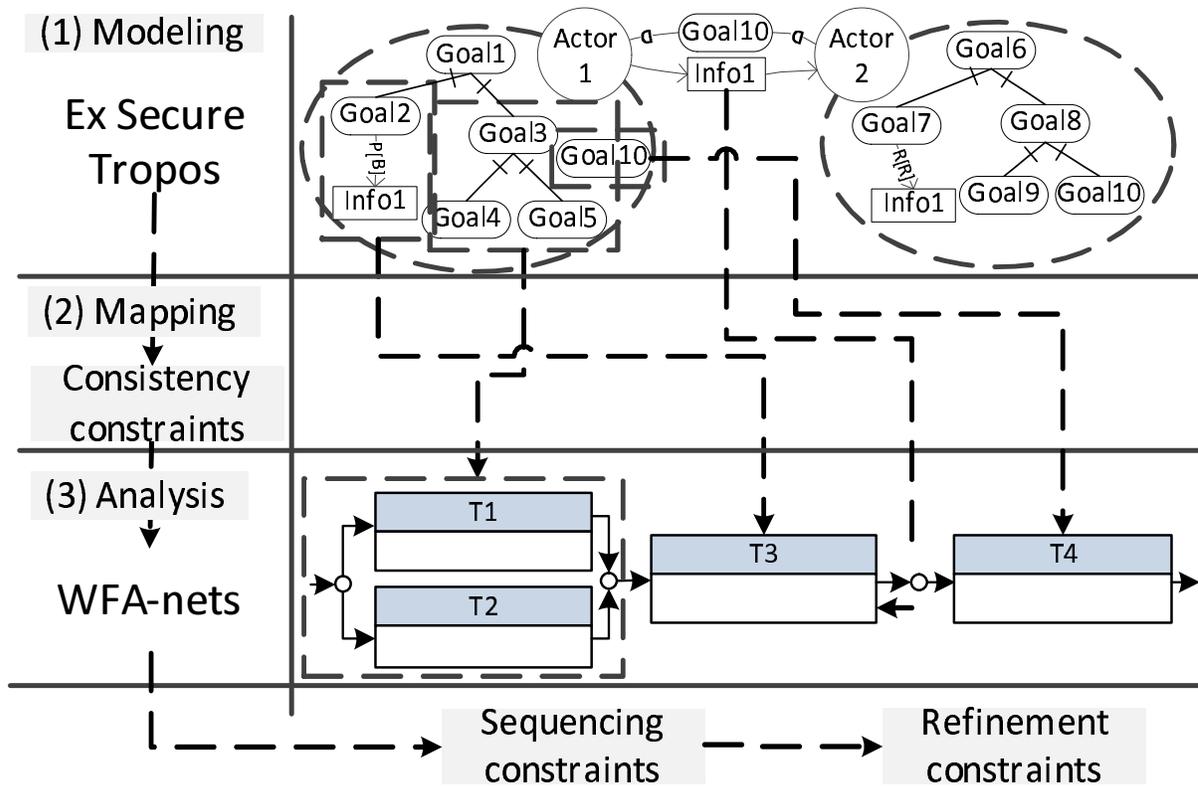


Figure 3.17: Overview of the approach for modeling and analyzing IQ in BP

relay on information that does not exist, but they say nothing about Information Quality (IQ) related concerns.

In the literature, we can find several techniques for dealing with IQ (e.g., preventing, detecting and correcting IQ related issues [Ham50]). Yet most of these techniques propose solutions that are able to address the technical aspects of IQ, and seem to be limited in addressing the social and organizational IQ related aspects. Such aspects are particularly important for BPs, since they are mainly executed by social actors and not only machines [YM94]. More specifically, most BPs occur in social context (e.g., socio-technical systems [ET60]), where humans and technical components are considered as an integral part of the BP. Thus, understanding the social and organizational context where the BPs are executed is essential to detect different kinds of vulnerabilities.

To this end, we propose a goal-oriented approach to capture IQ requirements of the social and organizational context where the BP is executed, and then introduce mechanisms for mapping these requirements into workflow net with actors (WFA-net) that is a modeling language, we propose, for capturing IQ requirements in BPs.

3.6.1 Approach for Modeling and Reasoning about IQ Requirements in Business Process

In this section, we propose our approach for modeling and analyzing IQ requirements in BP. An overview of the approach is shown in Figure 3.17, the process is composed of three main phases: (i) modeling phase; (ii) mapping phase; and (iii) analysis phase, each of them is discussed as follows:

Mapping phase

Aims to model the social, organizational and IQ requirements of the overall system where the BP occurs. In order to do that, we rely on an extended version of Secure Tropos [GG15d] that propose concepts for modeling IQ requirements. Figure 3.18 shows a portion of a goal model concerning the stock market system represented in the extended secure Tropos modeling language. For instance, `John` is an agent that plays `stock investor` role, which has a main goal of `G1. Make profit from trading securities` that is or-decomposed into `G1.1 Trade securities by itself` and `G1.2 Delegate trading activities to a trader`. Moreover, `G1.1.1 Trade securities by itself` is and-decomposed into `G1.1.1.1 Produce and send orders` and `G1.1.1.2 Finalize the trade`.

Moreover, the goal `G1.1.1.1 (P)roduces and (S)ends investor's orders`, and the goal `G1.1.1.2 (R)eads trade settlement`. Each of the previously mentioned relations between goals and information are required for the achievement of the goals, i.e., if a goal could not use (e.g., reads, sends, or produces) information as intend, it will not be achieved (it will be prevented). Moreover, the `stock investor` provides (Integrity Provision (IP)¹⁵) `investor's orders` to a `trader`, and it delegates the goal `G1.2 Delegate trading to a trader` to `stock trader`, and trusts it for its achievement. `Stock investor` is the owner of `investor's orders`, and it delegates (R)eads, (M)odifies and (S)ends permissions concerning it to the `trader`.

Mapping phase

In this section, we propose the workflow net with actors (WFA-net), and then we discuss several mechanisms that can be used for mapping IQ requirements model into WFA-net.

¹⁵IP provision preserves the integrity of the provided (transferred) information

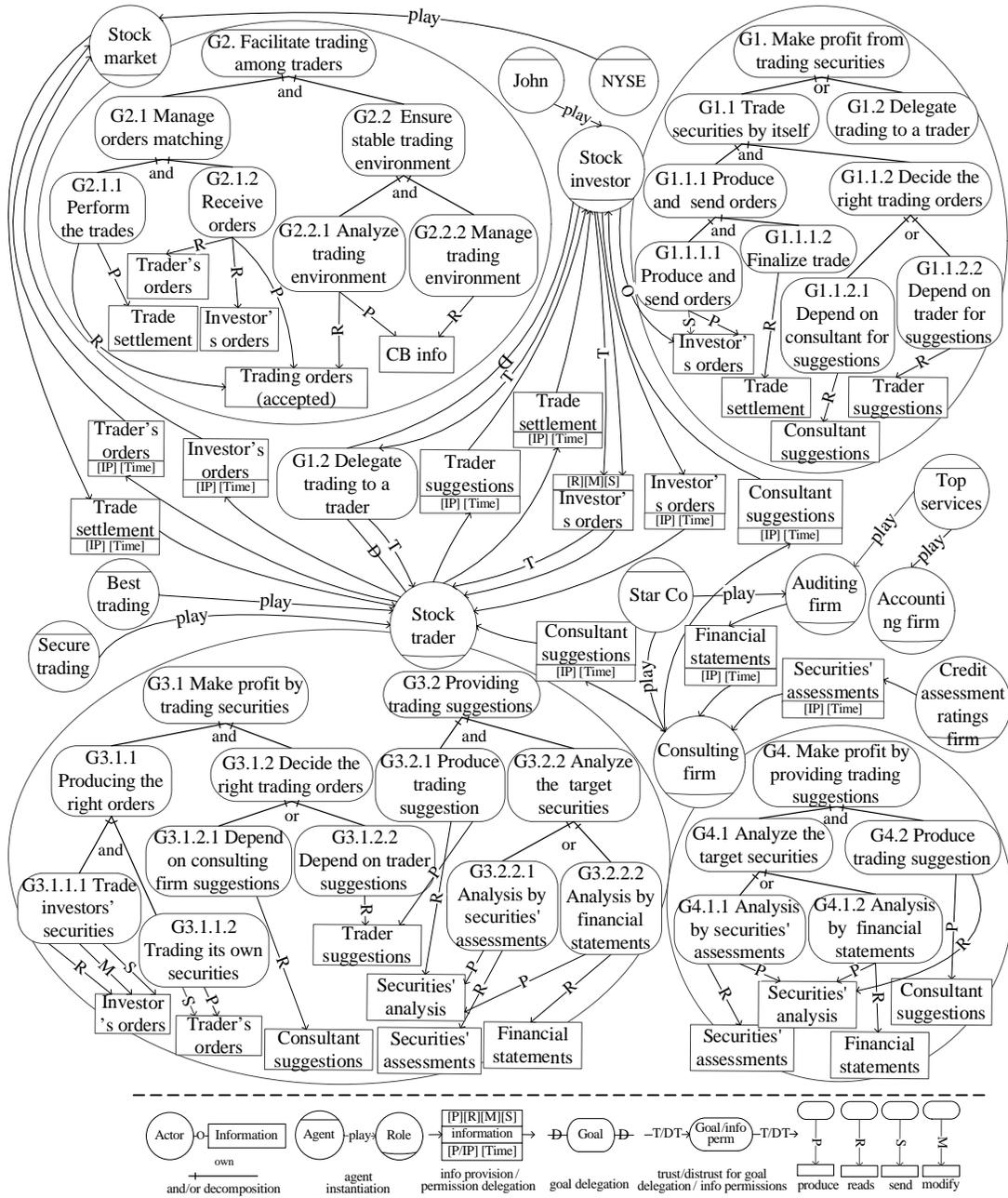


Figure 3.18: A partial goal model concerning the stock market structure

Work flow net with actors (WFA-net)

a workflow net with actors (WFA-net) adopts workflow net (WF-net) and extends it with the notion of social actor, and IQ related concerns. In WFA-net, activities (tasks) are assigned to social actors, and they may produce, read, modify, and send information items. In what follows, we define the semantics of WFA-nets. Let us consider a finite set

of social actors $A = \{a_1, a_2, \dots, a_n\}$, a finite set of information elements $I = \{i_1, i_2, \dots, i_m\}$, and a finite set of time intervals $T = \{t_1, t_2, \dots, t_m\}$, and we define $I_v \subseteq \{I \times T\}$ to describe information items along with their volatility values.

Moreover, to capture information send relations, we define $S \subseteq \{A \times I \times T\}$ that describe the target (actor) of send information, information to be send, and the required send time. Further, we define a set of responsibility predicates $\Pi_A = \{\pi_{a_1}, \pi_{a_2}, \dots, \pi_{a_k}\}$ to capture the relation between actors and activities they are responsible of (e.g., responsible(actor)); a set of produce predicates $\Pi_P = \{\pi_{p_1}, \pi_{p_2}, \dots, \pi_{p_j}\}$ to capture the relation between activities and information they produce (e.g., produce(info)); a set of read predicates $\Pi_R = \{\pi_{r_1}, \pi_{r_2}, \dots, \pi_{r_k}\}$ to capture the relation between activities and information they read (e.g., read(info)); a set of modify predicates $\Pi_M = \{\pi_{m_1}, \pi_{m_2}, \dots, \pi_{m_l}\}$ to capture the relation between activities and information they modify (e.g., modify(info)); a set of send predicates $\Pi_S = \{\pi_{s_1}, \pi_{s_2}, \dots, \pi_{s_l}\}$ to capture the relation between activities and information they send (e.g., send(actor, info, send_time)).

Furthermore, we define the following functions: $f_{\pi_a} = \Pi_A \rightarrow \{A\}$, responsibility function that assign responsibility predicates with actors responsible of achieving the related activities; function $f_{\pi_p} = \Pi_P \rightarrow 2^{I_v}$, production function that assigns produce predicates with information items that activities produce; function $f_{\pi_r} = \Pi_R \rightarrow 2^{I_v}$, reading function that assigns read predicates with information items that activities read; function $f_{\pi_m} = \Pi_M \rightarrow 2^{I_v}$, modify function that assigns modify predicates with information items that activities modify; and function $f_{\pi_s} = \Pi_S \rightarrow 2^S$, send function that assigns send predicates with information items that activities send.

Now, we define a WFA-net as a WF-net where every transition t is described with: an actor being assigned to perform the activity (res); a set of information items being produced (pd) by the activity when t fires; a set of information items being modified (md) by the activity when t fires; a set of information items being read (rd) by the activity when t fires; and a set of information items being send (sd) by the activity when t fires.

Definition 3.6.1 (WFA-net). *A workflow net with actors (WFA-net) $N = \langle P, T, F, res, pd, rd, md, sd \rangle$ consist of WF-net $N = \langle P, T, F \rangle$, an actor assigning function $res: T \rightarrow A$, information producing function $pd: T \rightarrow 2^{I_v}$, information reading function $rd: T \rightarrow 2^{I_v}$, information modifying function $md: T \rightarrow 2^{I_v}$, and information sending function $sd: T \rightarrow 2^S$.*

Example 26. *A WFA-net of a stock investor for trading securities is shown in Figure 3.19. Its actor set is $A = \{credit_firm, audit_firm, trader, investor, consulting_firm, stock_market\}$, and its information set is $I = \{securities_assessment,$*

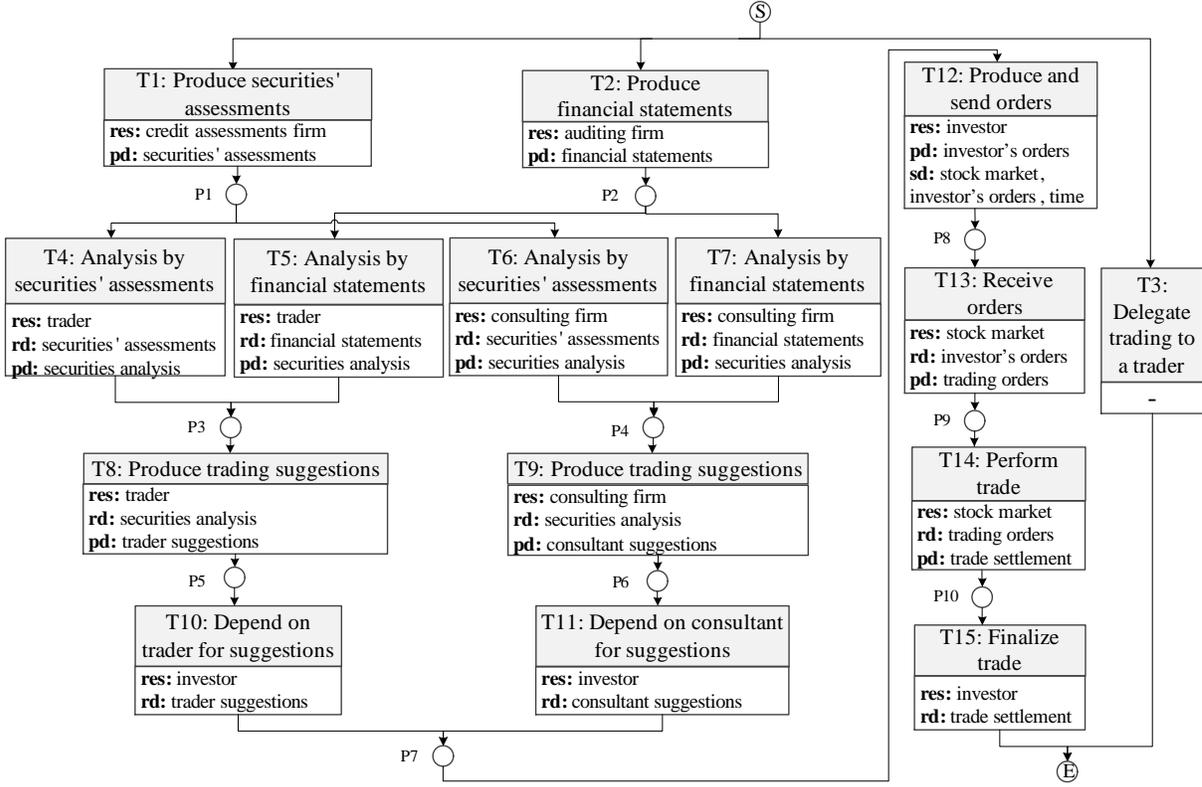


Figure 3.19: A WFA-net of a stock investor for trading securities

$financial_statement, trader_suggestion, trading_orders, consultant_suggestion, investors_orders, trading_settlement\}$. Considering the transition $T12$: *Produce and send orders*, the responsibility function $res(Produce\ and\ send\ orders) = \{investor\}$, the production function $pd(Produce\ and\ send\ orders) = \{investor's\ order\}$, the sending function $sd(Produce\ and\ send\ orders) = \{(stock\ market, investor's\ order, time)\}$, modify and read functions $md(Produce\ and\ send\ orders) = rd(Produce\ and\ send\ orders) = \{\emptyset\}$.

To capture the work flow in WFA-net, we should be able to evaluate the activities related predicates either to true (\top) or to false (\perp) based on an already defined criteria. Thus, we define the following functions, $\sigma_{\pi_a}: \Pi_A \rightarrow \{\top, \perp\}$ assigns to each responsibility predicate either \top , when the responsible actor can achieve the activity, or it assigns \perp , when the responsible actor cannot achieve the activity. Similarly, we define $\sigma_{\pi_p}: \Pi_P \rightarrow \{\top, \perp\}$, $\sigma_{\pi_r}: \Pi_R \rightarrow \{\top, \perp\}$, $\sigma_{\pi_m}: \Pi_M \rightarrow \{\top, \perp\}$, and $\sigma_{\pi_s}: \Pi_S \rightarrow \{\top, \perp\}$ that assign to each produce/ read/ modify/ send predicate either \top , when information item i can be produce/ read/ modify/ send by the activity, or it assigns \perp otherwise.

Finally, we define σ_{Π} function that sums the values of the previously mentioned func-

tions over their related predicates. $\sigma_{\Pi}: T \rightarrow (\top, \perp)$, where $\sigma_{\Pi} = \sigma_{\pi_a} \wedge \sigma_{\pi_p} \wedge \sigma_{\pi_r} \wedge \sigma_{\pi_m} \wedge \sigma_{\pi_s}$. Following WFD-net, we refer to a **state**¹⁶ of WFA-net as a configuration, where a WFA-net configuration is a state that includes responsible actors along with the produce, read, modify, and send information that the activity perform. Moreover, a **state** can be represented as σ_{Π} , and the set of all **states** is denoted by Σ .

Definition 3.6.2 (Configuration). *Let $N = \langle P, T, F, res, pd, rd, md, sd \rangle$ be a WFA-net, let m be a **marking** of N , and let σ_{Π} be as defined above. Then, $c = \langle m, \sigma_{\Pi} \rangle$ is a configuration of N . With Ξ we denote the set of all configurations, and the start configuration of N is defined by $\langle [start], \sigma_{\Pi}, (I_v = \emptyset) \rangle$. While $C_e = \{ \langle [end], \sigma_{\Pi} \rangle \mid \sigma_{\Pi} \in \Sigma \}$ defines the set of final configurations.*

In the initial configuration, only one place is marked [start], and I_v set is initialized to empty set. While a configuration is a final configuration, if it contains a marking [end]. A transition t of a WFA-net N can be enabled at a configuration $c = \langle m, \sigma_{\Pi} \rangle$, IFF: (1) the transition t is enabled at marking m (activity flow), and (2) the predicates related to configuration c enabling (σ_{Π}) must be true, which guarantees that IQ requirements are met. When a transition t is enabled, it may fires, where firing of transition t changes the marking as well as the related predicates and information set I_v , i.e., the firing of t enable a set of successor configurations $\langle m', \sigma'_{\Pi} \rangle$, and changes predicates and information.

Example 27. *consider transition T13: **Receive orders** in Figure 3.19, and suppose there is a token in place **p8**. The transition is enabled if stock market (responsible actor) has the capability to achieve such activity, information **investors_orders** fits for read from the perspective of the stock market, i.e., information has been already produced, stock market has it, and it does not suffer from any IQ related issue (e.g., information is valid (timeliness), accurate, stock market has the read permissions over it, etc.). Firing this transition means that the token in **p8** is removed, and a token is produced in **p9**. Moreover, information **trading_orders** is produced by the stock market. Note that we only consider information producing when a transition fires, since it affects all the other information related operations (e.g., sends, modifies). While other IQ aspects can be captured with the help of the automated reasoning support (discussed in section 4.3).*

Mapping IQ Requirements from the Goal model into WFA-nets

in this section, we describe how IQ requirements model can be mapped into WFA-net. In particular, we define rules for identifying complete building blocks that are used to represent the extended secure Tropos constructs, which can be mapped into WFA-net

¹⁶A state (also called marking) of a Petri net is a distribution of tokens over its places

activities. Moreover, we define several sets of constraints that should be followed during the mapping process to ensure the correctness of both the mapping and the resulting WFA-net:

Building blocks: we define 3 rules for identifying complete building blocks: (a) a goal that is not and/ or-decomposed of any other goal, and it is not composed into sub-goals as well, can be considered as a complete building block, and it can be mapped into a WFA-net activity (task) taking into consideration the actor, who is responsible of its achievement, and information it relies on (if any); (b) goals that are and-decomposed from a parent goal are considered as a complete building block, and they can be mapped into a sequence of WFA-net activities, where each of these activities represents a sub-goal; (c) goals that are or-decomposed from a parent goal are considered as a complete building block, and they can be mapped into parallel (alternatives) WFA-net activities, where each of these activities represents a sub-goal.

Consistency constraints: we define 3 consistency constraints that can be used to ensure a correct mapping between the identified building blocks and WFA-net activities: (i) mapping is allowed for complete building blocks only, i.e., no goal is allowed to be mapped unless it can be considered as a complete building block; (ii) if the WFA-net is used to model a plan to achieve a top-level goal, the full plan to achieve the top-level goal should be considered in the WFA-net; and (iii) no information is allowed in the WFA-net unless its source (the goal that produces it) exists in the WFA-net.

Sequencing constraints: we define 3 sequencing constraints that can be used to ensure the proper ordering of the WFA-net activities: (i) activities in WFA-net should be consistent with their sequencing order in their own building blocks; (ii) if an activity depend on another activity, it should appear after the activity it depends on in the WFA-net; (iii) if an activity depend on the outcome of another activity (e.g., information), it is desirable to appear after the activity it depends on its outcome.

Refinement constraints are constraints derived from the WFA-net semantics, and used to refine the WFA-net that results from the sequencing phase. We define two simple *refinement constraints*: (i) no two places (p_1, p_2) can appear in sequence without a transition (t) separating them; and (ii) no two transitions (t_1, t_2) can appear in sequence without a place (p) separating them.

Example 28. *in this example, we show how investor process for trading securities shown in Figure 3.18 can be mapped into WFA-net shown in Figure 3.19. The investor aims to achieve the top-level goal $G1$, but $G1$ cannot be considered as a building block, since it is or-decomposed into $G1.1$ and $G1.2$. Thus, instead of $G1$ we have $G1.1$ and $G1.2$ that can be mapped as parallel activities into WFA-net. $G1.2$ can be mapped into $T3$ activity, while $G1.1$ is and-decomposed into $G1.1.1$ and $G1.1.2$, which can be mapped into*

two sequential transactions. Moreover, *G1.1.1* is also and-decomposed into *G1.1.1.1* and *G1.1.1.2*, which can be mapped into two sequential transactions *T12* and *T15* respectively. While *G1.1.2* is or-decomposed into *G1.1.2.1* and *G1.1.2.2*, and they can be mapped into two parallel transactions *T11* and *T10* respectively. Furthermore, transaction *T10* needs to read trader suggestion information, and transaction *T11* needs to read consultant suggestion information. Since no information is allowed to exist without its source, we add *T8* and *T9* transactions that produce trader suggestion and consultant suggestion respectively. However, *T8* requires to read securities analysis that is produced by trader, which can be produced either by *T4* or *T5*. Similarly, *T9* requires to read securities analysis that is produced by consultant, which can be produced either by *T6* or *T7*. Moreover, *T4* and *T6* need to read securities assessment. Thus, *T1* is added since it is responsible of producing such information. Similarly, *T5* and *T7* need to read financial statement information. Thus, *T2* is added since it is responsible of producing such information. At the other hand, *T15* needs to read trade settlement information, thus, *T13-14* are also added. Finally, following the refinement constraints, we add some position between transactions when required.

In this section, we describe the automated reasoning support that our approach proposes to guarantee the correctness and consistency of information-flow, IQ requirements, and control-flow of BPs. In order to verify the correctness and consistency of BP model, we provide a Datalog [AHV95] formalization of all the concepts that have been introduced in the paper, along with the reasoning axioms¹⁷. Moreover, we define a set of properties of the design (shown in Table 4.8) that can be used to verify the correctness and consistency of the BP model; in what follows we discuss each of these properties:

3.6.2 Analysis Phase¹⁸

aims to verify the correctness and consistency of the BP model. In particular, we define a set of properties to check the correctness and consistency of the BP control-flow; information flow along with IQ requirements, i.e., BP is correct and consistent, if all of these properties hold.

3.7 Dealing with IQ requirements

A system usefulness is determined by both its functional and its non-functional (quality) requirements [CdPL09], where functional requirements are used to represent services that

¹⁷The formalization of the concepts and axioms is omitted due to space limitation, yet they can be found at <https://mohamadgharib.wordpress.com/iqbp/>

¹⁸For more information about the analysis phase refer to [GG15e]

the system is expected to deliver, and nonfunctional requirements refer to quality requirements that the system needs to satisfy while delivering the services [JFS06]. Thus, an integrated analysis of both functional and quality requirements is essential to determine the system usefulness, since as highlighted in some functional requirements might not be useful without their necessarily related non-functional requirements (e.g., IQ) [CdPL09].

However, non-functional requirements are more difficult to be expressed in a measurable way [NE00], i.e., they do not always have a clear-cut criteria for their satisfaction [MCN92; CdPL09]. Thus, we advocate that an appropriate integration of such requirements is not possible without removing any ambiguity related to the verification of quality requirements.

Like all non-functional (quality) requirements, IQ requirements use to be represented as generic qualitative properties of the system, without specific methods for their analysis [CdPL09]. To tackle this problem, we propose a systematic process for dealing with IQ requirements. In particular, we introduce a set of modeling constructs that enable for capturing the stakeholders' top-level IQ requirements as softgoals (usually informal), and then we discuss how such softgoals can be gradually refined until reaching their leaf softgoals (cannot be refined more than a leaf softgoal). Finally, we show how leaf softgoals can be approximated into Quality Constraint (QC) that can provide a means end (formal specifications) for achieving the leaf softgoal. The process along with an example about its modeling constructs is depicted in 3.20. Each of the previously mentioned concepts is defined as follows:

Top-level IQ softgoals: are softgoal concerning IQ requirements, and they are used as a starting point for identifying the stakeholders' needs concerning IQ. However, at this point IQ softgoals are likely to be informal and imprecise, but they became more precise during the latter refinement activities.

And-decomposition for IQ softgoals refinement: a softgoal can be refined into more specific sub softgoals, if the joint satisfaction of these softgoals is considered equivalent to the satisfaction of the refined softgoal [JMF08]. Usually, softgoals refinement into more specific sub softgoals can be performed based on some related taxonomy. For instance, Mylopoulos et al. [MCN92] propose taxonomy for refining accuracy and performance softgoals. While Antón et al. [AER02] introduce taxonomy for refining privacy softgoals. The same can be applied to IQ softgoals, i.e., they can be refined based on different IQ dimensions, we rely on the IQ taxonomy (IQ model) shown in 3.15.

In particular, we introduce *and-decomposition* relation between an IQ softgoal and its sub IQ softgoals instead of *contribution links* proposed in [MCN92], since such requirements will reach a point that enables for clearly deciding whether they can be achieved or not (operational specifications). Note that IQ softgoals cannot be refined more that the

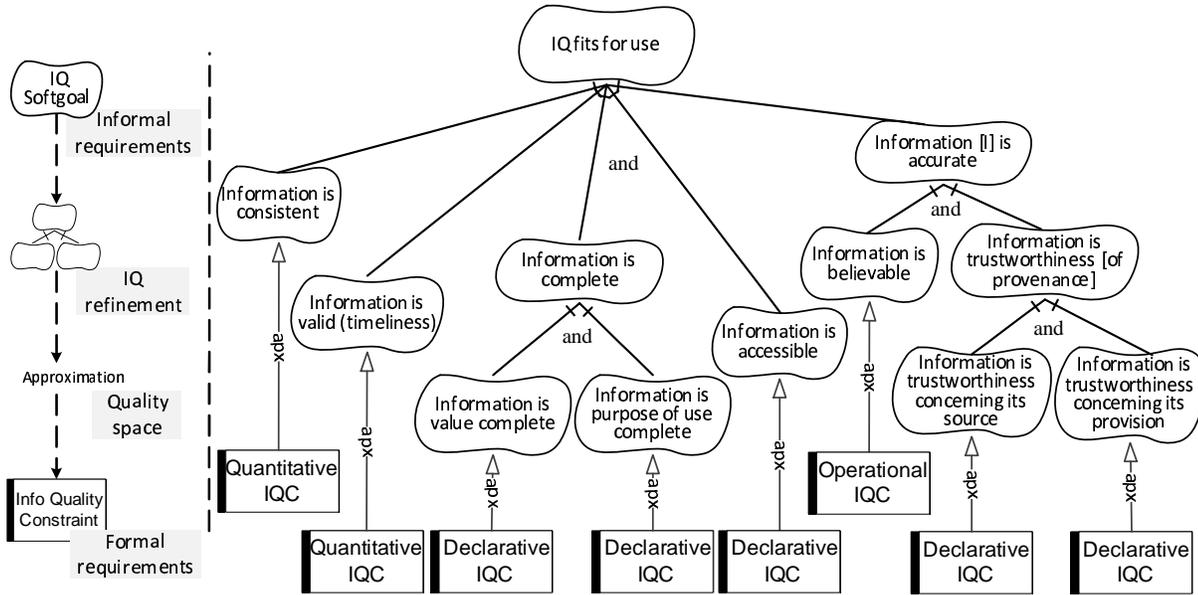


Figure 3.20: Process: from informal to formal IQ requirements

leaf IQ softgoals, which are used to represent leaf IQ dimensions (shown in Figure 3.15), where a leaf IQ dimension is an IQ dimension that does not have sub dimensions (e.g., leaf IQ believability softgoal is used to represent believability (leaf IQ dimension)).

Approximating leaf IQ Softgoals: as previously mentioned, softgoals are difficult to be expressed in a measurable way. Yet Jureta et al. [JMF08] introduce the *approximation* relation through which a softgoal can be satisfied by a Quality Constraint (QC), i.e., a QC can provide clear-cut criteria for the satisfaction of a softgoal. However, leaf IQ softgoals are used to capture different IQ dimensions (e.g., accuracy, completeness, etc.), i.e., each of them is used to describe different aspects of IQ. Thus, leaf IQ softgoals might not have the same nature/type, and in turn, they may need to be approximated in different ways. To tackle this problem, we rely on Glinz [Gli05] work¹⁹, to get better understanding of the nature/type of the leaf IQ softgoals, and to define the appropriate Information Quality Constraints (IQC)²⁰ for their approximation. Moreover, for the approximation to be consistent with the different types of leaf IQ softgoals, we define three different types of IQCs:

1- Operational IQC: are constraints that define the required actions to be performed in already determined situations.

Example 29. *IQ softgoal concerning information believability can be approximated*

¹⁹Glinz classify requirements based on their kind, satisfaction and representation

²⁰We use IQC to refer to QC, since no other type of constraints is used in this paper

Table 3.2: IQ softgoal classification & approximation into IQC

Leaf IQ softgoals	Kind	Satisfaction	Representation	Approximated into IQC
Believability	Functional	Hard	Operational	Operational IQC
Trustworthiness	Constraint	Hard	Declarative	Declarative IQC
Completeness	Constraint	Hard	Declarative	Declarative IQC
Timeliness	Performance	Hard	Quantitative	Quantitative IQC
Consistency	Performance	Hard	Quantitative	Quantitative IQC

into operational IQC that define a mine and max values to determine the believability of produced/ read information.

2- Declarative IQC: are constraints used to define properties of the system that should hold.

Example 30. *IQ softgoal concerning the trustworthiness of provision can be approximated into declarative IQC stated that information should be transferred only through IP provision.*

3- Quantitative IQC: are constraints used to specify properties of the system that should hold, and can be measured on an ordinal scale.

Example 31. *IQ softgoal concerning consistency can be approximated into quantitative IQC stated that interdependent readers should rely on information that has the same currency (age).*

Table 3.2 shows how leaf IQ softgoals can be classified, and how they can be approximated into the adequate IQCs. Finally, in order for the approximation relation between IQ softgoal and its related IQC to hold, a well-defined quality space should exist [JMF08], where a quality space can be defined as a certain conceptual space that can be used to describe the quality value [MBG⁺03]. The main purpose of the quality space is providing a general consensus among the stakeholders of the system on how quality aspects (e.g., IQ dimensions) can be measured, which removes any ambiguity related to the verification of IQCs, i.e., determining whether a certain IQC is satisfied or not. For instance, both information timeliness and consistency are time related aspects. Thus, how time can be represented and measured should be clear to all stakeholders of the system, i.e., the allowed number of digits along with the value they represent (e.g., seconds, milliseconds, etc.).

3.8 Chapter summary

In this chapter, we proposed our modeling language, we started by discussing the objectives of the language 3.1, and then we introduced the principles that have been taken into account while developing the language 3.2. In Section 3.3, we present our modeling language 3.3, that can be used for modeling the actors of the system along with their objectives, entitlements, and authorities. In addition, it propose more refined concepts for modeling trust among actors, and propose constructs for modeling IQ requirements. In 3.6, we discussed workflow net with actors (WFA-net) that is a modeling extension, we proposed, for modeling and analyzing IQ requirements in business processes. Finally in 3.7, we presented a systematic process for dealing with IQ requirements.

Chapter 4

Automated Reasoning Support

If you torture the data long enough, it will confess.

Ronald Coase

A modeling language allows for drawing graphical requirements models that can be used to represent requirements of the system-to-be. However, we cannot rely on such graphical models to perform any kind of automated analysis without a formal representation of their semantics. In this chapter, we present the formal framework that underlies our modeling language. The rest of this chapter is organized as follows, Section 4.1 describes how our modeling constructs are mapped into the formal framework, i.e., we define several predicates to represent the different modeling constructs along with their relations. While in Section 4.2, we discuss the reasoning rules (axioms), which can be used to derive new knowledge from the predicates that have been derived from the graphical model. Finally, in Section 4.3, we verify the requirements model by proposing the properties of the design, which can be used to verify the correctness and consistency of the requirements model.

4.1 Concepts and Relations

As previously mentioned, graphical models cannot support any kind of automated analysis without a formal representation of their semantics. Thus, we need to translate graphical models into formal specifications that enables for performing the required analysis to verify the correctness and consistency of the requirements model. To this end, we rely on Disjunctive Datalog [BFI⁺09] to define the formal semantics underlying our modeling language, since Disjunctive Datalog is more expressive than most other logic languages with negation [EGM97].

In addition, we need an inference engine that allows for performing the required analysis, i.e., deducing new knowledge from facts derived from the graphical requirements model. Among existing inference engines (e.g., Cmodels [Lie05], Smodel [NSS00], DLV system [LPF⁺06], etc.), we chose DLV system¹, which is a powerful deductive database system that allows for deducing new knowledge based on rules and facts stored in deductive database, and it has been reported in [Zan06] that DLV system is more efficient than the other available engines.

In order to be consistent with deductive database terminology [BFI⁺09], we distinguish between two types of predicates: (1) extensional predicates (extensional database (EDB)), which can be defined as a collection of facts that are used to represent the graphical model constructs along with their semantic relations; and (2) intensional predicates (intensional database (IDB)), which are used to deduce new facts depending on the extensional predicates and the reasoning rules (axioms). In what follows, we propose the extensional predicates in 4.1.1, and the intensional predicates in 4.1.2.

4.1.1 Extensional predicates

Extensional predicates (shown in Table 4.1) are used to represent the graphical model constructs along with their different relations. We describe each one of them as follows:

Type predicates: unary and binary predicates that are used for identifying actors, roles, agents, goals, information, and threat entities respectively. Type predicates are necessary for some rules, since they differentiate between modeling constructs based on their types.

Role and agent relations: binary predicates that are used for identifying specialization and instantiation relations among actors respectively.

- **instantiation** (*plays*) has two arguments, where the first is an agent (*a*) and the last is a role (*r*), and it is true if (*a*) plays (*r*).
- **specialization** (*is_a*) has two arguments that represent two roles, and it is true, if *r*₁ is specialization of *r*₂.

Actors' properties: binary predicates that are used to represent the different relations between an actors, goals and information.

- **aims:** the first argument of aims is an actor (*a*), and the second is a goal (*g*), where *aims(a, g)* is true if *g* is a top level goal of *a*.

¹<http://www.dlvsystem.com/>

Table 4.1: Extensional Predicates

Type predicates
actor(Actor: a)
role(Role: r)
agent(Agent: ag)
goal(Goal: g)
info(Info: i, Volatility: v)
threat(Threat:t)
Role and agent relations
isa(Role: r ₁ , Role: r ₂)
plays(Agent: x, Role: r)
Actors' properties
own(Actor: a, Info: i)
aims(Actor: a, Goal: g)
Goal analysis
andDecomposition(Goal: g, Goal: g ₁)
orDecomposition(Goal: g, Goal: g ₁)
contribute(Type:t, Goal :g, Goal:g ₁)
Goal-information relations
produce(BType: t, Goal :g, Info: i)
read(Type: t, POU: pou, BType: t, Goal :g, Info: i)
modify(Goal: g, Info: i)
send(Time: t, Goal: g, Actor: a, Info: i)
partOf(I, I1)
Threat analysis
threaten(Threat: t, Goal/info: g/i)
capable_threat(Actor: a, Threat: t)
incapable_threat(Actor: a, Threat: t)
threat_contribute(Type: t, Threat: t, Goal/info: g/i)
Social relations
provide(Type: tp, Time: tm, Actor: a, Actor: a, Info: i)
delegate(Actor: a, Actor: b, Goal :g)
delegate_perm(Actor:a, Actor:b, Perm:r, Perm:p, Perm:m, Perm:s, Info:i)
monitor(Actor: a, Actor: b, Goal/info: g/i)
trustReq(Type:t, Actor: a, Actor: a, Goal :g)
trust_perm(Actor:a, Actor:b, trust:t, trust:t, trust:t, trust:t, Info:i)

- **own**: the first argument of own is an actor (a), and the second is information (i), where $\text{own}(a, i)$ is true, if a is the legal owner of i .

Goal analysis: binary and ternary predicates that are used to capture the different re-

lations between goals.

- **andDecomposition**: the two arguments of andDecomposition are goals $(g, g1)$, and andDecomposition($g, g1$) is true, if $g1$ is and refinement of g .
- **orDecomposition**: the two arguments of orDecomposition are goals $(g, g1)$, and orDecomposition($g, g1$) is true, if $g1$ is or refinement of g .
- **contribute**: the first argument of contribute is a type (positive/negative), the second and the third arguments are goals $(g, g1)$. contribute(positive/negative, $g, g1$) is true, if g contributes positive/negative to $g1$.

Goal-information relations: predicates that are used to describe the different relations between goals and information.

- **produces**: the first argument of produces is believability check type, the second is a goal (g) , and the third is information (i) , where produces(b, g, i) is true, if g produces i , with believability check type b .
- **read**: the first argument of read is read type t (optional/ required), the second is purpose of use pou , the third is believability check type b , the fourth is goal (g) , while the last is information (i) . read(t, pou, b, g, i) is true, if g optional/ required reads i for purpose of use pou and believability check type b .
- **modify**: the first argument of modify is a goal (g) , and the second is information (i) , where modify(g, i) is true, if g modifies i .
- **send**: the first argument of send is period of time (t) , the second is a goal (g) , and the third is an actor a , while the last is information (i) , where send(t, g, a, i) is true, if g sends i to actor a within period of time t .
- **partOf** the two arguments of partOf are information items i and $i1$, where partOf($i, i1$) is true, if $i1$ is part of i .

Threat analysis: predicates that are used to describe threats along with their different relations.

- **threaten**: the first argument of threaten is a threat t , and the second is information or goal i/g , where threaten($t, i/g$) is true, if t threaten i/g .
- **capable_threat**: the first argument of capable_threat is an actor a , and the second is a threat t , where capable_threat(a, t) is true, if a is capable of achieving threat t .

- **incapable_threat**: the first argument of `incapable_threat` is an actor a , and the second is a threat t , where `incapable_threat(a, t)` is true, if a is incapable of achieving threat t .
- **threat_contribute**: the first argument is a type (positive/negative), the second argument is a threat t , and the third is a goal g . `threat_contribute(positive/negative, t, g)` is true, if goal g contributes positive/negative to threat t .

Social relations: predicates that are used for capturing actor' social interactions.

- **provide**: the first argument of `provide` is provision type tp , the second is provision time tm , while the third and fourth arguments are actors a and b , and the last is information i , where `provide(tp, tm, a, b, i)` is true if a provides b with information i through provision type tp , and within provision time tm .
- **delegate**: the first two arguments of `delegate` are actors a and b , and the third is a goal g , where `delegate(a, b, g)` is true if a delegates a goal g to b .
- **delegate_perm**: the first two arguments of `delegate_perm` are actors a and b , and arguments from number three until six are permissions for produce, read, modify, and send respectively, while the last is information i . We say `delegate_perm(a, b, p,r,m,s,i)` is true if a delegates permission $p/r/m/s$ over information i to b .
- **monitor**: the first two arguments of `monitor` are actors a and b , and the last is information i or goal g . We say that `monitor($a, b, i/g$)` is true, if actor a monitors actor b during the achievement of goal g / producing information i .
- **trustReq**: the first argument of `trustReq` is a trust type (trust/ distrust), the second and third arguments are actors a and b , while the last is a goal (g). We say that `trustReq(t, a, b, g)` is true if the trust level between a b for the achievement of goal g is t .
- **trust_perm**: the first two arguments are actors a and b , and arguments from number three until six are trust/distrust over produce, read, modify, and send permissions respectively, while the last argument is information i . `trust_perm(a, b, p, r, m, s, i)` is true if a trust/distrusts b concerning each of p, r, m, s permissions over information i .

4.1.2 Intensional predicates

Intensional predicates are used to deduce new facts (knowledge) depending on the extensional predicates and the reasoning rules (axioms). Some of the intensional predicates are

Table 4.2: Intensional Predicates

Actors analysis	
objective(A, G)	is_responsible(C, G)
can_achieve(B, G)	capable_achieve(R, G)
producer(A, I, T)	reader(T, POU, BT, A, I)
sender(T, A, B, I)	modifier(A, I)
has(A, I, T)	fits_reader(A, I)
need_perm(P, A, I)	has_perm(P, A, I)
achieved(A, G)	achieve(A, G)
motivation(T, A, G)	
Goals analysis	
fits_send(T, G, B, I)	not_leaf(G)
dependent(G)	prevented(G)
Information Quality analysis	
accessible_read(A, I)	accurate_read(A, I)
complete_read(A, I)	valid_read(A, I)
consistent_read(A, I)	only_reader(A, I)
interdependent_readers(A, B, I)	inconsistent_reader(A, I)
Social analysis	
prvChain(T, Z, A, B, I)	deleChain(A, B, G)
dele_permChain(A,B,P,R,M,S,I)	monitorChain(A, B, M)
Trust analysis	
trust_produce(T,A,B,I)	trust_permChain(A,B,TP,TR,TM,TS,I)
trust_threat(A, B, G)	threat_source(A, I/G)
threat_motivation(T, A, Th)	

shown in Table 4.2, we discuss them as follows:

Actors analysis: predicates that are used for capturing actors' objectives, entitlements and authorities.

- **objective:** the first argument of objective is an actor a , and the second is a goal g , where $\text{objective}(a, g)$ is true, if g is an objective of a .
- **is_responsible:** the first argument of is_responsible is an actor a , and the second is a goal g , where $\text{is_responsible}(a, g)$ is true, if a took the responsibility

of g achievement.

- **can_achieve**: the first argument of `can_achieve` is an actor a , and the second is a goal g , where `can_achieve(a, g)` is true, if a has the capability (directly or indirectly) to achieve g .
- **capable_achieve**: the first argument of `capable_achieve` is an actor a , and the second is a goal g , where `capable_achieve(a, g)` is true, if a has the self-capability to achieve g .
- **producer**: the first argument of `producer` is an actor a , and the second is an information item i , while the last is time (currency (age) of information). `Producer(a, i, t)` is true, if a is the producer of i .
- **reader**: the first argument is read type t , and the second is purpose of use pou , the third is the believability check type bt , while the fourth is an actor a , and the last is information i . `reader(t, pou, bt, a, i)` is true, if a reads i with read type t , believability check type bt , and for purpose of use pou .
- **sender**: the first argument of `sender` is the required time of send t , the second and third are actors a and b (sender and the receiver respectively), while the last is information i to be send. `sender(t, a, b, i)` is true, if a needs to send i to a within time period t .
- **modifier**: the first argument of `modifier` is an actor a , and the second is information i , where `modifier(a, i)` is true if a needs to modify i .
- **has**: the first argument of `has` is an actor a , and the second is an information item i , while the last is *read time* the currency (age) of information that an actor has. `has(a, i, t)` is true, if a has i with currency t .
- **fits_reader**: the first argument of `fits_read` is an actor a , while the last is an information item i . `fits_read(a, i)` is true if i fit for read from the perspective of a (reader).
- **need_perm** the first argument of `need_perm` is the permission type $p/r/m/s$, the second is an actor a , while the last is an information item i . `need_perm(p, a, i)` is true, if actor a needs permission $p/r/m/s$ over information i .
- **has_perm** the first argument of `has_perm` is the permission type $p/r/m/s$, the second is an actor a , while the last is an information item i . `has_perm(p, a, i)` is true, if actor a has permission $p/r/m/s$ over information i .
- **achieved**: the first argument of `achieved` is an actor a , and the second is a goal (g), where `achieved(a, g)` is true, if g is achieved (directly or indirectly) from the perspective of actor a .

- **achieve**: the first argument of `achieve` is an actor a , and the second is a goal g , where `achieve(a, g)` is true, if g is achieved directly by actor a .
- **motivation**: the first argument of `motivation` is a type t (positive/ negative), the second argument is an actor a , while the third is a goal g . `motivation(t, a, g)` is true if a is t motivated to achieve g .

Goals analysis: predicates that are used for representing goals' relations.

- **fits_send**: the first argument of `fits_send` is period of time t , the second is a goal g , and the third is an actor a , while the last is an information item i , where `fits_send(t, g, a, i)` is true, if g sends i to actor a within period of time t .
- **not_leaf**: the only argument of `not_leaf` is a goal g , where `not_leaf(g)` is true, if g is not a leaf goal (g is and/or decomposed into finer goals).
- **dependent**: the only argument of `dependent` is a goal g , where `dependent(g)` is true, if g is information dependent (e.g., `produce_dependent`, `read_dependent`, `modify_dependent`, `send_dependent`).
- **prevented**: the only argument of `prevented` is a goal g , where `prevented(g)` is true, if g was prevented for being achieved (e.g., `produce_prevented`, `read_prevented`, `modify_prevented`, `send_prevented`).

Information Quality analysis: predicates that are used for analyzing IQ.

- **accessible_read** the first argument of `accessible_read` is an actor a , while the last is an information item i . `accessible_read(a, i)` is true if a has the permission to read i .
- **accurate_read** the first argument of `accurate_read` is an actor a , while the last is an information item i . `accurate_read(a, i)` is true if i is accurate from the perspective of a (reader).
- **complete_read** the first argument of `complete_read` is an actor a , while the last is an information item i . `complete_read(a, i)` is true if i is complete from the perspective of a (reader).
- **valid_read** the first argument of `valid_read` is an actor a , while the last is an information item i . `valid_read(a, i)` is true if i is valid from the perspective of a (reader).
- **consistent_read** the first argument of `consistent_read` is an actor a , while the last is an information item i . `consistent_read(a, i)` is true if i is consistent from the perspective of a (reader).

- **only_reader** the first argument of `only_reader` is an actor a , while the last is an information item i . `only_reader(a, i)` is true if there is no other reader of i but a .
- **interdependent_readers** the first and second arguments of `interdependent_readers` are actors a and b , while the last an information item i . `interdependent_readers(a, b, i)` is true if a and b reads i for the same purpose of use.
- **inconsistent_reader** the first argument is an actor a , and the last is an information item i . `inconsistent_reader(a, i)` is true if there is at least one other actor (e.g., b) that reads i for the same purpose of use, and i value is inconsistent between a and b .

Social analysis: predicates that are used for analyzing social interactions among actors.

- **prvChain:** the first argument of `prvChain` is provision type tp , the second is provision time tm , while the third and fourth arguments are actors a and b , and the last is information i , where `prvChain(tp, tm, a, b, i)` is true if there is a provision chain between a and b concerning information i through provision type tp , and within provision time tm .
- **deleChain:** the first two arguments of `deleChain` are actors a and b , and the third is a goal g , where `deleChain(a, b, g)` is true if there is a delegation chain between a and b concerning goal g .
- **dele_permChain:** the first two arguments of `dele_permChain` are actors a and b , and arguments from number three until six are permissions for produce, reads, modify, and send respectively, while the last is an information item i . We say that `dele_perm_Chain(a, b, p,r,m,s,i)` is true if there is a permission delegate chain from a to b concerning $p/r/m/s$ permission over information i .
- **monitorChain:** the first two arguments of `monitorChain` are actors a and b , and the last is information i or goal g . We say that `monitorChain($a, b, i/g$)` is true, if there is a monitoring chain between a and b concerning the achievement of goal g / producing information i .

Trust analysis: predicates that are used for analyzing trust requirements.

- **trust_produce** the first argument of `trust_produce` is a type (trust/distrust), the second and third arguments are actors a and b , while the fourth is an information item i . We say that `trust_produce(t, a, b, i)` is true, if the trust type between a and b for producing i is t .

- **trust_permChain** the first two arguments are actors a and b , and arguments from number three until six are trust/distrust over produce, read, modify, and send permissions respectively, while the last argument is information i . $\text{trust_permChain}(a, b, p, r, m, s, i)$ is true, if there is a trust chain (trust/distrusts) between a and b concerning each of p, r, m, s permissions over information i .
- **trust_threat** the first two arguments of trust_threat are actors a and b , the third argument is a goal g . We say that $\text{trust_threat}(a, b, g)$ is true, if a belief that b represent a threat to trust relation between them concerning the achievement of goal g .
- **threat_source** the first argument of threat_source is an actor a , the second argument is a goal g or an information item i , where $\text{threat_source}(a, i/g)$ is true, if a is a threat source to i/g , i.e., a has the capability and positively motivated to achieve the trust related threat of i/g .
- **threat_motivation** the first argument of threat_motivation is a type (positive/ negative), the second argument is an actor a , while the third is threat Th . We say that $\text{threat_motivation}(t, a, th)$ is true, if a is t motivated to achieve threat th .

4.2 Reasoning rules (axioms)

In this section, we describe the reasoning rules (axioms) that are used to in derive new knowledge depending on extensional and intensional predicates, and helps in performing the required analysis to verify the requirements model.

4.2.1 Actors' objectives, entitlements and capabilities axioms

Table 4.3, lists the actors' objectives, entitlements and capabilities axioms.

Actors' objectives axioms: O1-4 axioms are used to identify actors' objectives. O1 states that if an actor aims for a goal, the goal became an objective for the actor, i.e., the actor aims to achieve such goal. O2 states that if a goal is delegated to an actor, and the actor can take the responsibility of its achievement, it became an objective of such actor. O3-4 state that if a goal belongs to actor objectives and such goal is and/ or decomposed, all of its sub-goals became objectives to the actor.

Actors' entitlements axioms: E1 states that an actor became responsible of a goal achievement, if the goal is an objective of the actor, and the actor has the capabilities to achieve it.

Table 4.3: Actors' objectives, entitlements and capabilities axioms

Actor's objectives	
O1	<code>objective(A, G) :- aims(A, G).</code>
O2	<code>objective(A, G) :- deleChain(B, A, G), is_responsible(A, G), objective(B, G).</code>
O3	<code>objective(A, G1) :- andDecomposition(G, G1), objective(A, G).</code>
O4	<code>objective(A, G1) :- orDecomposition(G, G1), objective(A, G).</code>
Actor's entitlements	
E1	<code>is_responsible(A, G) :- objective(A, G), can_achieve(A, G), not not_leaf(G).</code>
Actor's capabilities	
C1	<code>capable_achieve(A, G):-play(A, R), capable_achieve(R, G).</code>
C2	<code>capable_achieve(R1, G):-is_a(R1, R2), capable_achieve(R2, G).</code>
C3	<code>can_achieve(A, G) :- capable_achieve(R, G).</code>
C4	<code>can_achieve(A, G) :- deleChain(A, B, G), capable_achieve(B, G).</code>
C5	<code>can_achieve(A, G) :- orDecomposition(G, G1), can_achieve(A, G1).</code>
C6	<code>can_achieve(A, G) :- andDecomposition(G, G1), andDecomposition(G, G2), can_achieve(A, G1), can_achieve(A, G2), G1 != G2.</code>
C7	<code>producer(A, I, 0) :- is_responsible(A, G), produce(G, I, 0).</code>
C8	<code>reader(Type, POU, A, I):- is_responsible(A, G), read(Type, POU, BType, G, I) .</code>
C9	<code>sender(T, A, B, I):- is_responsible(A, G), send(T, G, B, I) .</code>
C10	<code>modifier(A, I):- is_responsible(A, G), modify(G, I) .</code>
C11	<code>has(A, I, 0) :- producer(A, I, T).</code>
C12	<code>has(A, I, T3) :- prvChain(Type, T1, B, A, I), has(B, I, T2), #int(T1), #int(T2), #int(T3), T3=T1+T2.</code>
C13	<code>has_perm(_, A, I) :- own(A, I).</code>
C14	<code>has_perm(r, A, I) :- dele_perm_chain(A, B, _, r, _ , _, I), has_perm(r, A, I) .</code>

Actors' capabilities axioms: axioms C1-12 are used to identify actors' capabilities. C1 states that an actor is capable of achieving a goal, if the actor plays a role that has such capability. While C2 states that a role is capable of achieving a goal, if the role is specialized of a role that has such capability. C3 states that an actor can achieve a goal, if the actor has the required capabilities to achieve such goal by itself. Moreover, C4 states that an actor can achieve a goal, if it delegates the goal to an actor who has the capability to achieve it. C5 states that an actor can achieve a goal, if the goal is or-decomposed, and the actor has the capability of achieving at

least one of the sub-goals. C6 states that an actor can achieve a goal, if the goal is and-decomposed, and the actor has the capability of achieving all its sub-goals. C7-9 state that an actor is a producer/ reader / sender / modifier of an information item, if the actor is responsible of achieving the goal that produces/ reads/sends/ modifies such information. C11 states that an actor has information, if it is its producer, and C12 states that an actor has information, if such information was provided to it.

4.2.2 Information Quality analysis axioms

In this section, we present IQ axioms in Table 4.4, and discuss each of them as follows:

Goal-information axioms: axioms GI1-12 are used to capture the relations between a goal and information it uses. For instance, GI1-GI8 state that a goal is dependent, if it produces, reads, modifies, and/or sends information. GI9-GI12 state that a goal might not be achieved (prevented), if it did not use information item as required (e.g., produces, reads, modifies, and/or sends).

Goal-information axioms:

Table 4.4: Information Quality analysis axioms

Goal-information axioms	
GI1	<code>dependent(G):- produce_dependent(G, I).</code>
GI2	<code>dependent(G):- read_dependent(G, I).</code>
GI3	<code>dependent(G):- modify_dependent(G, I).</code>
GI4	<code>dependent(G):- send_dependent(G, I).</code>
GI5	<code>produce_dependent(G, I):- produces(G, I, T).</code>
GI6	<code>read_dependent(G, I):- read(POU, G, I).</code>
GI7	<code>modify_dependent(G, I):- modify(G, I).</code>
GI8	<code>send_dependent(G, I):- send(T, G, B, I).</code>
GI9	<code>prevented(G):- modify_prevented(G, I).</code>
GI10	<code>prevented(G):- produce_prevented(G, I).</code>
GI11	<code>prevented(G):- send_prevented(G, I).</code>
GI12	<code>prevented(G):- read_prevented(G, I).</code>
Modifier axioms	
M1	<code>modify_prevented(G, I):- modify(G, I), is_responsible(A, G), not has_perm(m, A, I).</code>
Producer axioms	

-
- P1 produce_prevented(G, I):- produces(TY, G, I, T), is_responsible(A, G), not fits_produce(A, I).
- P2 fits_produce(A, I):- producer(A, I, T), is_responsible(A, G), has_perm(p, A, I), accurate_produce(A,I).
- P3 accurate_produce(A,G):- is_responsible(A, G), produce(chk_blv, G, I, T).
-

Sender axioms

- S1 send_prevented(G, I):- send(T, G, B, I), not fits_send(T, G, B, I).
- S2 fits_send(T, G, B, I):- is_responsible(A, G), send(T, G, B, I), fits_sender(T, A, B, I).
- S3 fits_sender(T, A, B, I):- accurate_send(T, A, B, I), complete_send(T, A, B, I), valid_send(T, A, B, I).
- S4 accurate_send(T, A, B, I):- sender(T, A, B, I), not unauthorized_modify(I).
- S5 complete_send(T, A, B, I):- sender(T, A, B, I), prvChain(ip, Tr, A, B, I).
- S6 valid_send(T, A, B, I):- sender(T, A, B, I), prvChain(_, Tr, A, B, I), #int(T), #int(Tr), Tr <= T.
- S7 unauthorized_modify(I):- modifier(A, I), own(B, I), not trustPerm(trust, A, B, modify, I).
-

Reader axioms

- R1 read_prevented(G, I):- read(POU, G, I), not fits_read(POU, G, I).
- R2 fits_read(POU, G, I):- is_responsible(A, G), read(POU, G, I), fits_reader(A, I).
- R3 fits_reader(A, I):- accessible_read(A, I), accurate_read(A, I), complete_read(A, I), valid_read(A, I), consistent_read(A, I).
-
- R4 accessible_read(A, I):- reader(POU,A, I), has_perm(r, A, I).
-
- R5 accurate_read(A, I) :- reader(Type, POU, BType, A, I), has(A, I, T), not inaccurate(A, I).
- R6 inaccurate(A, I):- read(Type, POU, no_chk_blv, G, I), reader(Type, POU, BType, A, I).
- R7 inaccurate(A, I):- reader(Type, POU, BType, A, I), has(A, I, T), unauthorized_modify(I).
- R8 inaccurate(A, I):- reader(Type, PoU, BType, A, I), producer(B, I, T), not trust_produce(A,B,I).
- R9 trust_produce(A, B, I):- reader(Type, PoU, BType, A, I), producer(B, I, T), not threat_source(B, I).
- R10 trust_produce(A, B, I):- reader(Type, PoU, BType, A, I), producer(B, I, T), threat_source(B, I), monitorChain(A, B, I).
- R11 threat_source(A, I):- threaten(Th, I), capable_threat(A, Th), motivation(positive, A, Th).
- R12 motivation(positive, A, Th):- threat_contribute(positive, Th, 0), objective(A, 0).

```

R13 motivation(negative, A, Th):- threat_contribute(negative, Th, 0),objective(A, 0).
R14 complete_read(A,I):- reader(Type, PoU, BType, A, I), complete_value(A, I), complete_pou(A, I).
R15 complete_value(A, I):- producer(A, I,_).
R16 complete_value(A, I):- reader(_, _, _, A, I), has(A, I, _), producer(B, I, 0), not prvChain(ip, T, B, A, I), #int(T).
R17 complete_pou(A, I):- reader(Type, PoU, BType, A, I), has(A, I, _), composedOfOne(I), partOf(I, I1), has(A, I1,_).
R18 complete_pou(A, I):- reader(Type, PoU, BType, A, I), has(A, I, _), composedOfTwo(I), partOf(I, I1), partOf(I, I2), has(A, I1,_), has(A, I2,_), I1 != I2.
R19 composed(I):- composedOfOne(I).
R20 composed(I):- composedOfTwo(I).
R21 composedOfOne(I):- numOfParts(I,1).
R22 composedOfTwo(I):- numOfParts(I,2).
R23 numOfParts(I, X):- partOf(I,I1),#count{Z:partOf(I,Z)}=X.

```

```

R24 valid_read(A, I):- producer(A,I, _).
R25 valid_read(A, I):- reader(POU, A, I), read_time(T, A, I), info(I, V), #int(T), #int(V), V >= T.

```

```

R26 consistent_read(A, I):- only_reader(A, I).
R27 only_reader(A, I):- reader(_, _, _, A, I), numOfReaders(X, I), X = 1.
R28 numOfReaders(X, I):- reader(_, _, _, A, I), #count{Z: reader(_, _, _, Z ,I)} = X.
R29 consistent_read(A, I) :- reader(_, _, _, A, I), not only_reader(A, I), not inconsistent_reader(A, I).
R30 inconsistent_reader(A, I):- interdependent_readers(A, B, I), read_time(X, A, I), read_time(Y, B, I), #int(X), #int(Y), X != Y, A!=B.
R31 interdependent_readers(A, B, I):- reader(Type, POU, _, A, I), reader(Type, POU, _, B, I), A!=B.
R32 interdependent_readers(B, A, I):- interdependent_readers(A, B, I).
R33 read_time(T, A, I):- reader(_, _, _, A, I), has(A, I, T).

```

Modifier axioms: axiom M1 states that a goal will be prevented from being achieved, if it need to modifies an information item, and the actor who is responsible of the goal achievement does not has the required permissions.

Producer axioms: P1-3 axioms state that a goal will be prevented from being achieved, if it need to produce an information item, and the actor who is responsible of the goal achievement does not has the required permissions (P2), and/or produce process do

not apply any believability check to verify the accuracy of the produced information (P3).

Sender axioms: S1-7 axioms state that a goal will be prevented from being achieved, if it need to send an information item to a particular destination within a defined period of time, if the send process does not satisfy the senders needs (S 1-2). S3 axiom specifies such needs as information should be accurate, complete, and valid at its final destinations from the perspective of the sender, where information is accurate at its end destination, if it was not modified in an unauthorized way (S4, S7). In addition, S5 states that information is complete (value complete) at its destination, if it has been transferred through Integrity Preserving (IP) provision only that guarantee its completeness against corruption and lost. Finally, S6 states that information is valid in term of time at its destination, if it has been transmitted through a provision time that is smaller than its required send time.

Reader axioms: R1-33 axioms are used to analyze whether information fits for purpose of read from the reader perspectives or not.

- R1-2 axioms state that a goal will be prevented from being achieved, if it reads an information item, and such information does not fit for use from the perspectives of its reader, where such information should be accessible, accurate, complete, valid, and consistent to fit for the purpose of use (R3).
- R4 axiom states that an information item is accessible from the reader perspectives, if it has the required read permissions.
- R5-13 axioms are used to analyze information accuracy from its reader perspective. For instance, R5 axiom states that information is considered accurate from the perspective of its reader, if there is no reason to consider it as inaccurate, where information can be considered inaccurate, if it is read without checking its believability (R6). In addition, information is inaccurate, if it was modified in an unauthorized way (R7).
- R8 axiom states that information reader consider information it reads as inaccurate, if no trust produce holds between the reader and information producer.
- R9-13 axioms are used to analyze trust produce in producer from the reader perspective. Axiom R9 states that a trust produce holds, if information reader consider that information producer is not a threat source for such information, or it is a threat source but the reader apply the required monitoring mechanisms to guarantee that the producer will not produce inaccurate information (R10).

- R11 states that an actor can be a threat source for the accuracy of an information item, if there is a threat that threatens such information, and the actor has the required capabilities, and positively motivated to achieve such threat.
- R12 state that an actor is positively motivated to achieve a threat, if such threat contributes positively to an least one of its objectives (goals). While R13 state that an actor is negatively motivated to achieve a threat, if such threat contributes negatively to an least one of its objectives (goals).
- R14-23 axioms are used to analyze information completeness from its reader perspective. For instance, R14 axiom states that information is considered complete from the perspective of its reader, if it is complete concerning its value and Purpose Of Use (POU).
- R15-16 state that information value is complete from the perspective of its reader, either if it is the producer of such information (P15), or it has been delivered o it by IP provision (P16).
- R17-18 state that information is complete for POU from the perspective of its reader, if it is composed of one item, and the actor has such sub item (R17), or if information is composed of two sub items, and the reader has both of them (P18).
- R19-20 state that information is composed, if it is composed of one sub item (R19), or two sub items (R20), where information is composed of one information item, if it has one sub item (R21), and it is composed of two information items, if it has two sub items (R22).
- R23 states that the number of information sub items, is equal to the number of part of relations that can be found in the model concerning such information.
- R24-25 axioms are used to analyze information validity from its reader perspective, where it is valid if it is produced by its reader, i.e., it is not possible to produce invalid information (R24), and it is valid if its currency is smaller than its volatility (R25).
- R26-33 axioms are used to analyze information consistency from its reader perspective. For instance, R26 axiom states that information is consistent from the perspective of its reader, if it is an only reader for such information, i.e., there is no other information item to be consistent with. R27 state that an information reader is an only reader for a specific information item, if it reads such information, and the number of reader in the model for such information is equal to one (R28).

Table 4.5: Social Relations Axioms

S1	prvChain(Type, T, A, B, I):- provide(Type, T, A, B, I).
S2	prvChain(Type, Z, A, C, I):- prvChain(Type, X, A, B, I), prvChain(Type, Y, B, C, I), #int(X), #int(Y), #int(Z), Z = X + Y.
S3	deleChain(A,B,G) :- delegate(A,B,G).
S4	deleChain(A,C,G) :- delegate(A,B,G), deleChain(B,C,G).
S5	dele_perm_chain(A, B, P, R, M, S, I) :- delegate_perm(A, B, P, R, M, S, I).
S6	dele_perm_chain(A, B, P, R, M, S, I) :- dele_perm_chain(A, C, P, R, M, S, I), dele_perm_chain(C, B, P, R, M, S, I).
S7	trustChain(A, B, G):- trust(A, B, G)
S8	trustChain(A, C, G):- trustChain(A, B, G), trustChain(B, C, G)
S9	trust_perm_chain(A, B, TP, TR, TM, TS, I):- trust_perm(A, B, TP, TR, TM, TS, I).
S10	trust_perm_chain(A, B, TP, TR, TM, TS, I) :- trust_perm_chain(A, C, TP, TR, TM, TS, I), trust_perm_chain(C, B, TP, TR, TM, TS, I).
S11	monitorChain(A, B, M):- monitor(A, B, M).
S12	monitorChain(A, C, M):- monitorChain(A, B, M), monitorChain(B, C, M)

- R29-33 axiom state that an information item is consistent from the perspective of its reader, if it is not an only reader for such information, but it is also not an inconsistent reader (R29). A reader can be considered as an inconsistent reader for an information item, if there is another interdependent reader and they do not have the same read time (R30).

R31-32 axioms are used to capture interdependent reader that can be defined as actors who read the same information for the same purpose of use, and R33 axiom is used to capture the actual information read time from the perspective of its reader.

4.2.3 Actors' social interactions

Table 4.5 lists the actors' goals/ information/ permissions/ monitoring axioms. For instance, axioms S1-2 are used to capture information provision among actors, S3-4 are used to capture goal delegation, and permissions delegation is captured by S5-6. Moreover, it lists trust axioms concerning goal delegation (S7-8) and permission delegation (S9-10). Finally, axioms S11-12 capture monitoring chains among actors concerning goals and information.

Table 4.6: Trust analysis axioms

T1	<code>trust(trust, A, B, G):- trustReq(trust, A, B, G), motivation(positive, B, G), not trust_threat(A, B, G).</code>
T2	<code>trust(trust, A, B, G):- trustReq(trust, A, B, G), motivation(negative, B, G), monitorChain(A, B, G).</code>
T3	<code>trust(trust, A, B, G):- trustReq(trust, A, B, G), trust_threat(A, B, G), monitorChain(A, B, G).</code>
T4	<code>trust(distrust, A, B, G):- trustReq(distrust, A, B, G), motivation(negative, A, G).</code>
T5	<code>trust(distrust, A, B, G):- trustReq(distrust, A, B, G), trust_threat(A, B, G).</code>
T6	<code>trust_threat(A, B, G):- trustReq(trust, A, B, G), threat_source(B, G).</code>
T7	<code>threat_source(A, G):- threaten(Th, G), capable_threat(A, Th), motivation(positive, A, Th).</code>
T8	<code>motivation(positive, A, G):- contribute(positive, G, G1), objective(A, G1).</code>
T9	<code>motivation(negative, A, G):- contribute(negative, G, G1), objective(A, G1).</code>
T10	<code>motivation(positive, A, Th):- threat_contribute(positive, Th, G), objective(A, G).</code>
T11	<code>motivation(negative, A, Th):- threat_contribute(negative, Th, G), objective(A, G).</code>

4.2.4 Trust analysis axioms

Table 4.6 lists trust analysis axioms. For instance, T1 states that a trust relation (extensional predicate) holds between a trustor and a trustee concerning the achievement of a goal, if trustee is positively motivated to achieve such goal, and the trustee is not a threat source for the trust relation. T2 states that a trust relation holds between a trustor and a trustee concerning the achievement of a goal, even if the trustee is negatively motivated to achieve the trustum, if the trustor monitors the trustee. T3 states that a trust relation holds between a trustor and a trustee, if the trustor believes that the trustee is a threat source to the trust relation, but there is a monitoring relation between the trustor and the trustee concerning the trustum (goal).

T4 states that a distrust relation holds between a trustor and a trustee concerning the achievement of a goal, if the trustee is negatively motivated to achieve the goal, or the trustor consider the trustee as a threat to the trust relation (T5). T6 states that a trustor consider a trustee as a threat to a trust relation concerning the achievement of a goal, if the trustee represent a threat source to the trustum (goal), where it can be a threat source to the trustum, only if, there is a threat that threaten the achievement of the trustum, and the trustee has the capability to achieve it, and it is positively motivated to achieve such threat (T7). T8-9 state that an actor is positively/ negatively motivated to achieve a goal, if such goal contributes positively/ negatively to at least one of its objectives. Similarly, T10-11 state that an actor is positively/ negatively motivated to achieve a threat, if such threat contributes positively/ negatively to at least one of its objectives.

4.2.5 Goals achievement axioms

Table 4.7 lists axioms that are used to identify whether a goal is achieved or not from the perspective of the actor, who aims for it. A1 states that a goal is achieved from the

perspective of the actor who aims for it, if the goal is not information dependent and the actor took the responsibility of achieving it by itself. A2 states that a goal is achieved for an actor, if the goal is information dependent, but not prevented, and the actor took the responsibility of achieving it by itself. A3 states that a goal is achieved, if the goal is achieved from the perspective of the actor who aims for it. A4 states that a goal is achieved for an actor, if the goal is delegated to another actor, who has the capability to achieve it, and a trust relation holds between the delegator and delegatee concerning the delegated goal. A5-6 state that a goal is achieved for an actor, if one (or decomposition) or all (and decomposition) of its sub goals is/are achieved from the perspective of the actor.

Table 4.7: Goals Achievement Axioms

A1	<code>achieve(A,G) :- is_responsible(A,G), not dependent(G).</code>
A2	<code>achieve(A,G) :- is_responsible(A,G), dependent(G), not prevented(G).</code>
A3	<code>achieved(A,G) :- achieve(A,G).</code>
A4	<code>achieved(A,G) :- deleChain(A,B,G), trustChain(A,B,G), achieve(B,G).</code>
A5	<code>achieved(A,G) :- andDecomposition(G,G1), andDecomposition(G,G2), achieved(A,G1), achieved(A,G2), G1 != G2 .</code>
A6	<code>achieved(A,G) :- orDecomposition(G,G1), achieved(A,G1).</code>

4.3 Verifying the requirements model (properties of the design)

In this section, we define a set of properties (shown in Table 4.8) that can be used to verify the correctness and consistency of the requirements model, i.e., these properties define constraints that the designers should consider during the system design to guarantee the correctness and consistency of the requirements model. In what follows, we discuss each of these properties:

- **Pro1** states that the model should not include any top-level goal that is not *achieved* from the perspective of the actor, who aims for it. However, top-level goal might not be achieved due to several reasons (e.g., not achieving their sub-goals, delegating them with no trust, etc.). Yet we rely on this property to quickly verify the correctness and consistency of the requirements model, i.e., if this property holds for all top-level goals, we can conclude that all the stakeholders' requirements will be achieved.
- **Pro2** states that the model should not include any goal delegation/ delegation chain, if there is no trust/ trust chain between the delegator and the delegatee, or there is

at least a compensation of the lack of trust among them, since delegation with no trust leaves the delegator with no guarantee that its goal will be achieved.

- **Pro3** states that the model should not include any false goal delegation / delegation chain, i.e., goals should be delegated only to actors, who have the capability to achieve them either directly or indirectly by delegating them to an actor who has such capability.
- **Pro4** states that actors should have all information that is required for the achievement of the goals they are responsible of
- **Pro5** states that information should be only provided to actors, who require them either for achieving their objectives, or they have a valid provision chain to actors who require them.
- **Pro6** states that actors should have all permissions they need to achieve their objectives.
- **Pro7** states that permissions should be only delegated to actors, who require them for the achievement of their objectives. Only information owners may have permissions they do not require.
- **Pro8** states that the model should not include actors who delegate permissions that they do not have.
- **Pro9** states that the model should not include actors who have permissions, and there is no trust/ trust chain between such actors and information owner concerning the delegated permissions.
- **Pro10** states that the model should not include produced information that is not accurate from the perspective of its producer.
- **Pro11-13** state that the model should not include send information that is not accurate, complete and valid at its destination from the perspective of its sender.
- **Pro14-17** state that the model should not include information to be read that is not accessible, accurate, complete, valid and consistent at from the perspective of its reader.
- **Pro18** states that the model should not include trust/ distrust conflicts concerning both goal achievement and information provision. In other words, the model is able to detect whether trust/ distrust relations in the model will actually hold or not.

Table 4.8: Properties of the design

Goal properties	
Pro1	$\text{:- aims}(A,G), \text{not achieved}(A,G)$
Pro2	$\text{:- deleChain}(A, B, G), \text{not trustChain}(A, B, G)$
Pro3	$\text{:- deleChain}(_,A,G), \text{not can_achieve}(A,G), \text{is_responsible}(C, G), \text{not deleChain}(B, C, G)$
Information availability properties	
Pro4	$\text{:- reader}(_, _, A, I), \text{information}(I, T), \text{not has}(A,I,T), \text{\#int}(T)$
Pro5	$\text{:- prvChain}(_,A,I), \text{reader}(_, _, B, I), \text{not reader}(TP, POU, TB, A, I), \text{not prvChain}(A,B,I)$
Permissions properties	
Pro6	$\text{:- need_perm}(P, A, I), \text{not has_perm}(P, A, I)$
Pro7	$\text{:- has_perm}(P, A, I), \text{not need_perm}(P, A, I), \text{not own}(A, I)$
Pro8	$\text{:- dele_perm}(P, A, B, I), \text{not has_perm}(P, A, I)$
Pro9	$\text{:- has_perm}(P, B, I), \text{own}(A,I), \text{not trust_perm_chain}(P, A, B, I)$
IQ properties	
Pro10	$\text{:- producer}(A, I, T), \text{not accurate_produce}(A,I)$
Pro11	$\text{:- sender}(T, A, B, I), \text{not accurate_send}(T, A, B, I)$
Pro12	$\text{:- sender}(T, A, B, I), \text{not complete_send}(\text{Tr}, A, B, I), \text{\#int}(\text{Tr}).$
Pro13	$\text{:- sender}(T, A, B, I), \text{not valid_send}(T, A, B, I)$
Pro14	$\text{:- reader}(T, P, A, I), \text{has}(A, I, T), \text{not accurate_read}(A, I)$
Pro15	$\text{:- reader}(T, P, A, I), \text{not complete_read}(A, I)$
Pro16	$\text{:- reader}(T, P, A, I), \text{not valid_read}(A, I)$
Pro17	$\text{:- reader}(T, P, A, I), \text{not consistent_read}(A, I)$
Trust properties	
Pro18	$\text{:- trustReq}(\text{Type}, A, B, S), \text{not trust}(\text{Type}, A, B, S)$
Pro19	$\text{:- motivation}(\text{positive}, A, G), \text{motivation}(\text{negative}, A, G)$
Pro20	$\text{:- monitorChain}(A, B, M), \text{not trust_threat}(A, B, M)$
Conflict of interest properties	
Pro21	$\text{:- play}(A, R1), \text{play}(A, R2), \text{conflict_roles}(R1, R2, \text{produce}, I), \text{producer}(A, I, T)$
Pro22	$\text{:- play}(A, R1), \text{play}(A, R2), \text{conflict_roles}(R1, R2, \text{read}, I), \text{reader}(A, I)$

- **Pro19** states that the model should not include actors with conflicting motivational beliefs toward a trustum i.e., an actor who is positively and negatively motivated toward achieving the same trustum.
- **Pro20** states that the model should not include a monitoring relation between a delegator and a delegee, if the last cannot be considered as a possible threat source for the subject of delegation.
- **Pro21-22** state that the model should not include any agent that plays conflicting roles in terms of producing and/or reading information. In particular, it is used to ensure that the model manage separation of duties among its actors to avoid any

conflict of interest that may leads to different kinds of vulnerability.

4.4 Chapter summary

In this chapter, we showed how our modeling constructs can be mapped into their corresponding formal specifications that enables for performing the required analysis to verify the correctness and consistency of the requirements model. In particular, we proposed a formalization of all our modeling language constructs in Disjunctive Datalog [BFI⁺09], i.e., such formalization enables for translating the graphical requirements model into formal specifications that can be used as an input for the inference engine (DLV system), and enables for deducing new knowledge and perform the required analysis with the help of the reasoning rules (axioms) that have been proposed and discussed throughout this chapter.

Chapter 5

Automated Information Quality Policy Specification

One of the great mistakes is to judge policies and programs by their intentions rather than their results.

Milton Friedman

Policies are very important to define the expected actors' behavior in any community, i.e., with the absence of policies, actors will be free to determine their own behavior [SD99]. The Oxford English Dictionary [Dic89] defines a policy as “a course of actions or principles adopted by a government, party, individual, etc.”. While security policies can be defined as the rules that governs the behavior of a system [SL02; HPL98], or as in [PP03] as the specification of requirements related to the security properties that a system must provide. More specifically, a policy statement concentrate on the permitted, forbidden and obligated actions to be carried out [SD00]. Relying on the previous definitions, we define IQ policy as a set of rules that control the behavior of actors toward information by defining their permitted, forbidden and obligated activities. However, defining such policies for complex systems is not a trivial task, since they might be used to represent different stakeholder' needs, which might be conflicting with one another, i.e., conflicting needs will lead to conflicting policies.

Several organizational and security policy languages have been proposed in the literature. For instance, Hoagland et al. [HPL98] purpose a visual security policy language called LaSCO that is able to describe constraints on a system, which must hold in certain situations. Steen and Derrick [SD00] introduce a method for expressing enterprise policies in terms of the permitted, forbidden or obligated action to be carried out. While the Policy Description Language (PDL) [LBN99] is an event-based language that uses the event-condition-action paradigm to define a policy as a function that maps a series

of events into a set of actions. Moreover, Security Policy Language (SPL) [RZFG01] is a constraint-based language consisting of four fundamental building blocks: entities, sets, rules and policies, which can be used to express the concepts of permission and prohibition, and some restricted forms of obligation. While Ponder language [DDLS01] is able to support a number of basic policies, including: obligation, information filtering, delegation, and refrain policies. However, the main focus of these approaches is the functionality of the system, and they seem to be limited in addressing IQ related issues properly.

In addition, we can find several policy-based approaches that can be used for improving IQ by defining the permitted/ forbidden actors' activities toward information (e.g., access control policies [SS94], security and reliability policies [SV84], integrity policies [Mot89], etc.). However, most of them put more emphasis on the technical aspects of the system, and leave the social and organizational aspects outside the system's boundary, where different kinds of vulnerabilities might arise. In other words, these approaches do not propose a holistic method that is able to capture the social, organizational along with the technical aspects of IQ. More specifically, they might not be able to satisfy the needs of current complex systems (e.g., socio-technical systems [ET60]). Moreover, these policies might be subject to several different social and/ or organizational aspects. Thus, we need to guarantee that the defined IQ policies are also consistent with the social and organizational context where the system will be employed.

To tackle these problems, we need to guarantee that the defined IQ policies are consistent with the social, organizational, and IQ requirements of the system. This can be done by deriving such policies from the IQ requirements model of the system, after verifying that the model is correct and consistent. The rest of this chapter is organized as follows; first we define our IQ policy specification language that can be used to represent the derived IQ policies 5.1, and then we define several rules that enable for the automatic derivation of such policies from the requirements model 5.2.

5.1 Information Quality Policy Specification Language

Our IQ policy specification language provides a clear way for specifying IQ policies in terms of the permitted, forbidden and obligated activities toward information. The language supports three types of policies, namely: permit, forbid, and obligate policies that are used to control four different types of activities over information, namely, produce, read, modify, and send. The IQ policy specification language can be represented in BNF notation [MR03] as:

IQ_Policy = (**Permit** | **Forbid** | **Obligate**)

IQ_Policy = **IQ_Policy_name**(Actor: a, [Actor: b,], information: i {[, (**for** | **to** | **in**), (Actor: c | Goal: g | Time: t)]})

The syntax of the language can be described as follows, **bold** is a language keyword, definitions are represented as =, alternations are enclosed in round brackets () separated by |, optional elements are enclosed within square brackets [], and repetition is enclosed with braces {}. In what follows, we discuss the three types of IQ policies:

Permit policy is used to define the activities that an actor is allowed to perform over information, and it can be represented in BNF notation as follows:

Permit = **permitted_policy_name**(Actor: a, information: i [, (**for** | **to**), (Actor: c | Goal: g)])

For example, stock investor (information owner) is permitted to produce its own *orders*, and a *trader*¹ is permitted to read and modify *investor's order* for its goal *make profit by producing the right orders*, and it is permitted to send *investor's order* to a *stock market*, each of the previous permit policies can be represented as follows:

```
permitted_produce(investor, investors_order)
permitted_read(trader, investors_order, for, make_profit_by_producing_orders)
permitted_modify(trader, investors_order, for, make_profit_by_producing_orders)
permitted_send(trader, investors_order, to, stock_market)
```

Forbid policy is used to define the activities that an actor is prohibited to perform over information, and it can be represented in BNF notation as follows:

Forbid = **forbidden_policy_name**(Actor: a, information: i [, (**for** | **to**), (Actor: b | Goal: g)])

Consider for example, a *trader* that is forbidden to produce some *investor's order* (e.g., does not has the required permissions), and it is forbidden to read/ modify *investor's order* for some goals, and it is forbidden to send *investor's order* to some actors, we can represent the previous statements in the form of forbid policies as follows:

```
forbidden_produce(trader, investors_order)
forbidden_read(trader, investors_order, for, analyzing_the_market)
```

¹We assume it has the required permissions

```
forbidden_modify(trader, investors_order, for, analyzing_the_market)
forbidden_send(trader, investors_order, to, consulting_firm)
```

Obligate policy is used to specify the activities that an actor must perform over information, and it can be represented in BNF notation as follows:

```
obligate = obligated_policy_name(Actor: a, [Actor: b,] information: i {[,
(for | to | in), (Actor: c | Goal: g | Time: t)]})
```

Obligation policies cover three different types of activities, namely: (1) provide, in which an actor is obligated to provide information to another one within a predefined period of time; (2) read, in which an actor is obligated to read information within a predefined period of time delay; and (3) send, in which an actor is obligated to transfer information to its defined destination (another actor) within a predefined period of time. For example, an obligation policy concerning information provision, in which a *stock market* (e.g., NYSE, CME, etc.) is obligated to provide *CTA* with information concerning trades/ quotes (e.g., *CQS/CTS info*) in the predefined period of time (e.g., 10 seconds), can be represented as:

```
obligated_provide(stock_market, CTA, CQS-info, in, 10)
```

While an obligation policy concerning information read, in which an actor (e.g., *NYSE*) is obligated to read information (e.g., *CME CB-info*) within a predefined period of time delay (e.g., 0 seconds) , can be represented as:

```
obligated_read(NYSE, CME_CB info, for, manage_trading_environment, in, 0)
```

Finally, an obligation policy concerning information send, in which an actor (e.g., *trader*) is obligated to send information (e.g., investor's order) to a predefined destination (e.g., stock market) within a predefined period of delay (e.g., 10 seconds) , can be represented as:

```
obligated_send(trader, investors_order, to, Stock_market, in, 10)
```

5.2 Rules for Automated Derivation of Information Quality Policy Specifications

In the previous section, we described our IQ policy specification language. Here, we discuss how the final IQ specifications can be derived from the requirements model.

Table 5.1: Rules for Automated Derivation of IQ Specifications

Permit policies	
P1	<code>permitted_produce(A,I):- has_perm(p,A,I), not forbidden_produce(A,I).</code>
P2	<code>permitted_read(A,I,for,G):- own(A,I), is_responsible(A,G), not forbidden_read(A,I,for,G).</code>
P3	<code>permitted_read(A,I,for,G):- has_perm(r,A,I), is_responsible(A,G), read(POU,G,I), not forbidden_read(A,I,for,G).</code>
P4	<code>permitted_send(A,I,to,B):- has_perm(s,A,I), is_responsible(A,G), send(T,G,B,I).</code>
P5	<code>permitted_modify(A,I,for,G):- has_perm(m,A,I), is_responsible(A,G), modify(G,I).</code>
Forbid policies	
F1	<code>forbidden_produce(A,I):- play(A,R1), play(A,R2), conflict_roles_produce(R1,R2,produce,I).</code>
F2	<code>forbidden_read(A,I,for,G):- play(A,R1), play(A,R2), is_responsible(A,G), read(POU,G,I), conflict_roles_read(R1, R2, read, I).</code>
F3	<code>forbidden_read(A,I,for,G1):- has_perm(r,A,I), is_responsible(A,G), read(POU1,G,I), is_responsible(A,G1), not read(POU2,G1,I), G != G1.</code>
F4	<code>forbidden_modify(A,I,for,G1):- has_perm(m,A,I), is_responsible(A,G), modify(G,I), is_responsible(A,G1), not modify(G1, I), G != G1.</code>
F5	<code>forbidden_send(A,I,to,C) :- has_perm(s,A,I), sender(T1, A, B, I), actor(C), not sender(T2, A, C, I), #int(T1), #int(T2), B!=C.</code>
Obligate policies	
O1	<code>obligated_read(A,I,in,0):- interdependent_readers(A,B,I).</code>
O2	<code>obligated_provide(A,B,I,in,T):- provideChain(_,T,A,B,I).</code>
O3	<code>obligated_send(A,I,to,B,in,T):- sender(T,A,B,I).</code>

In particular, we define three sets of rules² (shown in Table 5.1), namely: permit, forbid and obligate policy derivation rules that are used for the automated derivation of IQ specifications from the requirements model in terms of IQ policy specification language; in what follows we discuss each of these sets:

Permit policy derivation rules : are used to derive allowed actors' activities concerning information, and represent them in IQ specification language (*permit policies*). In particular, rules **P1-5** are used to identify the permitted actors' activities over information based on the permissions they have taking into consideration that actors that might be prevented from performing some activities (e.g., conflict of interests).

P1: states that an actor is permitted to produce an information item, if it has the produce permission, and it is not forbidden, by any reason, from producing such information.

P2: states that an actor is permitted to read an information item for any goal that is

²Derivation rules (axioms) that are also represented in DLV language [BFI⁺09] (Datalog with Disjunction)

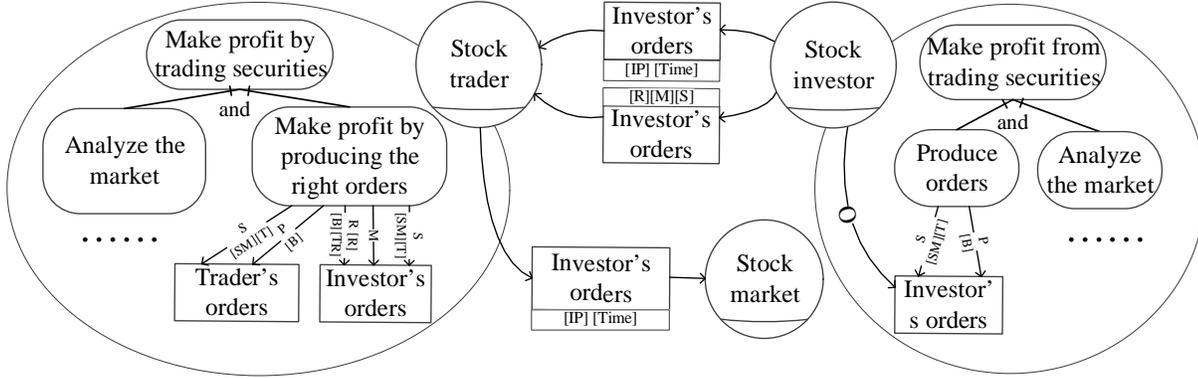


Figure 5.1: A partial goal model concerning the stock market structure

responsible of its achievement, if it is the owner of such information, and there is no reason forbidding it from reading such information.

P3: states that an actor is permitted to read an information item for a specific goal that is responsible of its achievement, if it has the related read permissions, and it is not forbidden, by any reason, from reading such information.

P4: states that an actor is permitted to send an information item to a specific actor, if it is responsible of a goal that sends such information to the actor, and it has the related send permissions.

P5: states that an actor is permitted to modify an information item by a specific goal, if it is responsible of the goal that modifies such information, and it has the related modify permissions.

Example 32. *in Figure 5.1, stock investor (information owner) is permitted to produce, read, and modify its own orders by any goal that is responsible of its achievement, and it is also permitted to send them to any actor. While the trader is permitted to read and modify the investor's orders only by its goal "Make profit by producing the right orders", and it is permitted to send the investor's orders only by its goal "Make profit by producing the right orders", and only to the stock market.*

Forbid policy derivation rules : are used to derive prohibited actors' activities concerning information, and represent them in IQ specification language (*forbid policies*). In particular, rules **F1-5** are used to identify the activities that an actor is forbidden to perform over information.

- F1:** states that an actor is forbidden to produce an information item, if it plays conflicting roles concerning the produce of such information.
- F2:** states that an actor is forbidden to read an information item, if it plays conflicting roles concerning the read of such information.
- F3:** states that an actor is forbidden to read an information item for any goal that is responsible of its achievement except the goal(s) that reads such information, and the actor has been granted the read permissions to achieve it/them. This rule is used to prevent actors from using (reading) information for any goal beside the one they have been granted the read permissions for.
- F4:** states that an actor is forbidden to modify an information item by any goal that is responsible of its achievement except the goal(s) that modifies such information, and the actor has been granted the modify permissions to achieve it/them.
- F5:** states that an actor is forbidden to send an information item to any actor, except the actor(s) that has been granted the send permissions to send information to it/them.

Example 33. *in Figure 5.1, the stock trader is forbidden to read or modify the investor's orders by its goal "analyze the market for targeted securities", and it is forbidden to send the investor's orders by any goal except "analyze the market for targeted securities" to any actor but the stock market.*

Obligate policy derivation rules: are used to derive obligated actors' activities toward information, and represent them in IQ specification language (*obligate policies*). Obligate policies are essential for addressing several IQ related vulnerabilities by defining the activities that actors must perform toward information. For example, if a *stock investor* depends on a *stock trader* for sending some trading orders to a *stock market*. The automated analysis is able to detect situations, where the *trader* is not able to provide the time to market that is required by the *investor*, which is important to determine information validity at its destination. However, even if the *trader* is able to provide the required time to market, it does not guarantee that the *trader* will not postponed sending the orders for some reason, which cannot be addressed by the automated reasoning, since it is hardly recognize the notion of obligation.

Similarly, there is no guarantee that a *stock market* will provide information concerning its trades and quotes to *CTA* with no delay, where any delay in such information might negatively influence the overall performance of the trading system. For instance, Nanex report [Nan10] listed both NYSE-CQS delay along with the DOW Jones delay (was a

result of the first delay) as a reason of the Flash Crash. Finally, the framework proposed *purpose of use*, *interdependent readers*, and *read time* concepts to address information consistency. Yet, such concepts do not guarantee that all *interdependent readers* will use (read) such information without delay, i.e., none of them is obligated to use information when it is received. Without such obligations, it is not guaranteed that the inconsistency problem among the *interdependent readers* will be solved. To this end, we define **O1-3** rules that can be used to derive obligation policies.

O1: states that all *interdependent readers* for information are obligated to read such information with no delay ('0').

Example 34. *considering the flash crash scenario, and in order to guarantee that NYSE and NASDAQ (interdependent readers) will perfectly coordinate their CBs activities to avoid any potential market crash, both of them should be obligated to read CME CB info (information used for CBs coordination) with no delay ('0').*

O2: states that an actor is obligated to another one to provide information within the provision time it claims.

Example 35. *to guarantee that a stock market will provide information concerning its trades and quotes (e.g., CTS-info, CQS-info) to CTA with no delay, we can use an obligation policy concerning information provision, in which a market is obligated to provide CTA with such information within a predefined period of time.*

O3: states that an actor is obligated to send information to its intended destination within the send time it claims.

Example 36. *to guarantee that a trader will not postponed sending investor's orders for some reason, we can use an obligation policy concerning information send, in which a trader is obligated to send investor's orders to trading market within a predefined period of time.*

5.3 Chapter summary

In this chapter, we argued that specifying IQ requirements is not an easy task, and we discussed the limitations of existing approaches for specifying IQ requirements in terms of permitted, forbidden, and obligated activities. In addition, we defined IQ specification language, and we provide mechanisms for the automatic derivation of the final IQ specifications from the requirements model and represent them in our language in terms of IQ policies that define the permitted, forbidden and obligated activities toward information.

Chapter 6

The Methodological Process & Computer-Aided Support: ST-IQ Tool

You know my methods. Apply them.

Arthur Conan Doyle

In this chapter, we describe our proposed RE methodology in 6.1, which should be followed by system designers during the different phases of the system design. In particular, it describes how the different modeling activities should be performed by system designers, how the requirements model can be analyzed against the properties of the design to verify its correctness and consistency, and finally, how the final IQ specifications can be derived from the IQ requirements model. While in 6.2, we presented our CASE tool (ST-IQ Tool) for modeling and analyzing IQ requirements.

6.1 The Methodological Process

The need for a methodological process for guiding of software engineers (designers) while designing the system-to-be is well justified in RE community [Yu95]. In particular, the process should specify who should do what, when and how. Further, the process should indicate how to verify the model, i.e., whether the model satisfies all the stakeholders' requirements or not. In the next section, we provide a detailed description of our proposed methodology.

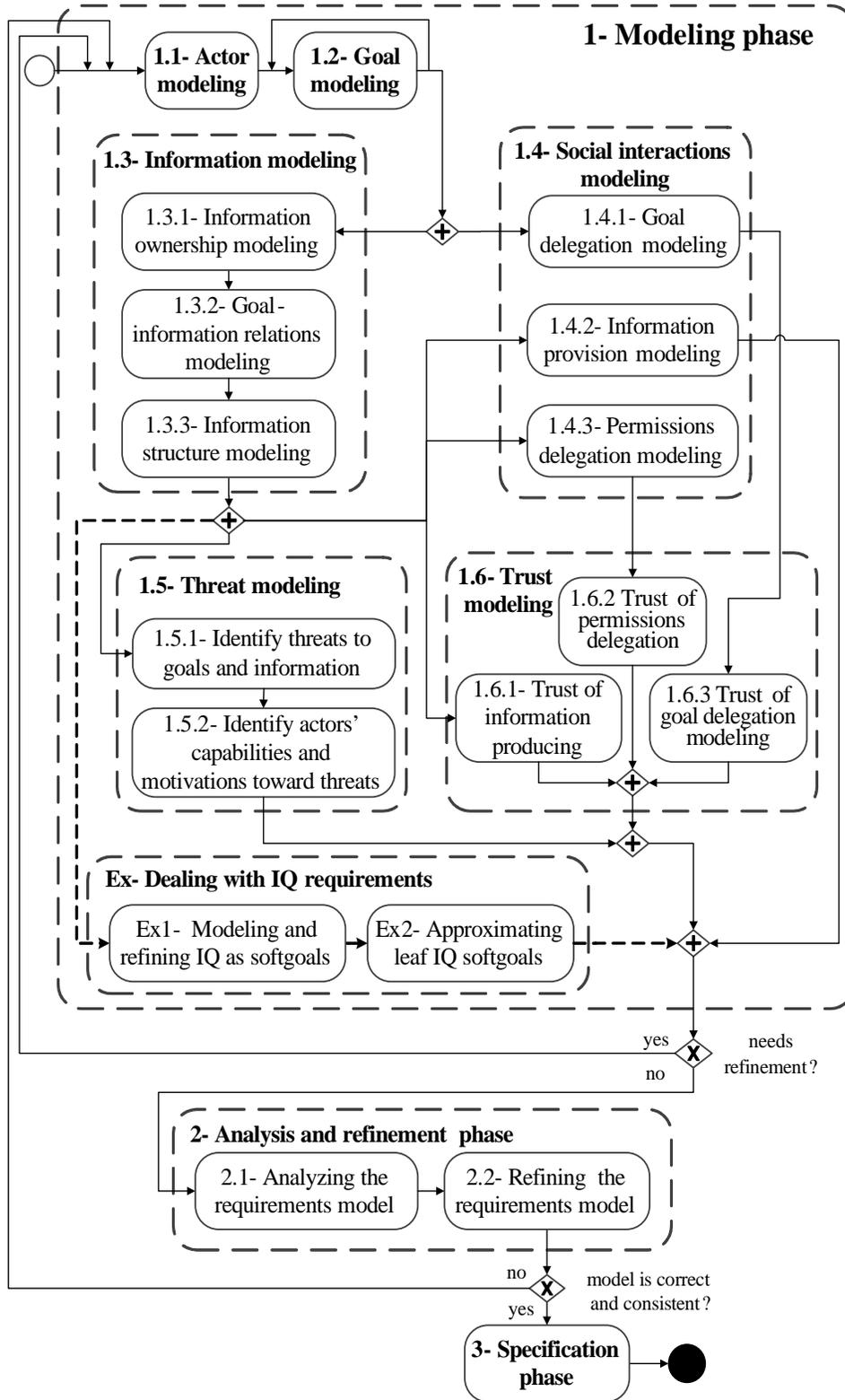


Figure 6.1: The Methodological Process

6.1.1 A detailed description of the methodological process

Our methodological process provides the required support for system designers during the system design, the overall methodological process is depicted in Figure 6.1. In particular, it aims to support designers during the requirements modeling, requirements analysis, and lastly, during the derivation of the final operational specification from the requirements model¹. The process is iterative, i.e., it can be repeated, if required, to refine the model until reaching the target design of the system-to-be. In the rest of this section, we give a detailed description of the methodological process activities along with the activities interdependencies. The process has three main phases; each of them is composed of several activities, in what follows we discuss these phases along with their sub activities:

(1) Modeling phase: aims to assist designers in modeling the system-to-be through several sequential modeling activities, where the outcome of each of these activities is used as an input to the following one(s). The modeling phase is composed of eight main modeling activities; each of them is discussed in details as follows:

1.1 Actor modeling: is the first modeling activity that aims to identify the actors (agents and roles) of the system along with their objectives, entitlements and capabilities. The outcome of the actor modeling activity is the actor diagram that represents the main stakeholders of the system represented in terms of agents and the role(s) they are playing, where each role/ agent has a clear definition, based on which, we can identify their objectives entitlements and capabilities. The actor diagram is used as an input for the goal modeling activity.

1.2 Goal modeling : aims to refine actors' root goals that have been defined in the previous activity through and/or decomposition until reaching their leaf sub goals that are fine enough to be achieved by actors. The outcome of this activity is the goal diagram that represents a refined model of actors' objectives in terms of leaf goals that can be achieved or delegated to other actors, who have the required capabilities to achieve them. The goal diagram is used as an input for 1.3 information modeling activity and 1.4 social interactions modeling (1.4.1 goal delegation modeling activity).

1.3 Information modeling: information modeling is the first activity toward capturing IQ requirements; it aims to identify the different relation between information and other modeling constructs in the system, including information ownership, goal- information relations, and information structure. In particular, this activity is composed of three sub activities:

¹Requirements elicitation is out of the scope of the methodological process

1.3.1 Information ownership modeling: aims to identify the legal owner(s) of information items, which is essential to capture their requirements concerning information they own. Further, such relation is required for identifying who has full control over information use (e.g., information permissions/permissions delegation);

1.3.2 Goal-information relations modeling: aims to identify the different relations between goals and information, where these goals may produce, read, modify, and/or send information. Yet defining such relation is essential for capturing the relation between information and its purpose of use, which is essential for capturing requirements about IQ;

1.3.3 Information structure modeling: aims to capture the internal structure of composite information, which enables to decide whether information is complete or not.

The outcome of this activity is the information diagram that is used as an input for five different activities: 1.4 social interactions modeling (1.4.2 information provision modeling and 1.4.3 permission delegation modeling), 1.5 threat modeling (1.5.1 identify actors' capabilities and motivations toward threats), 1.6 Trust modeling (1.6.1 trust of information producing), and EX- dealing with IQ requirements (EX1- modeling and refining IQ as softgoals).

1.4 Social interactions modeling: aims to identify the different social dependencies among the actors of the system concerning the transfer of entitlements and authorities. As shown in Figure 6.1, social interactions modeling is composed of three main sub activities:

1.4.1 Goal delegation modeling: aims to identify goal delegation among actors of the system, i.e., when an actor does not have the required capabilities to achieve a goal it aims for, it delegates the goal to another actor who has such capability. The outcome of this activity is the goal delegation diagram that identifies the transfer of responsibilities among the actors of the system, and it is used as an input to 1.6.3 trust of goal delegation modeling.

1.4.2 Information provision modeling: aims to identify information transmission among actors of the system, i.e., identify the source and destination of each information transmission relation. In addition, it helps in identifying how the quality of the transferred information might be affected by the provision process, and enables to identify information availability from information readers' perspective.

1.4.3 Permission delegation modeling: aims to identify actors, who have the capability to delegate permissions to other actors of the system, identify

actors who need permissions to achieve their objectives, and identify how actors can delegate permissions among one another. The outcome of this activity is the actors' authorities (permissions) diagram. The outcome of this activity is used as an input for 1.6.2 trust of permissions delegations social trust modeling.

1.5 Threat modeling: aims to identify goals and information related threats along with actors' capabilities and motivations toward such threats. In particular, this activity is composed of two main sub activities:

1.5.1 Identify threats to goals and information related: identify and model intentional threats that might threaten goals and information within the system.

1.5.2 Identify actors' capabilities and motivations toward threats: identify actors' capabilities and motivations toward threats based on the actors' internal structure.

1.6 Trust modeling: aims to identify the trust/ distrust relations among actors of the system concerning information producing, permissions and goals delegations. This activity is composed of three main sub activities:

1.6.1 Trust of information producing: aims to identify trust / distrust relation among actors concerning produced information;

1.6.2 Trust of permissions granting modeling: aims to identify trust / distrust relation among actors concerning their delegated permissions;

1.6.3 Trust of goal delegation modeling: aims to identify trust / distrust relation among actors concerning their delegated goal.

EX. Dealing with IQ requirements modeling: the main aim of this activity is removing any ambiguity while dealing with IQ requirements. In particular, it helps in dealing with IQ requirements by identify how root IQ requirement (IQ softgoals), and refine them until reaching their operational specifications. This activity is composed of two main sub activities:

Ex1. Modeling and refining IQ as softgoals: aims to identify root IQ softgoals, we can identify two sources of root softgoals: (1) root softgoals that are identified by information owners, owners might define different softgoals concerning information they own; (2) root softgoals that resulted from goal-information relations (e.g., produce, read, etc.), and they are used to identify the IQ needs of goals concerning information they use. After identifying root IQ softgals, we refine them until reaching their leaf IQ softgoals based on the actual purpose of use.

Ex2- Approximating leaf IQ softgoals: aims to identify the appropriate approximation of leaf softgoals into the adequate Information Quality Constraints (IQCs) based on their types.

After all the modeling activities are completed, the requirements model is checked to determine whether it needs to be refined or not. If there is a need for refinement, the process repeats the modeling activities again starting from the first activity to refine the model; while if no refinement is needed the process proceed to the analysis phase.

(2) Analysis Phase: aims to verify the correctness and consistency of the requirements model against some properties of the design. In particular, we define a set of properties to check the correctness and consistency of the requirements model, i.e., the model is correct and consistent, if all of these properties hold. When the model correctness and consistency are verified (no inconsistency and/or conflict is detected), we proceed to the final phase. Model analysis is an automated activity, and it is support by the ST-IQ Tool. In particular, this activity is composed of two main sub activities:

2.1 Analyzing the requirements: the main purpose of the analysis phase is detecting any incorrectness or inconsistency in the model (e.g., unachieved goal, information is not available, etc.), and identify their reasons (e.g., delegation without trust, etc.), which enable the system analyst to find proper solutions for them.

2.2 Refining the requirements model: in this activity, analyst tries to resolve any errors (incorrectness and/or inconsistency) in the model that have been detected while analyzing the model. For example, the analyst can compensate the lack of trust in a trustee concerning a delegated goal or permission by *monitoring* the trustee.

After all the analysis activities are completed, if there are non-solved inconsistencies and/ or conflicts among the overall requirements of the system, the process goes back to the modeling phase to reconstruct the model accordingly, otherwise the process continues to the final phase, since the requirements model is correct and consistent.

(3) Specification phase: is the final phase, and it aims to automatically derive the final IQ specifications from the requirements model. More specifically, we define IQ specification language and some rules that allow for the automatic derivation of the final IQ specifications from a requirements model that have been verified correct and consistent, which guarantee, in turn, to derive a correct and consistent

IQ specifications in terms of IQ policies, which are represented in the IQ policy specification language that clearly define the permitted, forbidden and obligated actions to be carried out by actors of the system toward information.

6.2 Computer-Aided Support: ST-IQ Tool

Usually, Computer-Aided Software Engineering (CASE) tools are developed to assist software engineers during the different phases of the system design, and they are considered as a key component of any proposed RE framework [BCM⁺94]. In this section, we describe our CASE tool (ST-IQ Tool) that have been developed to help designers in modeling and reasoning about IQ requirements. The rest of this chapter is organized as follows; ST-IQ Tool architecture is described in 6.2.1, ST-IQ modeling component is discussed in 6.2.2, the Model-To-Text (M2T) component is described in 6.2.3. Finally, in 6.2.4, we discuss its reasoning component.

6.2.1 ST-IQ Tool Architecture

Our CASE tool called Secure Tropos IQ Tool (ST-IQ Tool)² consists of four main components (ST-IQ Tool architecture is shown in Figure 6.2):

- Control component (JAVA based-program) that controls and coordinates the activities of the three other components;
- Graphical user interface (GUI) that enable designers for drawing the requirements model by drag-and-drop modeling elements from palettes, and allows for specifying the properties of the modeling elements along with their interrelations;
- Model-To-Text (M2T) transformation mechanism that supports the translating of the graphical models into Disjunctive Datalog formal specifications;
- Automated reasoning support (DLV system³) takes the Disjunctive Datalog specification that resulted from translating the graphical model along with the reasoning axioms, and then perform the required analysis to verify the correctness and consistency of the requirements model against the properties of the design.

²<https://github.com/disi-unitn-RE-IQ/RE-IQ>

³<http://www.dlvsystem.com/dlv/>

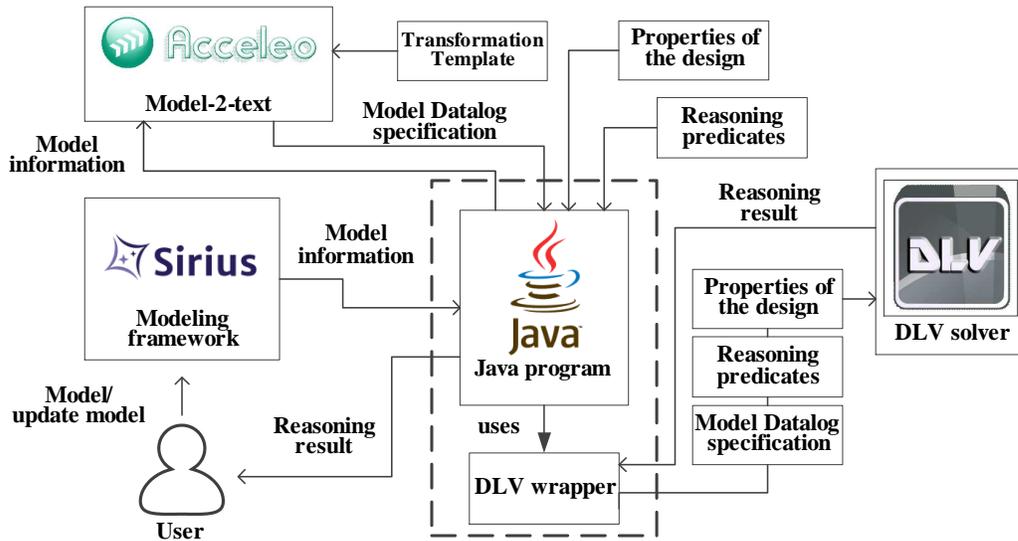


Figure 6.2: ST-IQ Tool architecture

6.2.2 Modeling component

The modeling component (A snapshot is shown in Figure 6.3) has been developed using Sirius⁴, which is an Eclipse project that allows for developing graphical-based modeling framework relying on several Eclipse Modeling technologies (e.g., EMF and GMF). It aims to support designer while drawing the requirements model. In particular, it provides four palettes:

- General pallet that provides the general constructs for creating the requirements model, including: roles, agents, goals, information, etc.
- Social interaction pallet that provides social interaction and dependency constructs, including: goal delegation, information provision, permissions delegation, etc.
- Threat pallet that provides threat related constructs, including: threat, threaten, positive/ negative threat to goal contributions, etc.
- Social Trust pallet that provides trust related constructs, including: trust for goal delegation, threat, threat contribution, etc.

Designers can rely on these palettes for drawing the different requirements diagrams (Actors diagram, goals diagram, social interactions diagram, trust diagram, etc.) by drag-and-drop modeling constructs from the adequate pallet, and define the different

⁴<https://projects.eclipse.org/projects/modeling.sirius>

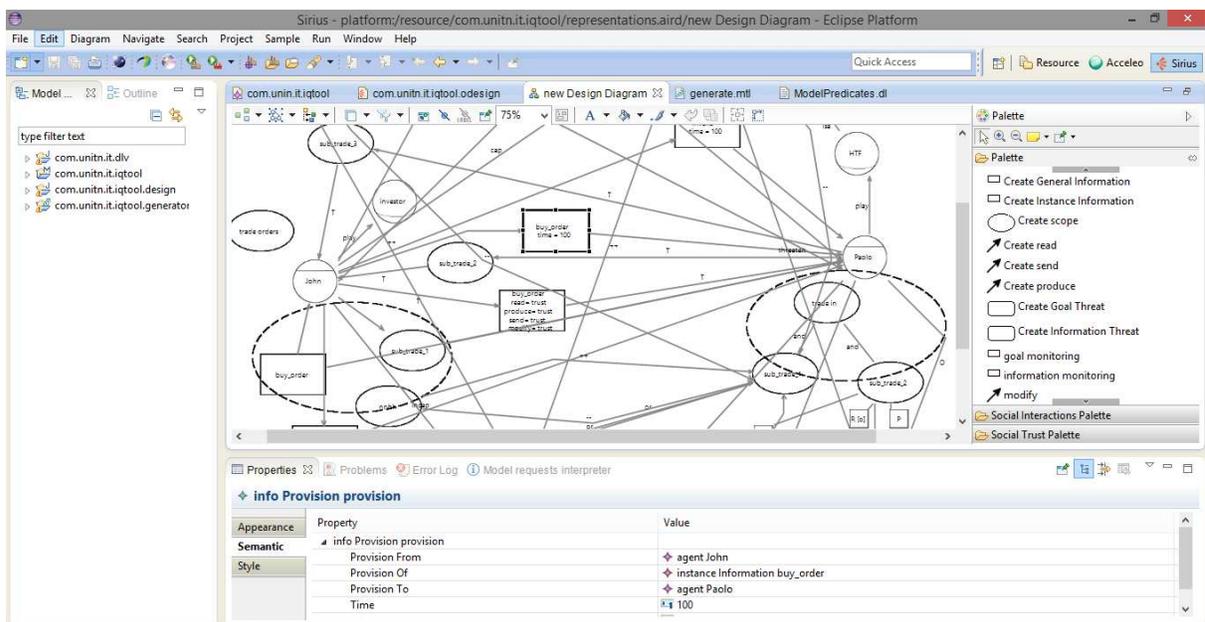


Figure 6.3: ST-IQ Tool: Modeling Interface

relations among these constructs. In addition, the designer is able to specify or modify the properties of modeling elements from the property panel.

6.2.3 Model-To-Text (M2T) component

A modeling language can be used for representing the system requirements in a simple way, which enables the system designer along with the different stakeholders of the system to communicate and understand each other perfectly [NE00]. However, to perform the required reasoning about the IQ requirements model, we need to transform the visual requirements model into formal specifications, which enables for performing the required analysis to verify the correctness and consistency of the requirements model.

To this end, we depend on Acceleo Model-To-Text Transformation Language (MTL)⁵ to translate the graphical (visual) requirements model into Disjunctive Datalog formal specifications. In particular, Acceleo takes the graphical requirements as an input, and applies it to some transformation templates (shown in Figure 6.4) that are able to read the concepts that underlay the visual constructs along with their different relations and properties, and then produce the corresponding formal specifications in terms of Disjunctive Datalog language. A sample of the Acceleo output is shown in Figure 6.5.

⁵<https://projects.eclipse.org/projects/modeling.m2t.acceleo>

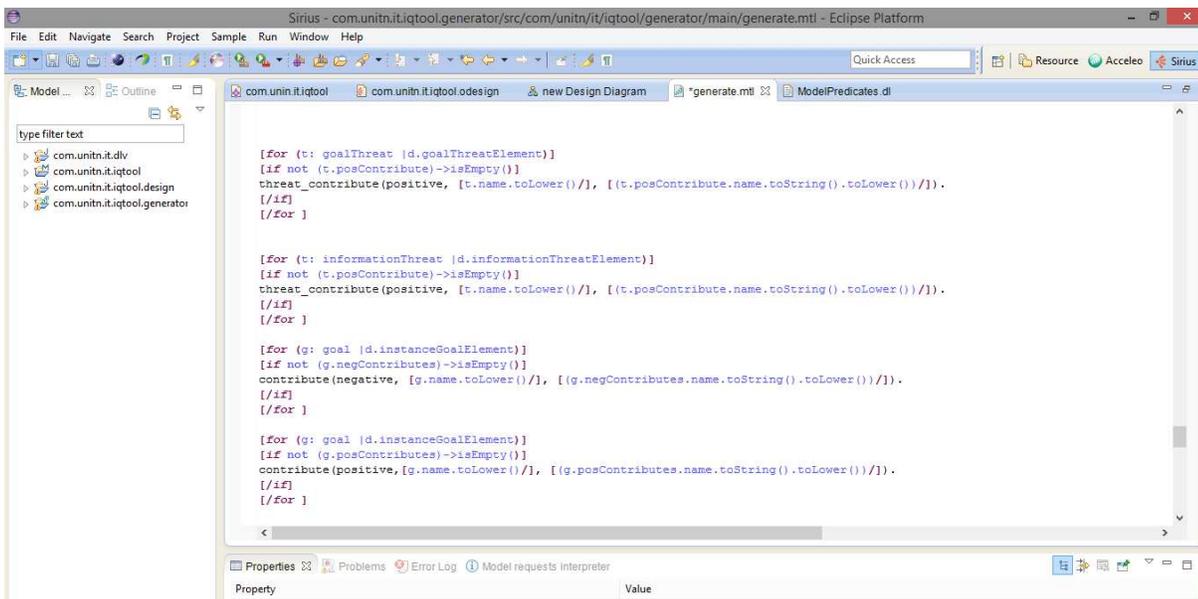


Figure 6.4: ST-IQ Tool: Model-2-text Component

6.2.4 Reasoning component

The main aim of the reasoning components is verifying the requirements model. In particular, the control component, provide the Disjunctive Datalog specification (extensional predicates) that resulted from translating the graphical model along with the reasoning rules (axioms) and the properties of the design to the DLV engine. The DLV engine derive new knowledge (intensional predicates) depending on extensional predicates and the reasoning rules (axioms), and then it uses the extensional and intensional predicates to perform the required reasoning to verify the correctness and consistency of the require-

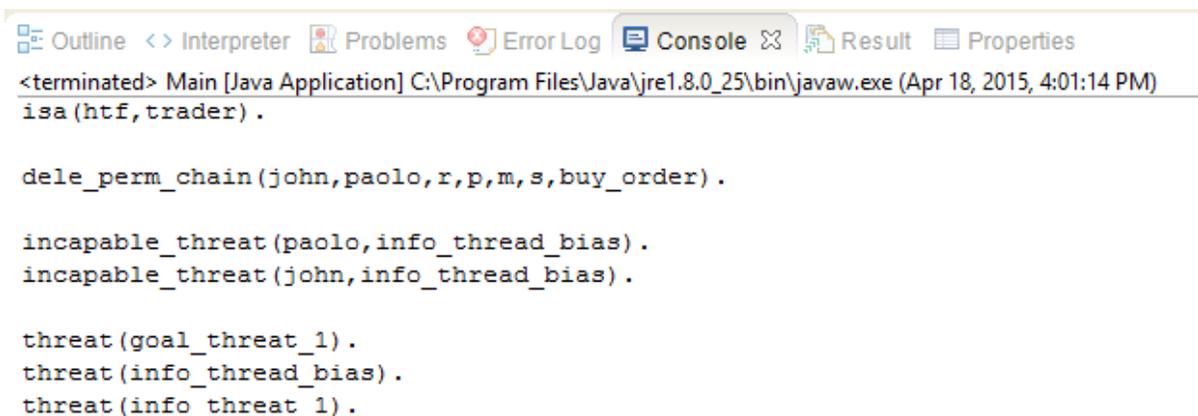


Figure 6.5: Disjunctive Datalog formal specifications

ments model against the properties of the design.

6.3 Chapter summary

In this chapter, we provide a detailed description of our proposed RE methodology should be followed by system designers during the different phases of the system design. In other words, we discussed the different modeling activities that should be performed by system designer, and how the model can be analyzed against the properties of the design to verify its correctness and consistency. Moreover, we discuss how the final operational specification can be derived from the verified requirements model. In addition, we presented the ST-IQ Tool, which can be used for modeling and analyzing IQ requirements.

Chapter 7

Case study: the Flash crash

I have been impressed with the urgency of doing. Knowing is not enough; we must apply. Being willing is not enough; we must do.

Leonardo da Vinci

In this chapter, we present the application of our proposed RE framework to a case study concerning a stock market system crash (the Flash Crash). In particular, we apply our framework to a real case study concerning the United States stock market crash on Thursday May 6, 2010 (the Flash Crash), in which the Dow Jones Industrial Average dropped about 1000 points (about 9%), and then it recover those losses within minutes. In particular, we demonstrate the utility and effectiveness of the framework in identifying and resolving the different vulnerabilities that manifested themselves in the system structure and how they were exploited by some actors of the system, which led or contributed to the crash.

In the rest of this chapter, we start by briefly describing the stock market system structure in 7.1, the Flash Crash chronology of events in 7.1.1, and then we list and discuss the main reasons of the crash 7.1.2. In 7.1.3, we apply our framework to the Flash Crash case study. First, we use our modeling language for modeling the case study in 7.1.4. In 7.1.5, we discuss its analysis technique, i.e., we demonstrate its adequacy for reasoning about IQ requirements. While 7.1.6 describes the automated derivation of the final operational specifications of IQ requirements. Finally, we perform scalability experiments to demonstrate the efficiency of the automated reasoning techniques for dealing with big size models in 7.2.

7.1 The stock market system structure

A stock market system can be defined as an aggregation of buyers and sellers of securities (also called shares), which can be listed on different stock exchanges. Based on [KKST11; Mis03], several stakeholders of the system can be identified, including: *stock investors* are individuals or companies, who have a main goal of “making profit from trading securities”. While *stock traders* are persons or companies involved in trading securities in *stock markets* either for their own sake or on behalf of their *investors* with a main goal of “making profit by trading securities”. *Stock markets* are places where *traders* gather and trade securities, and they have a main goal of “making profit by facilitating security trading”. Furthermore, we can identify several financial companies that provide supporting services (e.g., *accounting firms*, *auditing firms*, *consulting firms*, etc.). Finally, the *Consolidate Tape Association (CTA)* and Consolidated Quotation System (CQS) that provides information concerning trades and quotes¹ respectively.

The stock market is an example of a complex socio-technical system, where IQ is very important for the system efficient performance. In such system, actors interact and depend on one another for information. For instance, *investors* need trading suggestions to assist their trading decision, where the quality of the suggestion information plays a main role in the trading decision that the investor may take. Trading suggestions can be directly provided by *consulting firms*, or through *traders*.

Stock markets receive trading orders from *traders*, and they apply mechanisms to match and perform trades. Moreover, they are responsible of ensuring stable trading environment for the *traders* by slowing down or halting the trading activities when needed, for such task they rely on what is called CB information that can be produced by analyzing the trading activities in the markets. In addition, *markets* produce information about their quotes (Quotation info-CQS) and trades (trades info-CTS) they perform, where such information are used by *Consolidate Tape Association (CTA)* and Consolidated Quotation System (CQS) to produce CTS-info and CQS-info respectively. Both of CTS-info and CQS-info are very important to analyze the trading market and make the right trading decisions.

Each of the system stakeholders has its own objectives it aims to achieve, where achieving some of these objectives is highly influenced with the quality of information it relies on. Thus, the analysis should capture stakeholders' IQ requirements and provide a clear mechanism to verify whether such requirements are achieved or not.

¹A quote is an order that has not been performed

7.1.1 The Flash Crash: chronology of events

The following sequence of events is based on the joint report of CFTC and SEC regarding the market events of May 6, 2010 [Tra10; SC⁺10] and Nanex² Flash Crash summary report [Nan10].

- On May 6, U.S. stock markets opened and trended down for most of the day on worries about the European debt crisis (Greece).
- By 2:30 p.m., the selling pressure had pushed the DJIA down about 2.5% of its value.
- At 2:32 p.m., a large fundamental trader initiated an order to sell a total of 75,000 E-Mini contracts (valued at approximately \$4.1 billion). The sell was initially absorbed by HFTs and fundamental buyers. Usually, such big sell order may take more than 5 hours to execute. However, on May 6, it was executed extremely fast in only 20 minutes.
- Between 2:41 p.m. and 2:44 p.m., HFTs and other traders drove the price of the E-Mini down by more than 5%.
- At 2:45:28 p.m., trading on the E-Mini was paused for five seconds when the Chicago Mercantile Exchange (CME), CME Circuit Breakers (CB) was triggered in order to prevent a cascade of further price declines. Yet NYSE did not halt trading [Sub13].
- At 2:45:33 p.m., prices stabilized when trading resumed, and the E-Mini began to recover.

In summary, between 2:40 p.m. and 3:00 p.m., approximately 2 billion shares were traded with a total volume exceeding \$56 billion. Over 98% of all shares were executed at prices within 10% of their 2:40 p.m. value.

7.1.2 Main reasons of the Flash Crash

The Flash Crash has raised many questions concerning the efficiency of the stock market system. A deep analysis of the Flash Crash shows that several reasons that contributed or led to the failure can be avoided if the IQ requirements of the system were captured properly during the early phases of the system design. Several researchers investigate IQ and their influence on the overall performance of the stock market (e.g., [FJK⁺11;

²Nanex is a firm that offers streaming data on all market transactions and distributes the data in real time to clients and allows them to do analysis and visualization in real time

FJK⁺11]), and other researchers propose different theories to explain what happened, including:

- Fat-finger trade that is a human error caused by pressing a wrong key when using a computer to input data. However, this theory was quickly disproved after it was determined that the E-mini S&P 500 order, which was under suspicion of triggering the Flash Crash, was not a result of a fat-finger trade [EDPO11; EDPO11];
- The highly fragmented nature of the market along with the inefficient coordination mechanisms among the CBs of the trading markets also played a role in the Flash Crash [GHLZ12; Sub13]. More specifically, if the trading markets do not coordinate their CBs, HFTs will simply search for a market other than the closed ones and continue trading [Mad11].
- The suspicious behavior of HFTs that effects the market prices and contributed to the Flash Crash[SC⁺10; Bow10; KKST11].
- Fraud information (e.g., falsified, inaccurate, etc.) that have been used by some actors and compromised the overall system performance. For instance, HFTs' flickering quotes that last very short time, which make them unavailable for most of traders [MU12], and Market Makers' stub quotes that are orders with prices far away from the current market prices, such orders can also be considered as falsified information; since they are orders were not intended to be performed [KKST11].
- Undetected vulnerabilities in the socio-technical system structure that have been exploit by some actors and led to a failure in the overall system [Cli11; SCC⁺12].

7.1.3 Applying the framework to the Flash Crash case study

In this section, we apply our framework to the Flash Crash case study. In particular, we use the modeling language to produce the different requirements diagrams (e.g., actors, goals, information, etc.), and then we rely of the automated analysis support to verify the correctness and consistency of the requirements model. Finally, we derive the final IQ specifications (in terms of IQ policies) from the requirements model.

7.1.4 Requirements Modeling

This section presents the modeling activities that our framework proposes, i.e., we use the modeling concepts and constructs proposed in Section 3 to model the Flash Crash scenario. The modeling language proposed in this thesis has been improved through three main stages. In each of these stages, we tested the adequacy of the proposed concepts for

modeling the case study, identify their limitations in capturing some important aspects of the system, and then overcome the identified limitations by introducing more refined and expressive concepts.

For example, in [GG15d], we proposed concepts for capturing four IQ dimensions, namely: accuracy, completeness, timeliness and consistency. Although the proposed concepts were able to capture the IQ aspects we considered, but they seem to be inadequate to capture several aspects of the system. Moreover, some of them were at high abstraction level, and we cannot rely on them to derive a clearly defined IQ specification. To tackle this problem, first we extended the IQ dimensions we consider to seven dimensions [GG15a], including: accessibility, accuracy, believability, trustworthiness, completeness, timeliness and consistency.

Second, we proposed new concepts, and we extended the previously introduced ones with attributes that enable for accommodating the new extensions. For instance, we exchanged the *trusted provision* concept that helps in analyzing the accuracy of the transferred information with several finer concepts that enable for analyzing the quality of the transferred information. More specifically, we extended the relations between goal and information by adding a new one (modify), and we enriched the modeling language by supporting the notion of permissions, which is very important for identifying the permitted/ forbidden actors' activities toward information, and in turn, for addressing several IQ related issues. Thus, we refine the modeling language by proposing four different types of permissions concerning the four types of information usage (e.g., (P)roduce, (R)ead, (M)odify and (S)end).

Moreover, we extend the language to model permission delegation among actors, and to model trust/ distrust concerning the delegated permissions. Furthermore, we refine information provision (transfer) concept by proposing two different types of provisions, namely: normal provision (P), and Integrity Preserving provision (IP provision) which can preserve the integrity of the provided (transferred) information [GG13a]. Still, the framework hardly recognizes the obligation concept, which is essential for addressing several IQ related vulnerabilities. Thus, in [GG15c] we proposed several techniques to deal with this issue.

Following our methodology, we start by actor modeling, i.e., actors of the system are identified and modeled along with their objectives, entitlements and capabilities. Second, we proceed to goal modeling activity, in which actors' goals are refined through and/or-decompositions if needed. Third, information modeling activity, in which we identify and model information, their owner(s) (if any), their relations with goals, and finally their internal structure if they are composed. Fourth, we model the social interaction among actors concerning information provision, permissions and goals delegations. Fifth,

we identify and model threat concerning goals and information, and we model actors' competencies and motivations toward such threats. Finally, we identify and model actors' social interactions and dependencies, and then we model trust among them. Each of the previously mentioned activities is supported with the ST-IQ Tool, at the end of each activity we discuss how it was performed with the help of the Tool.

Actor modeling

The modeling activities start by modeling the actors of the system along with their objectives, entitlements and capabilities. In what follows, we list and define the main actors (agents and roles) of the system along with their specialization and instantiation relations.

A stock investor: is an individual or company, who trades securities in stock market with a main aim of making a potential profit.

A stock trader: is a person or company involved in trading securities in the stock market, and able to employ different strategies for making profit out of the trading activities. In particular, he/she is able to analyze the targeted securities in the stock market, and make proper trading offers accordingly.

Based on [KKST11; SC⁺10], we can classify stock traders under five main categories based on their: (1) trading speed³; (2) position limit⁴; and (3) the percentage of the market volume. These categories are listed and defined as follows:

Fundamental Traders: are those who either buying or selling significant number of securities in one direction with a low frequency rate. Fundamental Traders can be further classified under **Fundamental Buyers** and **Fundamental Sellers** depending on the direction of their trades.

Market Makers: are firms that hold a certain number of shares of a particular security in order to facilitate trading in that security. They take the position at both sides of the market by taking long and short positions on a particular security.

High-frequency Traders: are traders take long and short positions at both sides of the market, and trade with very high frequency [Ald13]. Usually, they use some sort of algorithmic trading to rapidly trade securities, i.e., they rely on advanced computer systems to record high earnings in the market.

³Average amount of time taken between order placements

⁴Number of orders allowed to be held

Small /Noise Traders: are traders who take either a long or short positions on the market, and trade with a very low frequency.

Opportunistic Traders: traders who do not fall in the other categories.

According to [PHT⁺12], the numbers of market participation for the E-mini S&P 500 securities⁵ during the Flash Crash day were: 1268 fundamental buyers; 1276 fundamental sellers; 176 market makers; 16 HFTs; 6880 small traders; 5808 opportunistic traders, and the number of investor is much greater than that.

Stock markets (trading venues): are places where traders gather and trade securities [Har02]. Markets may be a physical trading floor (e.g., The New York Stock Exchange (NYSE), the Chicago Mercantile Exchange (CME)), or it may be an electronic system (e.g., NASDAQ) [Har02; OY11].

Markets have a main goal of “making profit by facilitating security trading”. Usually, they “manage traders’ order matching”, and they should “ensure stable trading environment”, which can be done by depending on their Circuit Breakers (CBs), where a CB is a technique that is used to slow down or halt trading activities to prevent a potential market crash. Moreover, they share all information concerning the quotes and trades they receive/ perform accordingly.

Accounting firms: are specialized for performing accounting activities, i.e., they provide companies with clear and reliable information about their economic activities and the status of their assets.

Auditing firms: are specialized for providing efficient monitoring of the quality of information produced by companies concerning their financial statements.

Consulting firms: are specialized for providing professional advices for a fee about securities to *traders* and *investors*.

Credit assessment ratings firms: are specialized for assessing of the credit worthiness of companies’ securities, i.e., such firms help *traders* in deciding how risky it is to invest money in a certain security.

Consolidate Tape Association (CTA): produces information that describes the last orders (trades) information, such information is very important for analyzing the market and make trading decisions [HJ11].

⁵The security that was under suspicion of triggering the crash

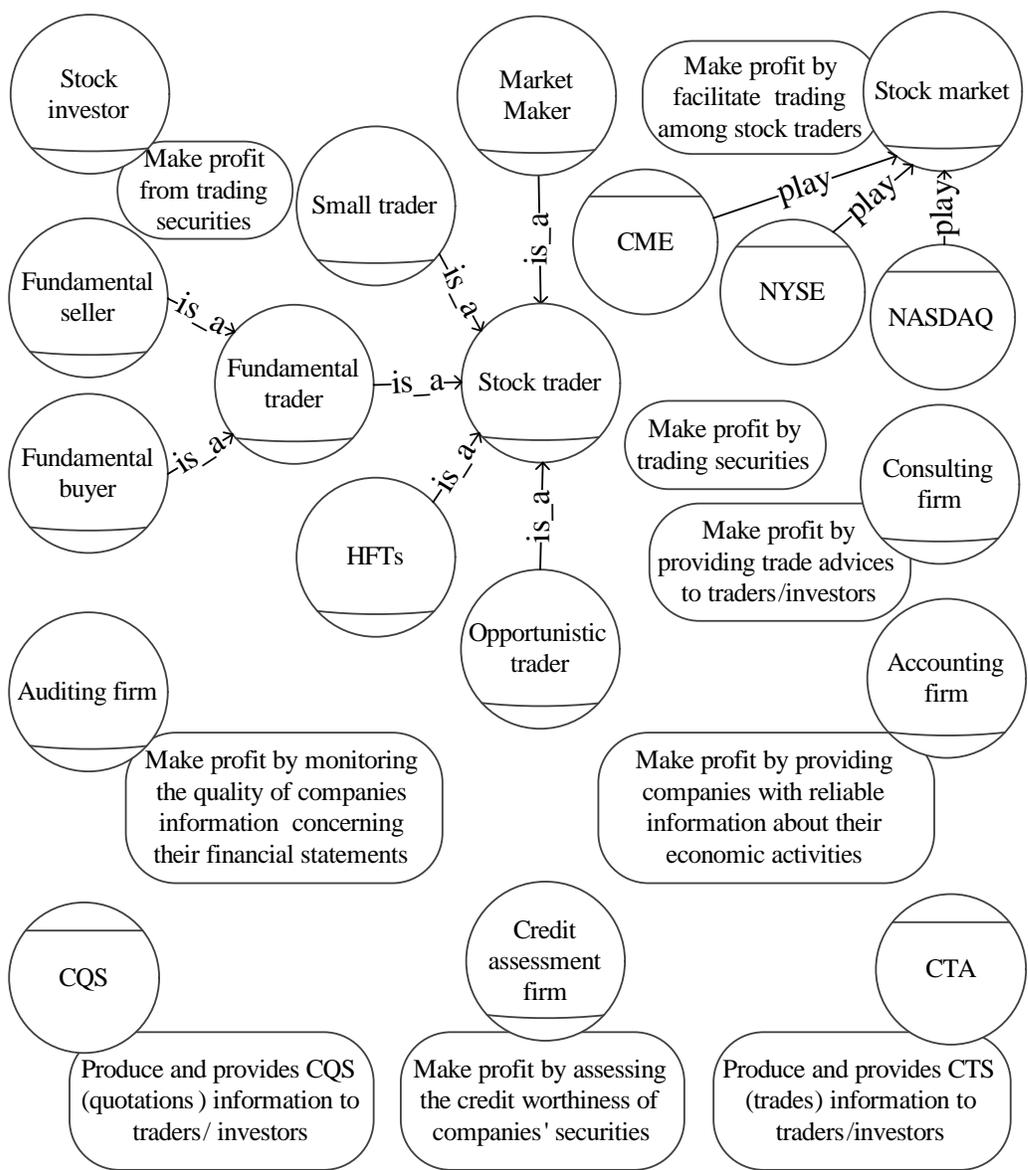


Figure 7.1: Actor Diagram

Consolidated Quotation System (CQS): produces information concerning quotation⁶ information, and provides it to different actors who require such information.

Figure 7.1 shows the actor diagram that resulted from the actor modeling activity. In which, we can identify *Stock market* that has a main goal of “make profit by facilitate trading among stock traders”. A *Fundamental seller* that is specialized from *Fundamental trader* role, which is also specialized from *Stock trader* role that has a main goal of “make

⁶A quote is an order that has not been performed

profit by trading securities”.

ST-IQ Tool support: actor modeling activity starts by modeling the main roles in the system (e.g., stock trader, stock investor, CQS), and then we model specialization relation among the roles (e.g., fundamental seller role is specialized from fundamental trader role that is also specialized from stock trader role). After identifying the roles along with their specialization relations (if any), we model the agents of the system, and we define the role(s) that such agents are instantiated of (e.g., CME is an agent that is instantiated from the stock market role). Note that some agents are not instantiated from any role (e.g., CQS, CTS). After all the roles and agents of the system are modeled along with their interrelations, their top-level goals are identified and modeled as well.

Goal modeling

Goal modeling activity aims to produce the goal diagram that is a refined model of actors’ top-level goals in terms of leaf goals, which can be achieved or delegated to other actors. In particular, goals are refined through and/or-decomposition until reaching their leaf goals, which cannot be refined anymore, and they can be achieved by actors. Figure 7.2) shows a partial goal diagram concerning the refinement of the *Stock trader’s* goal “make profit by trading securities”, which is and-decomposed into two goals “Producing the right orders” and “Decide the right trading order”. The first goal is also and-decomposed into two goals “Trade investors’ securities” and “Trading its own securities”. While the last goal is or-decomposed into two goals “Depend on consulting firm suggestions” and “Depend on trader suggestions”.

ST-IQ Tool support: the tool provides the main concepts for modeling and refining the actors’ goals, i.e., in the pallet we can find goal along with and/or-decompositions constructs.

Information modeling

Information modeling and it is the first activity toward capturing IQ requirements, and it aims to produce the information diagram that identifies the different relation between information and other modeling constructs in the system. In particular, information modeling can be composed of three sub modeling activities:

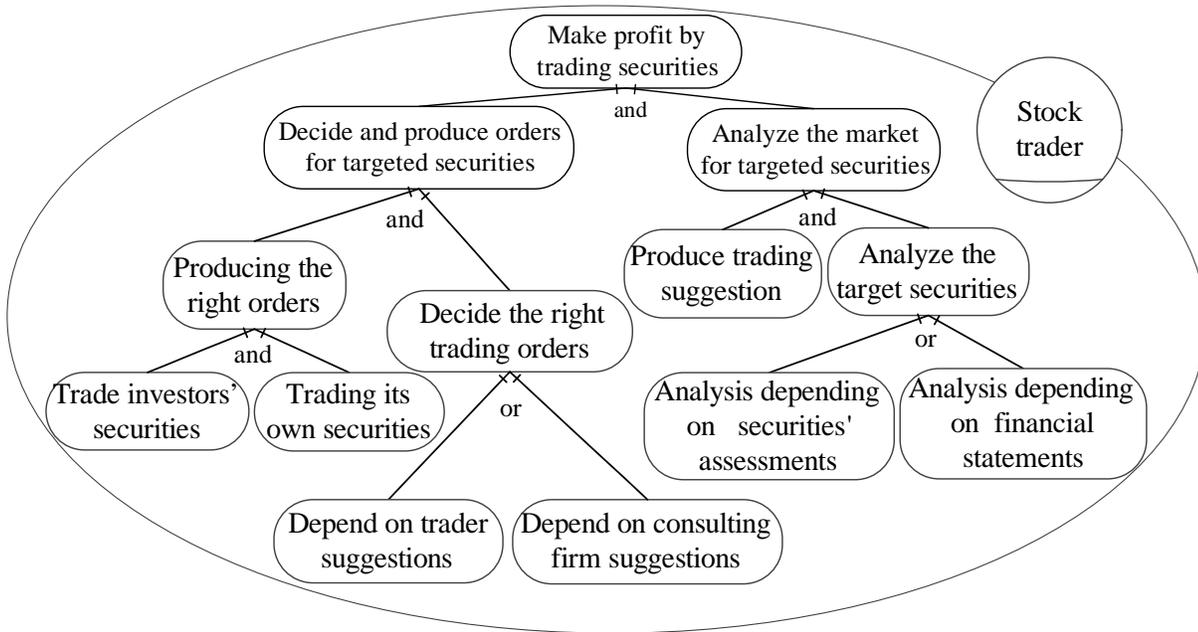


Figure 7.2: Partial Goal Diagram

- (1) **Information ownership modeling:** in which we model the legal owner(s) of information items, which is essential to capture their requirements concerning information they own. In addition, it identifies who has full control over information use (e.g., information permissions/ permissions delegation);

Example 37. *a stock investor is the [O]wner of investor 's order, and CME is the [O]wner of CME CB info*

- (2) **Goal-information relations modeling:** in which we model the different relations between goals and information, where these goals may produce, read, modify, and/or send information. Moreover, we define the different attributes of the previously mentioned relations base on the stockholders' needs.

Example 38. *a stock investor is [P]roduces and [S]ends investor 's order through its goal "Receive orders from traders", and it can define several attributes related to produce and send relations.*

- (3) **Information structure modeling:** in which we identify the internal structure of composite information, which enables to decide whether such information is complete or not.

Example 39. *CME CB info is considered as a sub item of both (part of) NYSE CB info and NYSE CB info.*

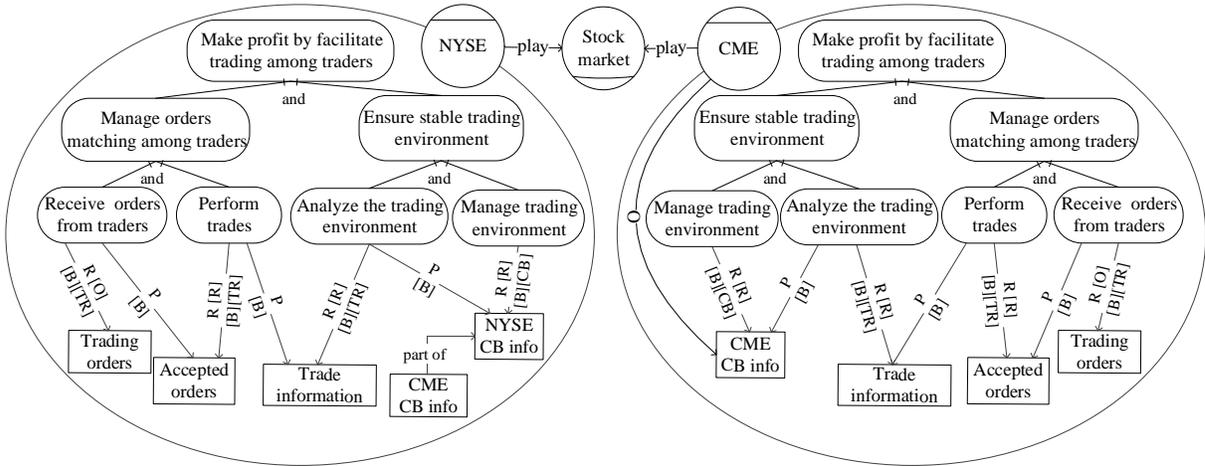


Figure 7.3: Partial Information Diagram

ST-IQ Tool support: the tool provides concepts for modeling the ownership relation between an actor with information it owns, and it provide the different relation between goals and information they produce, read, modify and send, and allows to define the different attributes related to these relations. Finally, it allows for modeling the relation between an information item and its sub items by part of relation.

Figure 7.3, shows a partial information diagram concerning stock markets. In which, we can identify, the goal “Receive orders from traders” ([o]ptional) reads *Trading orders* for the purpose of use [Tr]adding, and it produces *Accepted orders*. While the goal “Manage trading environment” ([R]equired) reads *NYSE CB info* for the purpose of use [CB]reaker. In addition, *CME* is the [O]wner of *CME CB info*, where *CME CB info* is considered as a *part of* *NYSE CB info*. Thus, *CME CB info* should be provided to *NYSE*.

Social interaction modeling

Social interaction modeling aims to produce the social dependency diagram that captures the actor interactions and dependencies, including information provision, objectives and authorities’ delegation. In particular, goals are delegated from actors who do not have the capability to achieve to the actors, who have such capabilities. In addition, information /authorities are provided /delegated from actors who have the capabilities to provide/delegate them to the actors who require them to achieve their objectives.

Figure 7.4, shows a partial social dependency diagram. In which, we can identify that *CQS* receives *Quotations info-CQS* from *Stock market*, and it provides *CQS-info* to *stock trader*. *Stock investor* provides *investor’s orders* to the *stock trader*, and it delegates read,

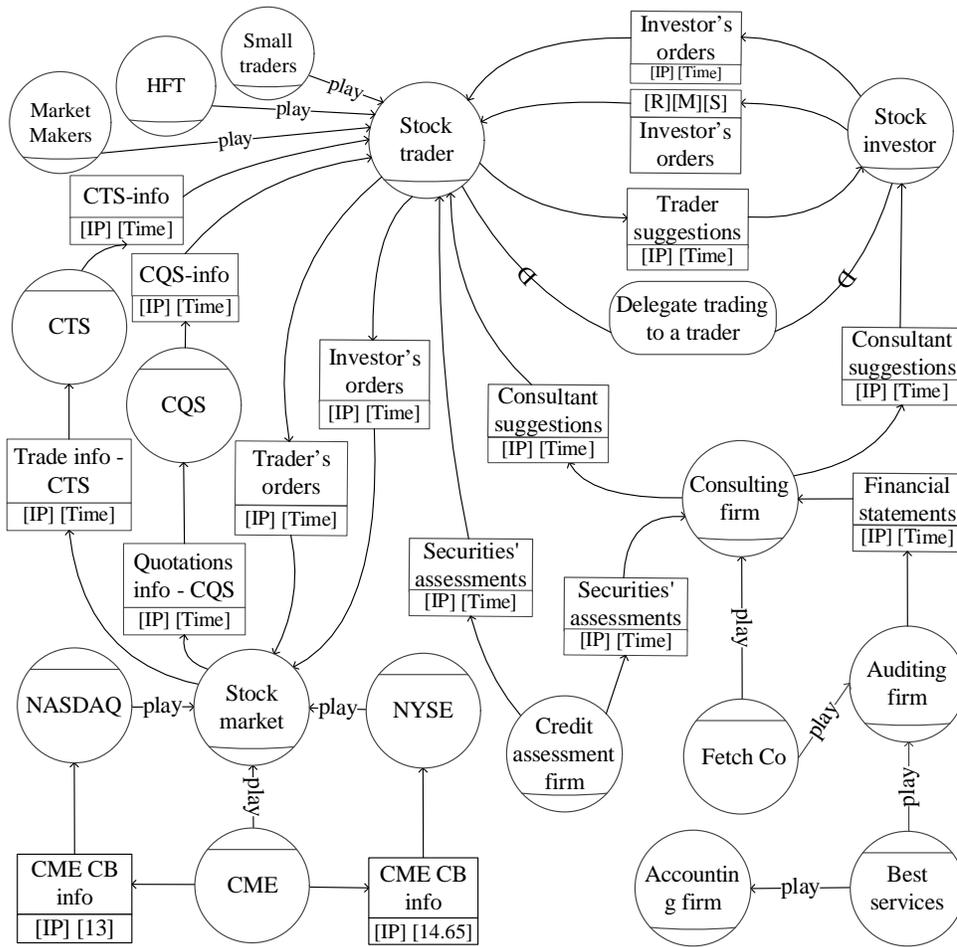


Figure 7.4: Partial Social Interaction Diagram

modify and send permissions to it. In addition, it delegates the “ delegate trading to a trader”. *CME* provides *CME CB info* to both part of *NYSE* and *NASDAQ*, with two different provision times (14.65 ms, 13 ms).

ST-IQ Tool support: the tool provides the required concepts for modeling all the social interactions we consider, and it enables for specifying their related attributes. For example, the provision construct allows for defining the provided information, the provision source and destination. Moreover, it allows choosing the type and time of the provision.

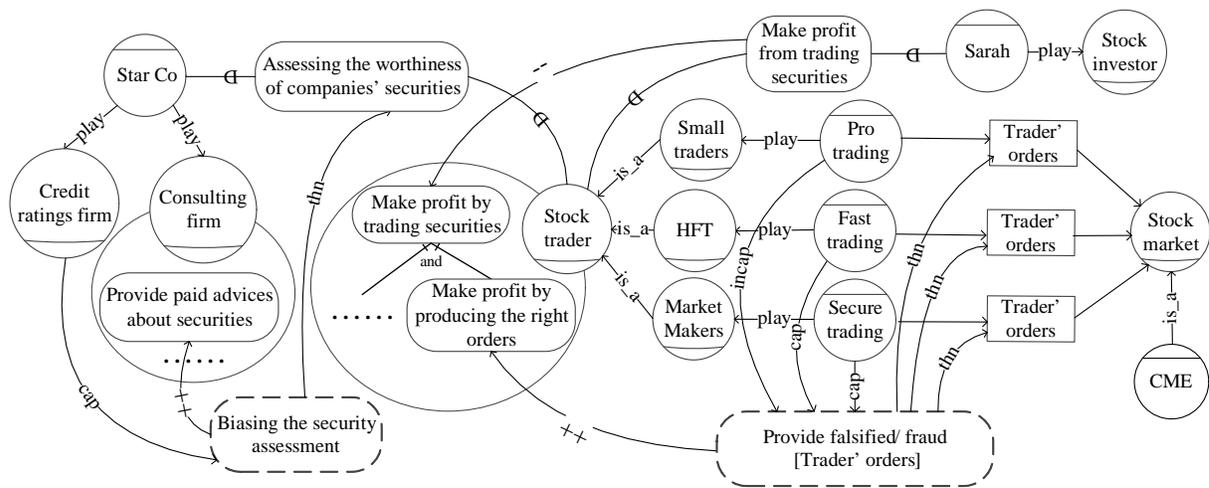


Figure 7.5: Partial Threat Diagram

Threat modeling

Threat modeling aims to produce the threat diagram that identify and models threats to goals and information, and identify actors' competencies and motivations toward such threats. Usually, threats to information/goals are defined by stockholders or domain expert, and based on the threats characteristics we can determine which actors have the capabilities to achieve them.

Figure 7.5 shows a partial threat diagram. In which, we can identify a threat “Biasing the security assessment” to the goal “Assessing the worthiness of companies’ securities”, and a threat “Provide falsified/fraud orders” to the information “trader orders”. In addition, actors’ capabilities toward such threats are identified and modeled, and threats motivations related constructs (e.g., goals contribution, goal-threat threats) are identified and modeled.

ST-IQ Tool support: propose constructs for modeling threats, and their relations to both goals and information (threaten). Moreover, it enables for modeling the capability/incapability relations between actors and such threats. In addition, it introduces positive and negative contribution relations between goals, and between threat and goals.

Trust modeling

Trust modeling aims to produce the trust diagram that identify and models trust/ distrust relations among actors of the system concerning information producing, goal and permissions delegations. Figure 7.6, shows a partial trust diagram concerning our case

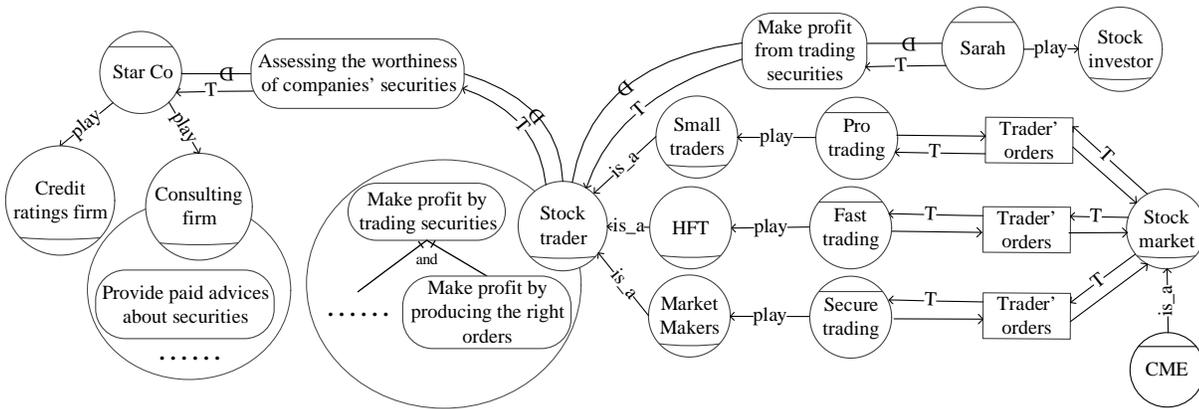


Figure 7.6: Partial Trust Diagram

study. In which, we can identify a trust relations concerning “trader’s orders” between the stock market at one hand, and Secure trading, Fast trading, and Pro trading at the other hand. Moreover, we can identify a trust relation between a stock trader and Star Co concerning the delegated goal “Assessing the worthiness of companies’ securities”.

ST-IQ Tool support: introduce three types of constructs for modeling trust concerning produced information, delegated permissions and goals, in which we identify the trustor, trustee, and the trustum (information, permissions, goal). In addition, we can define the level of trust (trust/distrust) that the trustor put in the trustee.

7.1.5 Analyzing the requirements model

Aims to verify the correctness and consistency of the requirements model diagrams that resulted from the previously mentioned modeling activities. As previously mentioned, graphical models cannot support any kind of automated analysis without a formal representation of their semantics. Thus, we translate graphical models into Disjunctive Datalog [BFI⁺09] formal specifications that enables for performing the required analysis to verify the correctness and consistency of the requirements model. The translation of these models produces the extensional predicates, which represent all the modeling constructs along with their semantic relations. In addition, the extensional predicates are applied to a set of reasoning rules (axioms) to infer knew knowledge from the model that is the intensional predicates. Finally, we use all the knowledge we have about to model to check its correctness and consistency. In order to do that, we have defined a set of properties of the design; these properties define constraints that the designers should consider during

the system design to guarantee the correctness and consistency of the requirements model.

For example, in the trust diagram (shown in Figure 7.6) a trust relation should hold concerning “trader’s orders” between the stock market at one hand, and Secure trading, and Fast trading at the other hand. However, according to the threat diagram (shown in Figure 7.5), such relation cannot hold, since both Secure trading, and Fast trading have the capabilities and motivations to achieve the related threat (“Provide falsified/fraud orders”).

ST-IQ Tool support: the tool enables for translating the different requirements model diagrams into Disjunctive Datalog formal specifications, and then it reads the properties of the design from a file, and perform the required analysis. In case there is a violation to the design properties, it notifies the designer about it, which allows him to resolve such violation by modifying the requirements model.

We run the automated analyses supported by our framework over the created models, and the analysis captured several violations of the properties of the design, including:

Inaccurate information: markets (e.g., *CME*) consider information received from any trader that plays the role of *HFT trader* as inaccurate information, since no trust in information production holds between them at one hand, and markets at the other.

Inaccurate information due to playing conflicting roles: *Best services* is playing two conflicting roles “Accounting firm” and “Auditing firm” concern producing “Financial statement”, it provides accounting and auditing services for the same company. However, we cannot trust a company for providing accurate auditing information (“Financial statement”) for a company that it gets paid from to perform their accounting services. Thus, *Best services* should not has produce permissions concerning “Financial statement” information, and in turn, if it produce such information it is considered as inaccurate (no produce permissions).

Unauthorized read due to playing conflicting roles: *Fetch Co* is playing two conflicting roles “Auditing firm” and “Consulting firm” concern reading “Security assessment”, since it might use such information for providing paid consulting services. Thus, *Fetch Co* should not has read permissions concerning “Security assessment” information.

Incomplete information: “NYSE CB info” and “Nasdaq CB info” are identified as incomplete information from the perspectives of their readers, since they miss some sub parts related to the purpose of use. However, this can be solved by providing

these markets with the missed information sub items, i.e., providing both *NYSE* and *NASDAQ* with “CME CB info”.

Inconsistent information: both of *NYSE* and *NASDAQ* are *interdependent readers* concerning “CME CB info”. However, *CME CB info* is provided to them with two different provision times. According to [Lew14], provision time from *CME* to *NASDAQ* was 13 (ms), while provision time from *CME* to *NYSE* was 14.65 (ms), which leads to different *read times* between them, which results in inconsistency between them.

The proposed properties of the design enable to capture all the violations that we consider in this thesis.

7.1.6 Deriving the final IQ Specifications

As discussed in Chapter 5, after verifying that the requirements model is correct and consistent, we can derive correct and consistent IQ specifications from the requirements model. In particular, we rely on the IQ policy specifications derivation rules to automatically derive the final IQ policies. These rules enable for deriving three types of IQ policies (permit, forbid, and obligate) that are used to control four different types of activities over information, namely, produce, read, modify, and send. More specifically, *permit policies* are used to define the activities that an actor is allowed to perform over information, *forbid policies* are used to define the activities that an actor is prohibited to perform over information, and *obligate policies* that are used to specify the activities that an actor must perform over information. In addition, the derivation rules represent the derived policies in the IQ policy specifications language. In what follows, we list some of the derived IQ policies:

```
permitted_produce(investor, investors_order)
permitted_read(trader, investors_order, for, make_profit_by_producing_orders)
permitted_modify(trader, investors_order, for, make_profit_by_producing_orders)
permitted_send(trader, investors_order, to, stock_market)

forbidden_read(trader, investors_order, for, analyzing_the_market)
forbidden_send(trader, investors_order, to, consulting_firm)

obligated_read(NYSE, CME_CB_info, for, manage_trading_environment, in, 0)
obligated_read(NASDAQ, CME_CB_info, for, manage_trading_environment, in, 0)
obligated_provide(stock_market, CTA, CQS-info, in, 10)
obligated_provide(stock_market, CTA, CTS-info, in, 10)
```

ST-IQ Tool support: the tool enables for deriving the final IQ specifications and represent them in the IQ policy specification language that provide a clear way for specifying IQ policies in terms of the permitted, forbidden and obligated activities toward information.

7.2 Scalability experiments

In this section, we report the result of our scalability experiments to demonstrate how well the automated reasoning techniques perform while applied to large requirements models, i.e., we investigate the relation between the size of the model and the reasoning execution time.

7.2.1 Design of the study

We start by selecting a base model, and then we clone the model to obtain larger models. We apply the reasoning techniques to each of these models, and we calculate the reasoning execution time for each of them. In particular, we select the model shown in Figure 3.7, as a base model, and we increasing the number of its modeling elements from 142 to 9340 through six steps, i.e., and investigate the reasoning execution time at each step by repeating the reasoning execution seven times, discarding the fastest and slowest execution times, and then computed the average execution times of the rest.

7.2.2 Experiment results

We have performed the experiment on laptop computer, Intel(R) core(TM) i3- 3227U CPU@ 190 GHz, 4GB RAM, OS Window 8, 64-bit. The result is shown in Figure 7.7, and it is easy to note that the relation between the size of the model (the number of its nodes) and the execution time is not exponential, i.e., the reasoning techniques should work fine with real world scenarios, where there sizes probably will not exceed the sizes we considered.

7.3 Chapter summary

In this chapter, we briefly described the stock market system structure, where the Crash occurs in 7.1, the Flash Crash chronology of events 7.1.1, and then we discussed the main reasons of the Flash Crash 7.1.2. In 7.1.3, we described the applicability of our framework by applying it to the Flash Crash case study. The evaluation considers both the modeling language 7.1.4 along with the analysis techniques for verifying the correctness

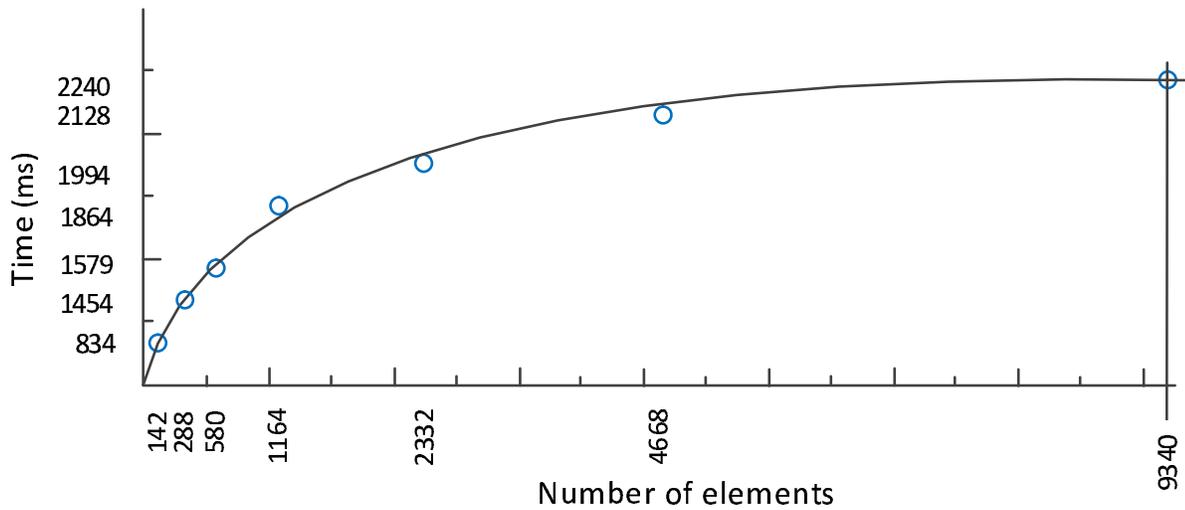


Figure 7.7: Scalability results with increasing the number of modeling elements

and consistency of the IQ requirements model 7.1.5, and we describe the automated derivation of the final operational specifications of IQ requirements in 7.1.6. Finally, we performed the scalability experiments to demonstrate the efficiency of the automated reasoning techniques proposed by the framework in 7.2. In particular, in the modeling concepts and constructs that our language propose seem to be adequate for capturing all IQ related aspects that we consider in this thesis. In addition, the proposed properties of the design along with the supporting reasoning techniques were able to identify all the violations that we consider in this thesis, and the scalability experiments we performed demonstrate the ability of the reasoning techniques to deal with large models.

Chapter 8

Conclusions and future work

I think and think for months and years. Ninety-nine times, the conclusion.

Albert Einstein

8.1 Summary of the Thesis

Usually, IQ is a problem handled at technical level, however, IQ concerns also how information are managed by people in the organization and how business processes make the correct use of information during their execution. In this thesis, we argued that IQ is not only a technical problem, but it is also a social and organizational issue. Thus, any solution to IQ should consider the social and organizational context where the system will eventually operates. Furthermore, we advocate that such needs should be considered from the early phases of the system development, which may prevent revising the entire system to accommodate such needs after the system deployment, which might be very costly. This thesis addresses this issue by proposing a RE framework that supports system designers in modeling and analyzing IQ requirements in their social and organizational context since the early phases of the system development. We summarize the contributions of this thesis, as follows:

A modeling language: we proposed a modeling language that has been developed based on a set of objectives, and several principles proposed by Paige et al [POB00]. The language adopts concepts from both Secure Tropos [MG07] and SI* [Zan06] modeling languages, and extends them with concepts and constructs for modeling IQ requirements of the system-to-be. In particular, our language offers concepts and constructs for modeling actors of the system along with their objectives, entitlements, authorities, and it proposes more refined concepts for trust analysis based on

the internal structure of the actors. In addition, it proposes concepts and constructs to model IQ requirements in terms of seven IQ dimensions, namely: accessibility, believability, trustworthiness, accuracy, completeness, timeliness (validity), and consistency.

A formal framework: that underlies our modeling language, and supports system designers while performing the required analysis to verify the correctness and consistency of the IQ requirements model. In particular, we formalized all constructs that our modeling language offer in disjunctive Datalog, and we proposed several reasoning rules (axioms) that enable for deriving new knowledge about the requirements model. Moreover, we proposed a set of properties of the design that can be used to verify the IQ requirements model. In other words, these properties are able to detect violations to the system design, and notify the designer to resolve them, either by modifying the requirements model, or by relaxing some requirements.

A mechanism for the automated derivation of IQ specifications: that allows for the derivation of IQ specifications from the IQ requirements model, and represents them clearly defined IQ specification language that is able to define the permitted, forbidden, and obligated activities toward information. One main advantage of this mechanism is its ability to specify correct and consistent IQ policies, since it derives these policies from a verified requirements model of the system, which guarantee that the derived policies are correct and consistent with one another, and with the system requirements. Moreover, it offers a great flexibility in determining the right implementation of the defined policies, if there are several alternatives, since it only specify the required IQ policies.

An engineering methodology: that proposes a detailed process to be followed by system designers during the different phases of modeling and analyzing IQ requirements of the system-to-be, and then deriving the final operational specifications from a verified IQ requirements model. In particular, the process has three main phases, namely: modeling, analyzing, and specification phases. The process is iterative, i.e., it can be repeated, if required, to refine the model until reaching the target design of the system-to-be.

A CASE tool (ST-IQ Tool): consists of four main components (i) a control component (JAVA based-program) that controls and coordinates the activities of the three other components; (ii) modeling component that allows designers for drawing the requirements models by drag-and-drop modeling elements from palettes; (iii) Model-to-text transformation component that supports the translating of the graphical re-

quirements models into Disjunctive Datalog formal specifications depending; (iv) an automated reasoning component that allows for performing the required analysis to verify the correctness and consistency of the requirements model against some properties of the design. ST-IQ tool aims to support system designers during all the system design activities (e.g., modeling, analysis, specifications activities).

8.2 Limitations of the framework

The following list provides a summary of limitations that we have identified in our framework:

End users' evaluation: although we empirically evaluated the utility, applicability, and the reasoning techniques scalability of our framework. Yet we did not evaluate it with end users (designers), i.e., we did not perform experiments to assess the adequacy of our proposed framework with end users.

Evaluation with other case studies: we evaluated our framework by applying it to only one case study concerning a stock market domain. However, this can be considered as a limitation to the applicability of the framework, i.e., several concerns can be raised whether the framework can be applied to systems that belong to different domains. Thus, we are planning to apply the framework to several complex case studies that belong to other domains.

ST-IQ Tool installation: the installation of the Tool is not user friendly, and it requires several applications at the host environment (e.g., Java, Sirius, Acceleo) to be installed, and run appropriately, since it has been developed by integrating several existing technologies.

8.3 Ongoing and Future Work

In what follows, we discuss several ongoing and future works to improve our framework.

Enhancing IQ analysis: we are planning to extend the IQ dimensions that we considered in this thesis, and to better investigate how each IQ dimensions can be analyzed. In addition, we believe that the interrelations among these dimensions need to be studied more deeply. Moreover, information production needs more investigation, since information might be produced depending on other information item(s), and the quality of the produced information might be influenced by the quality of the information item(s) that has/have been used in the production process.

Better evaluating the framework: user-oriented evaluation is surely one mandatory step in the evaluation of any RE framework. We are currently working on that, and we are confident that we will have interesting results to show very soon. In addition, we aim to apply the framework to several complex case studies that belong to other domains.

Improving ST-IQ Tool: although ST-IQ Tool is able to provide the basic requirements for modeling and analyzing IQ requirements of the system, yet it can be improved by several means, including: (i) we are planning to implement mechanisms that automatically support the Model-2-text mapping; (ii) improve its usability by proposing an integrated installation of the tool, i.e., users do not need to install several applications that are essential for installing and using the Tool; (iii) enhance the modeling component by adopting the multi-view modeling.

Improving the automated specification of IQ of policies: we plan to refine our proposed IQ policy specification language, and extend it with events that can be used to trigger these policies, and we are planning to make the language more expressive by adding different kinds of constraints to it. Moreover, we aim to propose more expressive derivations rules, and investigating if other kind of policies needs to be considered.

Refining the social trust analysis: the concept of trust has been studied within several research areas (e.g., philosophy, sociology, organization theory, etc.), and there is general consensus that trust is complex and multidimensional concept [CW03]. In addition, several researchers tried to answer the challenging question “how trust can be built?”. For instance, many researchers focused propose approaches for building trust based on some related beliefs, including: competence, disposition, fulfillment, dependence, willingness, and motivation beliefs [ARH00; CF01]. While other researchers introduce computational models for trust [Ste94; SFR00; EC01; ARH00], where trust can be built based on an already defined constructs (e.g., repetition, motivations, competences, etc.). To this end, we are currently extending our framework by proposing more refined concepts for modeling and analyzing trust among actors based on sets of beliefs related to the actors’ competencies and motivations.

Bibliography

- [Ack89] R.L. Ackoff. From data to wisdom. *Journal of applied systems analysis*, 16(1):3–9, 1989.
- [AER02] Annie I Antón, Julia Brande Earp, and Angela Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 23–31. IEEE, 2002.
- [AGMZ07] Y. Asnar, P. Giorgini, F. Massacci, and N. Zannone. From trust to dependability through risk analysis. 2007.
- [AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Citeseer, 1995.
- [Ald13] Irene Aldridge. *High-frequency trading: a practical guide to algorithmic strategies and trading systems*. John Wiley & Sons, 2013.
- [AN95] A. Aamodt and M. Nygard. Different roles and mutual dependencies of data, information, and knowledge—an ai perspective on their integration. *Data & Knowledge Engineering*, 16(3):191–222, 1995.
- [ARH00] Alfarez Abdul-Rahman and Stephen Hales. Supporting trust in virtual communities. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 9–pp. IEEE, 2000.
- [AS04] Ruth Sara Aguilar-Saven. Business process modelling: Review and framework. *International Journal of production economics*, 90(2):129–149, 2004.
- [Bai86] A. Baier. Trust and antitrust. *Ethics*, 96(2):231–260, 1986.
- [Bak85] M. Baker. The mirror principle and morphosyntactic explanation. *Linguistic inquiry*, 16(3):373–415, 1985.
- [BCM⁺94] Alan W Brown, David J Carney, Edwin J Morris, Dennis B Smith, et al. *Principles of CASE tool integration*. Oxford University Press, 1994.
- [BCM04] G. Bellinger, D. Castro, and A. Mills. Data, information, knowledge, and wisdom. URL: <http://www.systems-thinking.org/dikw/dikw.htm>, 2004.
- [BDL06] D. Basin, J. Doser, and T. Lodderstedt. Model driven security: From uml models to access control infrastructures. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1):39–91, 2006.
- [BFI⁺09] Robert Bihlmeyer, Wolfgang Faber, Giuseppe Ielpa, Vincenzino Lio, and Gerald Pfeifer. Dlv-user manual. *The DLV Project*, 2009.

- [Bie92] E.W. Biersack. Performance evaluation of forward error correction in atm networks. In *ACM SIGCOMM Computer Communication Review*, volume 22, pages 248–257. ACM, 1992.
- [BKWC01] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In *Database Theory UICDT 2001*, pages 316–330. Springer, 2001.
- [Blo97] K. Blomqvist. The many faces of trust. *Scandinavian journal of management*, 13(3):271–286, 1997.
- [BN96] P. Bernus and L. Nemes. A framework to define a generic enterprise reference architecture and methodology. *Computer Integrated Manufacturing Systems*, 9(3):179–191, 1996.
- [Bom08] S. Bommena. Cyclic redundancy check (crc). 2008.
- [Bow10] Graham Bowley. Lone \$4.1 billion sale led to \$flash crash\$in may. *The New York Times*, 1, 2010.
- [BP85] D.P. Ballou and H.L. Pazer. Modeling data and process quality in multi-input, multi-output information systems. *Management science*, 31(2):150–162, 1985.
- [BP03] Donald P. Ballou and Harold L. Pazer. Modeling completeness versus consistency tradeoffs in information decision contexts. *Knowledge and Data Engineering, IEEE Transactions on*, 15(1):240–243, 2003.
- [BPG⁺01] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. A knowledge level software engineering methodology for agent oriented programming. In *Proceedings of the fifth international conference on Autonomous agents*, pages 648–655. ACM, 2001.
- [BPG⁺04] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
- [Bri95] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565, 1995.
- [BS00] K. Blomqvist and P. Stahle. Building organizational trust. In *Proceedings of 16th Annual IMP Conference*. Citeseer, 2000.
- [BS06] C. Batini and M. Scannapieco. *Data quality: concepts, methodologies and techniques*. Springer, 2006.
- [BSM03] Matthew Bovee, Rajendra P Srivastava, and Brenda Mak. A conceptual framework and belief-function approach to assessing overall information quality. *International journal of intelligent systems*, 18(1):51–74, 2003.
- [Buc91] M.K. Buckland. Information as thing. *Journal of the American Society for information science*, 42(5):351–360, 1991.
- [BWHW05] C. Braun, F. Wortmann, M. Hafner, and R. Winter. Method construction-a core approach to organizational engineering. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1295–1299. ACM, 2005.
- [BWPT98] Donald Ballou, Richard Wang, Harold Pazer, and Giri Kumar Tayi. Modeling information manufacturing systems to determine information product quality. *Management Science*, 44(4):462–484, 1998.

- [CCG⁺02] G. Caire, W. Coulier, F. Garijo, J. Gomez, J. Pavón, F. Leal, P. Chainho, P. Kearney, J. Stark, R. Evans, et al. Agent oriented analysis using message/uml. *Agent-oriented software engineering II*, pages 119–135, 2002.
- [CCRC13] Cinzia Cappiello, Angelica Caro, Alfonso Rodriguez, and Ismael Caballero. An approach to design business processes addressing data quality issues. In *ECIS*, page 216, 2013.
- [CdPL09] L. Chung and J. do Prado Leite. On non-functional requirements in software engineering. *Conceptual modeling: Foundations and applications*, pages 363–379, 2009.
- [CEH⁺09] M. Chen, D. Ebert, H. Hagen, R.S. Laramée, R. Van Liere, K.L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver. Data, information, and knowledge in visualization. *Computer Graphics and Applications, IEEE*, 29(1):12–19, 2009.
- [CF98] C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 72–79. IEEE, 1998.
- [CF01] C. Castelfranchi and R. Falcone. Social trust: A cognitive approach. *Trust and deception in virtual societies*, 2005(11 July):55–90, 2001.
- [CFP03] C. Castelfranchi, R. Falcone, and G. Pezzulo. Trust in information sources as a source for trust: a fuzzy approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 89–96. ACM, 2003.
- [Cli11] Dave Cliff. The flash crash of may 6th 2010: Wtf. Technical report, mimeo, University of Bristol, 2011.
- [Coh87] F. Cohen. A cryptographic checksum for integrity protection. *Computers & Security*, 6(6):505–510, 1987.
- [CPG11] Amit K Chopra, Elda Paja, and Paolo Giorgini. Sociotechnical trust: an architectural approach. In *Conceptual Modeling–ER 2011*, pages 104–117. Springer, 2011.
- [CPS96] Phillip Cykana, Alta Paul, and Miranda Stern. Dod guidelines on data quality management. In *IQ*, pages 154–171, 1996.
- [CW03] K. Chopra and W.A. Wallace. Trust in electronic environments. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. Ieee, 2003.
- [DC97] P.M. Doney and J.P. Cannon. An examination of the nature of trust in buyer-seller relationships. *the Journal of Marketing*, pages 35–51, 1997.
- [DDL01] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *Policies for Distributed Systems and Networks*, pages 18–38. Springer, 2001.
- [DHL⁺86] E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert, and A. Rifaut. A knowledge representation language for requirements engineering. *Proceedings of the IEEE*, 74(10):1431 – 1444, oct. 1986.
- [Dic89] Oxford English Dictionary. Oxford: Oxford university press, 1989.
- [Die06] J.L.G. Dietz. *Enterprise ontology: theory and methodology*. Springer Verlag, 2006.

- [DLBK08] Chenyun Dai, Dan Lin, Elisa Bertino, and Murat Kantarcioglu. An approach to evaluate data trustworthiness based on data provenance. In *Secure Data Management*, pages 82–98. Springer, 2008.
- [DM92] William H DeLone and Ephraim R McLean. Information systems success: the quest for the dependent variable. *Information systems research*, 3(1):60–95, 1992.
- [DP01] T.H. Davenport and L. Prusak. Working knowledge: how organizations manage what they know. Boston MA, USA: Harvard Business, 2001.
- [dSDMM03] Paulo Pinheiro da Silva, Silva Deborah, Deborah L McGuinness, and Rob Mccool. Knowledge provenance infrastructure. 2003.
- [DWS01] S.A. DeLoach, M.F. Wood, and C.H. Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231–258, 2001.
- [EC01] Babak Esfandiari and Sanjay Chandrasekharan. On how agents make friends: Mechanisms for trust acquisition. In *Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, Montreal, Canada*, pages 27–34, 2001.
- [EDPO11] David Easley, MM Lopez De Prado, and Maureen OŠhara. The microstructure of the flash crash: Flow toxicity, liquidity crashes and the probability of informed trading. *Journal of Portfolio Management*, 37(2):118–128, 2011.
- [EGM97] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive datalog. *ACM Transactions on Database Systems (TODS)*, 22(3):364–418, 1997.
- [ES05] Adir Even and Ganesan Shankaranarayanan. Value-driven data quality assessment. In *IQ*, 2005.
- [ES07] Adir Even and Ganesan Shankaranarayanan. Utility-driven assessment of data quality. *ACM SIGMIS Database*, 38(2):75–93, 2007.
- [ET60] FE Emery and EL Trist. Socio-technical systems. management sciences, models and techniques. churchman cw et al, 1960.
- [FBGL98] M.S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin. An organization ontology for enterprise modelling. *Simulating organizations: Computational models of institutions and groups*, pages 131–152, 1998.
- [FH94] S. Fein and J.L. Hilton. Judging others in the shadow of suspicion. *Motivation and Emotion*, 18(2):167–198, 1994.
- [FJK⁺11] Mark Flood, HV Jagadish, Albert Kyle, Frank Olken, and Louiqa Raschid. Using data for systemic financial risk management. In *CIDR*, pages 144–147, 2011.
- [FK01] C.W. Fisher and B.R. Kingma. Criticality of data quality as exemplified in two disasters. *Information & Management*, 39(2):109–116, 2001.
- [For06] A Behrouz Forouzan. *Data Communications & Networking*. Tata McGraw-Hill Education, 2006.
- [Fow97] Martin Fowler. *Analysis patterns: reusable object models*. Addison-Wesley Professional, 1997.

- [FP04] Chiara Francalanci and Barbara Pernici. Data quality assessment from the user's perspective. In *Proceedings of the 2004 international workshop on Information quality in information systems*, pages 68–73. ACM, 2004.
- [GA93] P.W.P.J. Grefen and P.M.G. Apers. Integrity control in relational database systems—An overview. *Data & Knowledge Engineering*, 10(2):187–223, 1993.
- [Gam00] D. Gambetta. Can we trust trust. *Trust: Making and breaking cooperative relations*, pages 213–237, 2000.
- [GG13a] Mohamad Gharib and Paolo Giorgini. Analysing information integrity requirements in safety critical systems. *The 3rd International Workshop on Information Systems Security Engineering WISSEŠ13.*, 2013.
- [GG13b] Mohamad Gharib and Paolo Giorgini. Modeling and analyzing information integrity in safety critical systems. In *Advanced Information Systems Engineering Workshops*, pages 524–529. Springer, 2013.
- [GG14] Mohamad Gharib and Paolo Giorgini. Detecting conflicts in information quality requirements: the may 6, 2010 flash crash. Technical report, Università degli studi di Trento, 2014.
- [GG15a] Mohamad Gharib and Paolo Giorgini. Dealing with information quality requirements. In *Enterprise, Business-Process and Information Systems Modeling (EMMSADŠ15)*. Springer, 2015.
- [GG15b] Mohamad Gharib and Paolo Giorgini. A framework for information quality requirements engineering. 2015.
- [GG15c] Mohamad Gharib and Paolo Giorgini. A goal-based approach for automated specification of information quality policies. In *Research Challenges in Information Science (RCIS), 2015 IEEE Ninth International Conference*. IEEE, 2015.
- [GG15d] Mohamad Gharib and Paolo Giorgini. Modeling and reasoning about information quality requirements. In *Requirements Engineering: Foundation for Software Quality*, pages 49–64. Springer, 2015.
- [GG15e] Mohamad Gharib and Paolo Giorgini. Modeling and reasoning about information quality requirements in business processes. In *Enterprise, Business-Process and Information Systems Modeling (BPMDSŠ15)*. Springer, 2015.
- [GHLZ12] Peter Gomber, Martin Haferkorn, Marco Lutat, and Kai Zimmermann. The effect of single-stock circuit breakers on the quality of fragmented markets. In *FinanceCom*, pages 71–87, 2012.
- [GJKL01] Gunter Gans, Matthias Jarke, Stefanie Kethers, and Gerhard Lakemeyer. Modeling the impact of trust and distrust in agent networks. In *Proc. of AOISSŠ01*, pages 45–58, 2001.
- [Gli05] Martin Glinz. Rethinking the notion of non-functional requirements. In *Proc. Third World Congress for Software Quality*, volume 2, pages 55–64, 2005.
- [GMB82] Sol J Greenspan, John Mylopoulos, and Alex Borgida. Capturing more world knowledge in the requirements specification. In *Proceedings of the 6th international conference on Software engineering*, pages 225–234. IEEE Computer Society Press, 1982.
- [GMMZ04] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Filling the gap between requirements engineering and public key/trust management infrastructures. *Public Key Infrastructure*, pages 630–632, 2004.

- [GMMZ05] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling social and individual trust in requirements engineering methodologies. *Trust Management*, pages 161–176, 2005.
- [GMNS03] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with goal models. *Conceptual Modeling UER 2002*, pages 167–181, 2003.
- [GMP03] Fausto Giunchiglia, John Mylopoulos, and Anna Perini. The tropos software development methodology: processes, models and diagrams. In *Agent-Oriented Software Engineering III*, pages 162–173. Springer, 2003.
- [GZF04] Zahia Guessoum, Mikal Ziane, and Nora Faci. Monitoring and organizational-level adaptation of multi-agent systems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 514–521. IEEE Computer Society, 2004.
- [Ham50] R.W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.
- [Har02] Larry Harris. *Trading and exchanges: Market microstructure for practitioners*. Oxford University Press, 2002.
- [HH03] Bernd Heinrich and Markus Helfert. Analyzing data quality investments in crm-a model-based approach. 2003.
- [HJ11] Craig W Holden and Stacey Jacobsen. The breakdown of standard microstructure techniques: And what to do about it. *Available at SSRN 1911491*, 2011.
- [HKK07] Bernd Heinrich, Marcus Kaiser, and Mathias Klier. How to measure data quality? a metric-based approach. 2007.
- [HPL98] James A Hoagland, Raju Pandey, and Karl N Levitt. Security policy specification using a graphical approach. *arXiv preprint cs/9809124*, 1998.
- [HSB02] J. Hübner, J. Sichman, and O.y Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. *Advances in Artificial Intelligence*, pages 439–448, 2002.
- [HV07] D. Harrison and L. Varveris. *Togaf: Establishing itself as the definitive method for building enterprise architectures in the commercial world*, 2007.
- [Jac01] M. Jackson. *Problem frames: analysing and structuring software development problems*. Addison-Wesley, 2001.
- [Jen00] N.R. Jennings. On agent-based software engineering. *Artificial intelligence*, 117(2):277–296, 2000.
- [JFS06] Ivan J Jureta, Stéphane Faulkner, and Pierre-Yves Schobbens. A more expressive softgoal conceptualization for quality requirements analysis. In *Conceptual Modeling-ER 2006*, pages 281–295. Springer, 2006.
- [JG98] G.R. Jones and J.M. George. The experience and evolution of trust: Implications for cooperation and teamwork. *Academy of management review*, pages 531–546, 1998.
- [JGB79] JM Juran, FM Gryna, and RS Bingham. *Quality control handbook*. New York [etc.]: McGraw-Hill, 1979.

- [Jia10] Lei Jiang. *DATA QUALITY BY DESIGN: AGOAL-ORIENTED APPROACH*. PhD thesis, University of Toronto, 2010.
- [JMF08] Ivan J Jureta, John Mylopoulos, and Stephane Faulkner. Revisiting the core ontology and problem in requirements engineering. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 71–80. IEEE, 2008.
- [Jøs97] A. Jøsang. Prospectives for modelling trust in information security. In *Information Security and Privacy*, pages 2–13. Springer, 1997.
- [JT99] C. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. *Multi-Agent System Engineering*, pages 221–231, 1999.
- [Jür05] Jan Jürjens. *Secure systems development with UML*. Springer-Verlag New York Incorporated, 2005.
- [Ken98] E.A. Kendall. Agent roles and role models: New abstractions for intelligent agent system analysis and design. In *International Workshop on Intelligent Agents in Information and Process Management*. Citeseer, 1998.
- [KKST11] Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: The impact of high frequency trading on an electronic market. *Manuscript, U of Maryland*, 2011.
- [KLP97] Rita Kovac, Yang W Lee, and Leo Pipino. Total data quality management: The case of iri. In *IQ*, pages 63–79, 1997.
- [Kos95] K. Kosanke. Cimosà—overview and status. *Computers in Industry*, 27(2):101–109, 1995.
- [Kra99] R.M. Kramer. Trust and distrust in organizations: Emerging perspectives, enduring questions. *Annual review of psychology*, 50(1):569–598, 1999.
- [LBN99] Jorge Lobo, R Bhatia, and S Naqvi. Apolicy description language. In *Proceedings of AAAI*, pages 291–298, 1999.
- [LC02] Liping Liu and Lauren Chi. Evolutional data quality: A theory-specific view. In *IQ*, pages 292–304, 2002.
- [Lew14] Michael Lewis. *Flash boys: a Wall Street revolt*. WW Norton & Company, 2014.
- [Lie05] Yuliya Lierler. Disjunctive answer set programming via satisfiability. 2005.
- [LJMU95] D.H. Liles, M.E. Johnson, L. Meade, and D.R. Underdown. Enterprise engineering: a discipline? In *Society for Enterprise Engineering Conference Proceedings*, volume 6, pages 45–47. Citeseer, 1995.
- [LMB98] R.J. Lewicki, D.J. McAllister, and R.J. Bies. Trust and distrust: New relationships and realities. *Academy of management Review*, pages 438–458, 1998.
- [LNI⁺03a] Lun-Cheng Lin, Bashar Nuseibeh, Daniel Ince, Michael Jackson, and Jonathan Moffett. Analysing security threats and vulnerabilities using abuse frames. *ETAPS-04*, 2003.
- [LNI⁺03b] Luncheng Lin, Bashar Nuseibeh, Darrel Ince, Michael Jackson, and Jonathan Moffett. Introducing abuse frames for analysing security requirements. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 371–372. IEEE, 2003.

- [LPF⁺06] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The dlvs system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499–562, 2006.
- [Luh88] Niklas Luhmann. Familiarity, confidence, trust: Problems and alternatives. *D. Gambetta, editor, Trust: Making and Breaking of Cooperative Relations*, Basil Blackwell, Oxford, 1988, 1988.
- [Mad11] Ananth Madhavan. Exchange-traded funds, market structure and the flash crash. *SSRN Electronic Journal*, pages 1–33, 2011.
- [MBG⁺03] C Masolo, S Borgo, A Gangemi, N Guarino, A Oltramari, and L Schneider. Dolce: a descriptive ontology for linguistic and cognitive engineering. *WonderWeb Project, Deliverable D17 v2*, 1, 2003.
- [MCN92] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, pages 483–497, 1992.
- [MG07] H. Mouratidis and P. Giorgini. Secure tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.
- [Mis03] Frederic S Mishkin. Policy remedies for conflicts of interest in the financial system. In *Macroeconomics, Monetary Policy and Financial Stability Conference*. Citeseer, 2003.
- [Mot89] A. Motro. Integrity= validity+ completeness. *ACM Transactions on Database Systems (TODS)*, 14(4):480–502, 1989.
- [MR03] Daniel D McCracken and Edwin D Reilly. Backus-naur form (bnf). *Encyclopedia of Computer Science*, pages 129–131, 2003.
- [MS81] R.J. McEliece and D.V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.
- [MU12] Thomas McNish and James Upson. Strategic liquidity supply in a market with fast and slow traders. *Available at SSRN 1924991*, 2012.
- [Nan10] Nanex flash crash summary report. <http://www.nanex.net/FlashCrashFinal/FlashCrashSummary.html>, 2010. Accessed: 2014-05-30.
- [NE00] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000.
- [Nil71] Nils J Nilsson. Problem-solving methods in ai. *Artificial Intelligence*, 1971.
- [NSS00] Ilkka Niemela, Patrik Simons, and Tommi Syrjanen. Smodels: a system for answer set programming. *arXiv preprint cs/0003033*, 2000.
- [Ode98] James J Odell. *Advanced object-oriented analysis and design using UML*, volume 12. Cambridge University Press, 1998.
- [OY11] Maureen O’Hara and Mao Ye. Is market fragmentation harming market quality? *Journal of Financial Economics*, 100(3):459–474, 2011.

- [PADAD05] V. Prabhakaran, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau. Analysis and evolution of journaling file systems. In *Proceedings of the Annual USENIX Technical Conference*, pages 105–120, 2005.
- [PGK88] D.A. Patterson, G. Gibson, and R.H. Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17. ACM, 1988.
- [PHT⁺12] Mark Paddrik, Roy Hayes, Andrew Todd, Steve Yang, Peter Beling, and William Scherer. An agent based model of the e-mini s&p 500 applied to flash crash analysis. In *Computational Intelligence for Financial Engineering & Economics (CIFEr), 2012 IEEE Conference on*, pages 1–8. IEEE, 2012.
- [PLW02] Leo L Pipino, Yang W Lee, and Richard Y Wang. Data quality assessment. *Communications of the ACM*, 45(4):211–218, 2002.
- [PO99a] Richard F Paige and Jonathan S Ostroff. A comparison of the business object notation and the unified modeling language. In *«UML»Š99ŰThe Unified Modeling Language*, pages 67–82. Springer, 1999.
- [PO99b] Richard F Paige and Jonathan S Ostroff. Developing bon as an industrial-strength formal method. In *FMŠ99ŰFormal Methods*, pages 834–853. Springer, 1999.
- [POB00] Richard F. Paige, Jonathan S. Ostroff, and Phillip J Brooke. Principles for modeling language design. *Information and Software Technology*, 42(10):665–675, 2000.
- [PP03] Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing*. Prentice Hall Professional, 2003.
- [PS03] Joseph A Petrick and Robert F Scherer. The enron scandal and the neglect of management integrity capacity. *American Journal of Business*, 18(1):37–50, 2003.
- [PSJ04] Amir Parsanian, Sumit Sarkar, and Varghese S Jacob. Assessing data quality for information products: impact of selection, projection, and cartesian product. *Management Science*, 50(7):967–982, 2004.
- [PWKR05] Leo Pipino, Richard Wang, David Kopcsó, and William Rybold. Developing measurement scales for data-quality dimensions. *ME Sharpe, New York*, 2005.
- [QD02] S. Quinlan and S. Dorward. Venti: a new approach to archival storage. In *Proceedings of the FAST 2002 Conference on File and Storage Technologies*, volume 4, 2002.
- [RB94] Carol A Reeves and David A Bednar. Defining quality: alternatives and implications. *Academy of management Review*, 19(3):419–445, 1994.
- [Red95] T.C. Redman. Improve data quality for competitive advantage. *Sloan Management Review*, 36:99–99, 1995.
- [Red05] TC Redman. Measuring data accuracy: A framework and review. *RY Robert B, Mackay M.,(1998), IT supporting supplier relationships: The role of electronic commerce, European Journal of Purchasing & Supply Management*, 4, 2005.
- [RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.

- [RSB⁺98] D.M. Rousseau, S.B. Sitkin, R.S. Burt, C. Camerer, et al. Not so different after all: A cross-discipline view of trust. *Academy of management review*, 23(3):393–404, 1998.
- [RZFG01] Carlos Ribeiro, Andre Zuquete, Paulo Ferreira, and Paulo Guedes. Spl: An access control language for security policies and complex constraints. In *NDSS*, volume 1, 2001.
- [SC⁺10] Securities, Exchange Commission, et al. Findings regarding the market events of may 6, 2010. *Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues*, 2010.
- [SC12] Laura Sebastian-Coleman. *Measuring Data Quality for Ongoing Improvement: A Data Quality Assessment Framework*. Newnes, 2012.
- [SCC⁺12] Ian Sommerville, Dave Cliff, Radu Calinescu, Justin Keen, Tim Kelly, Marta Kwiatkowska, John Mcdermid, and Richard Paige. Large-scale complex it systems. *Communications of the ACM*, 55(7):71–77, 2012.
- [SCFY96] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
- [SD99] MWA Steen and John Derrick. Formalising odp enterprise policies. In *Enterprise Distributed Object Computing Conference, 1999. EDOC'99. Proceedings. Third International*, pages 84–93. IEEE, 1999.
- [SD00] Maarten WA Steen and John Derrick. Odp enterprise viewpoint specification. *Computer Standards & Interfaces*, 22(3):165–189, 2000.
- [SFR00] Michael Schillo, Petra Funk, and Michael Rovatsos. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14(8):825–848, 2000.
- [SL02] Morris Sloman and Emil Lupu. Security and management policy specification. *Network, IEEE*, 16(2):10–19, 2002.
- [SLW97] D.M. Strong, Y.W. Lee, and R.Y. Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.
- [SOSF04] S. Sadiq, M. Orlowska, W. Sadiq, and C.s Foulger. Data flow and validation in workflow modelling. In *Proceedings of the 15th Australasian database conference-Volume 27*, pages 207–214. Australian Computer Society, Inc., 2004.
- [SPG05] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *ACM Sigmod Record*, 34(3):31–36, 2005.
- [SPP02] M. Scannapieco, B. Pernici, and E. Pierce. Ip-uml: Towards a methodology for quality improvement based on the ip-map framework. In *7th Int'l Conf. on Information Quality (ICIQ-02)*, pages 8–10, 2002.
- [SPP06] P. Stevens, R.J. Pooley, and R. Pooley. *Using UML: software engineering with objects and components*. Addison-Wesley Longman, 2006.
- [SS94] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, 1994.

- [SS05] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.
- [SST10] N. Sidorova, C. Stahl, and N. Trčka. Workflow soundness revisited: checking correctness in the presence of data while staying conceptual. In *Advanced Information Systems Engineering*, pages 530–544. Springer, 2010.
- [Sta96] J. Stader. Results of the enterprise project. *Proceedings of Expert Systems*, 96, 1996.
- [Ste94] Marsh Stephen. Formalising trust as a computational concept. *Ph. n dissertation. University of Stirling, scotland*, 1994.
- [Ste01] D. Stenmark. The relationship between information and knowledge. In *Proceedings of IRIS*, volume 24, pages 11–14, 2001.
- [Sub13] Avaniidhar Subrahmanyam. Algorithmic trading, the flash crash, and coordinated circuit breakers. *Borsa Istanbul Review*, 13(3):4–9, 2013.
- [SV84] E. Simon and P. Valduriez. Design and implementation of an extendible integrity subsystem. In *ACM SIGMOD Record*, volume 14, pages 9–17. ACM, 1984.
- [SWZ00] G. Shankaranarayanan, R.Y. Wang, and M. Ziad. Ip-map: Representing the manufacture of an information product. In *Proceedings of the 2000 Conference on Information Quality*, pages 1–16, 2000.
- [TF99] S. Tseng and BJ Fogg. Credibility and computing technology. *Communications of the ACM*, 42(5):39–44, 1999.
- [Tra10] US Commodity Futures Trading. Preliminary findings regarding the market events of may 6, 2010. 2010.
- [TvdAS09] N. Trčka, W. van der Aalst, and N. Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *Advanced Information Systems Engineering*, pages 425–439. Springer, 2009.
- [UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The knowledge engineering review*, 13(1):31–89, 1998.
- [VL04] Axel Van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering*, pages 148–157. IEEE Computer Society, 2004.
- [VLBDLJ03] A. Van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens. From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering. *Requirements Engineering for High Assurance Systems (RHAS'03)*, page 49, 2003.
- [VLDM95] A. Van Lamsweerde, R. Darimont, and P. Massonet. Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt. *re*, page 194, 1995.
- [Wan98] R.Y. Wang. A product perspective on total data quality management. *Communications of the ACM*, 41(2):58–65, 1998.
- [WKM93] R.Y. Wang, H.B. Kon, and S.E. Madnick. Data quality requirements analysis and modeling. In *Data Engineering, 1993. Proceedings. Ninth International Conference on*, pages 670–677. IEEE, 1993.

- [WRK95] Richard Y Wang, Martin P Reddy, and Henry B Kon. Toward quality data: An attribute-based approach. *Decision Support Systems*, 13(3):349–372, 1995.
- [WS96] R.Y. Wang and D.M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, pages 5–33, 1996.
- [WSF95] Richard Y Wang, Veda C Storey, and Christopher P Firth. A framework for analysis of data quality research. *Knowledge and Data Engineering, IEEE Transactions on*, 7(4):623–640, 1995.
- [WW96] Y. Wand and R.Y. Wang. Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95, 1996.
- [YM94] E. Yu and J. Mylopoulos. From er to ŞarTŪmodelling strategic actor relationships for business process reengineering. *Entity-Relationship Approach ŪER'94 Business Modelling and Re-Engineering*, pages 548–565, 1994.
- [Yu95] Eric Siu-Kwong Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, 1995.
- [Zan06] N. Zannone. *A requirements engineering methodology for trust, security, and privacy*. PhD thesis, PhD thesis, University of Trento, 2006.
- [ZJW03] F. Zambonelli, N.R. Jennings, and M. Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, 2003.
- [ZMP98] A. Zaheer, B. McEvily, and V. Perrone. Does trust matter? exploring the effects of interorganizational and interpersonal trust on performance. *Organization science*, pages 141–159, 1998.