A Goal-based Approach for Automated Specification of Information Quality Policies

Mohamad Gharib
University of Trento
Department of Information and
Communication Technology
Via Sommarive, 9, 38123 Povo, Italy
Email: gharib@disi.unitn.it

Paolo Giorgini
University of Trento
Department of Information and
Communication Technology
Via Sommarive, 9, 38123 Povo, Italy
Email: paolo.giorgini@disi.unitn.it

***Abstract*—Organizational and social aspects are key factors for the analysis of Information Quality (IQ) requirements. This is particularly true in the case of complex socio-technical systems where computerized components coexist with humans and social structures and effective IQ policies can be defined only as a combination of all these factors. Although, several policy-based approaches have been proposed in the literature, none of them offer an adequate conceptual support to combine the technical and organizational perspective in the analysis of IQ requirements. In this paper, we propose a goal-based approach based on an extended version of Secure Tropos to model and analyze IQ requirements from a socio-technical perspective. Our approach provides mechanisms for an automatic derivation of IQ policies, which are specified in terms of permitted, forbidden and obligated activities. Verification of correctness and consistency of the derived policies are supported by automated analysis techniques. We illustrated our approach with an example concerning the stock market system.**

Keywords*—*Information Quality; requirements engineering; goal model; reasoning; specification language; IQ policies.

I. Introduction

The importance of Information Quality (IQ) for organizations is out of discussion, since they depend on information for managing their daily tasks, make important decisions, etc., i.e., and relying on low-quality information may leads to undesirable outcomes [1], or even disasters in the case of critical systems (e.g., air traffic management systems, healthcare systems, etc.). Usually, organizations rely on different kinds of policies for improving the quality of their information (e.g., access control policies [2], security and reliability policies [3], integrity policies [4], etc.), i.e., they use such policies to define the permitted/ forbidden actors' activities toward information in order to deal with IQ related issues.

However, most of the proposed policy-based approaches for dealing with IQ related issues 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 [5]), which are composed of both humans and their social interactions along with the technical components of the system. For example, the Flash Crash (a main U.S market crash) was not caused by a mere technical failure, but it was due to several socio-technical IQ related vulnerabilities that manifested themselves in the system structure and led to the crash [6]. Moreover, many 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).

At the other hand, we have several Requirements Engineering (RE) approaches that are able to capture the organizational along with the social needs of the system-to-be (e.g., *i** [7], Secure Tropos [8] etc.), but their main focus is on the functionality of the system, and they either ignore or loosely define IQ needs. Moreover, we can find several security approaches (e.g., [9]–[12]) that provide concepts for modeling authorizations over information entities, which enable to address accessibility and integrity related issues to a certain limit, but they seem to be limited in addressing several main IQ dimensions (e.g., accuracy, timeliness, consistency, trustworthiness, believability, etc.). In [13], [14], we proposed a RE framework for capturing IQ requirements, and we showed how several IQ related vulnerabilities could be avoided, if the social and organizational aspects of the system were considered properly during the system design. However, our framework did not offer any conceptual support to specify IQ policies in terms of permitted, forbidden, and obligated activities toward information.

In this paper, we propose an extension to our previous framework [14] to model and analyze IQ requirements of the system-to-be. In particular, we provide mechanisms for an automatic derivation of IQ final specifications in terms of IQ policies that clearly identify the permitted, forbidden and obligated activities concerning information usage. Moreover, we propose automated analysis techniques to verify the correctness and consistency of the derived policies. More specifically, the contributions of this paper include:

- Extending the concepts proposed in [14] for modeling IQ requirements by proposing more refined concepts that enable for specifying detailed IQ specifications;
- Extending the analysis support concerning IQ requirements to reason about the new modeling concepts, which is essential to guarantee the correctness and consistency of the requirements model, and in turn the

978-1-4673-6630-4/15/$31.00

correctness and consistency of the derived IQ policies;

- Proposing IQ policy specification language that is able to clearly represents the final IQ specifications in terms of permitted, forbidden and obligated actors' activities concerning information usage;
- Introducing a methodological process for the automatic derivation of IQ policies from the requirements model.

The rest of this paper is organized as follows; we present the baseline of our research in Section II, while Section III describes our motivating example concerning the US stock market system that is used throughout the paper to illustrate our approach. In Section IV, we present our approach for the automatic derivation of the final IQ specifications, and we implement and evaluate our approach in Section V. Section VI presents the related work, and we conclude the paper and discuss the future work at Section VII.

II. Research baseline

Our research baseline is based on three main areas.

(i) Goal-Oriented Requirements Engineering (GORE): is used to represent the stakeholders' requirements in terms of actors of the systems along with their objectives, entitlements, capabilities and their social dependencies. Among the existing approaches (e.g., KAOS [15], *i** [7], Secure Tropos [8]), we adopt an extended version of Secure Tropos [14] as a baseline for our approach, which provides concepts for capturing IQ requirements, beside the basic modeling concepts offered by Secure Tropos. In particular, it introduces primitives for modeling actors that cover two concepts, namely role and agent, where the first is an abstract characterization of the behavior of a social actor within some specialized context, and the latter is an actor with concrete, physical manifestations. Goals are used to represent actors' strategic interests, i.e., a goal is a state of affairs that an actor intends to achieve. Goals can be refined through And/Or decomposition into finer sub-goals. While information is used to represent any informational entities, and it has volatility attribute that can be used to determine its timeliness (validity). An actor can be a legal owner of information item, which gave it a full control over its use.

Goals may produce, read, modify and send information, where produce indicates that an information item can be created by achieving the goal that is responsible of its creation process; reads indicates that consuming an information item is required for the achievement of the goal. Modify indicates that the goal achievement depends on modifying an information item, and send indicates that the goal achievement depends on transferring an information item to a specific destination under predefined criteria. Moreover, the framework extends information provision proposed in Secure Tropos, with a time attribute to represent the transmission time. Finally, it adopts the notion of delegation to model the transfer of entitlements and authorities among actors, and the notion of trust and distrust to capture the actors' expectations of one another concerning their delegated entitlements.

(ii) Organizational and security policies: policies are very important to define the expected actors' behavior in any community, i.e., with the absence of policies, actor will be free to determine their own behavior [16]. The Oxford English Dictionary [17] 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 [18], [19], or as in [20] 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 [21].

(iii) Information Quality: there is a general consensus that IQ is a hierarchal multi-dimensional concept [22], [23], that can be characterized by several dimensions [23], [24], including: accuracy, completeness, timeliness, accessibility, trustworthiness, etc., where each of these dimensions can be used to represent a certain aspect of IQ. We focus on four main IQ dimensions that enable us to address the IQ related issues that we consider in this paper, namely: *Accessibility*: the extent to which information is available, or easily and quickly retrieved [22], we limit accessibility definition to information availability and having the required permission to perform a task at hand; *Accuracy*: means that information should be true or error free with respect to some known, designated or measured values [23]. Some researchers use the term correctness instead of accuracy, yet according to [24] these two terms seems to be equivalent; *Timeliness*: can be defined as to which extent information is valid in term of time, i.e., sufficiently up-to-date for a task at hand [22]; *Consistency*: means that multiple records of the same information should be the same across time and space [23]. In our work, consistency is a time related aspect, while in [25], consistency is used to refer to the "representational consistency" of information.

III. US Stock Market Example

Our example concerns the U.S stock market domain, based on [26], [27], we can identify several stakeholders of the system 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*, and they can be classified based on their behavior and trading capabilities under several main categories, including: *Market Makers*: facilitate trading on a particular security in the market, and they are able to trade large number of securities; *High-Frequency Traders (HFTs)*: are able to trade with very high trading frequency; and *Small traders*: trade small amount of securities with very low trading frequency.

Moreover, *stock markets* are the places where *traders* gather and trade securities (e.g., NYSE, CME, NASDAQ, etc.), and they have a main goal of "making profit by facilitate security trading among traders", i.e., *markets* receive, match, and perform trades from different *traders*. Moreover, *markets* should ensure stable trading environment, i.e., they should halt or slow down the trading frequency when it is needed. In order to do that, they rely on Circuit Breakers (CBs) that are techniques used to stabilize the trading environment by halting or slowing down trading activities to prevent a potential market crash when necessary [28]. In general, CBs rely on CBs information that is produced by analyzing the trading environment.

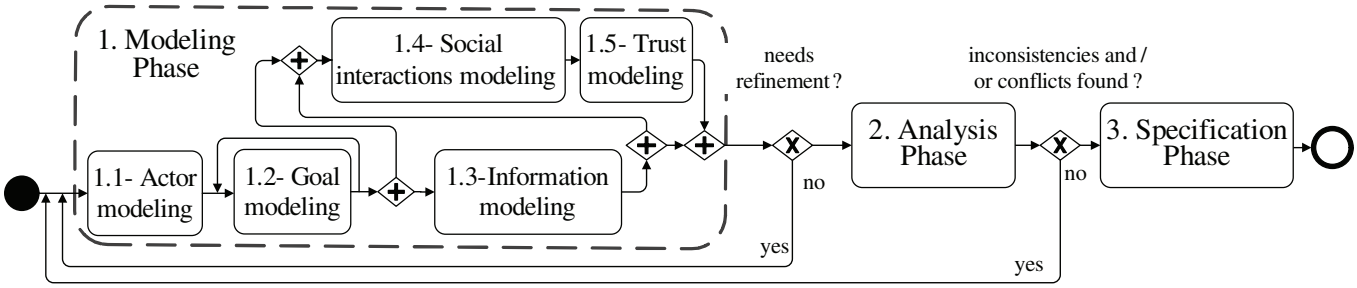

Fig. 1. The process for the automated derivation of IQ policies

Furthermore, *accounting firms* can be defined as firms that provide accounting services to companies for a fee. While *auditing firms* are responsible of auditing the financial statements of legal entities (e.g., persons, companies), where a financial statement is a formal record of the financial activities of such entities. *Consulting firms* provide professional advices concerning some financial securities to *traders* and *investors* for a fee. *Credit assessment ratings firms* are specialized for providing assessments of the credit worthiness of companies' securities. In other words, such firms help traders in deciding how risky it is to invest money in a certain security. Finally, the *Consolidate Tape Association (CTA)* is responsible of producing information about securities (trades and quotes), where such information is very important for analyzing the market and make trading decisions [29]. In particular, *CTA* provides Consolidated Tape System (CTS) information that describes the last orders (trades) information; and Consolidated Quotation System (CQS) information that concerns quotation[1] related information.

IV. Automated Information Quality Policy Specification Approach

IQ policy can be defined 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 an easy 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. Moreover, these policies might be subject to several different social and 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 implemented.

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-to-be. This can be done by deriving such policies from the requirements model of the system, after verifying the correctness and consistency of the model. To this end, we propose a goal-driven approach that builds on top of our previous framework [14], and refines its modeling and analysis techniques for capturing IQ requirements. Moreover, our approach provides mechanisms for the automatic derivation of the final IQ specifications based on the system requirements in terms of clearly defined IQ policies.

[1]A quote is an order that has not been performed

The rest of this section is organized as follows; first we give an overview of the methodological process that underlies our approach. Second, we describe our extended modeling language that proposes refined concepts, which enable for defining detailed IQ specifications, and then we discuss the automated reasoning support that is used to guarantee the correctness and consistency of the requirements model, which is essential to guarantee the correctness and consistency of the derived IQ policies. Finally, we describe our proposed IQ policy specification language, and then we list and discuss the rules that are used for the automatic derivation of IQ policies from the requirements model.

A. Overview of the Approach

An overview of the methodological process that underlies our approach for the automated derivation of IQ policies is shown in Figure 1. The process is specialized for modeling and analyzing IQ requirements of the system-to-be, and then automatically deriving the final IQ specifications based on the verified requirements model in terms of clearly defined IQ policies. In particular, the process is composed of 3 main phases; we briefly describe each of them as follows[2]:

(1) **Modeling phase**: aims to model IQ requirements of the system in their social and organizational context; and it is composed of 5 main steps: (1.1) *Actors modeling*: aims to model the actors (e.g., roles and agents) of the system along with their objectives, entitlements and capabilities; (1.2) *Goals modeling*: aims to identify and model actors' top-level goals and refine them, if needed, through And/ Or-decomposition into leaf goals, which might leads to new iteration of this step; (1.3) *Information modeling*: that is composed of 2 main sub activities: (i) *Information ownership modeling*: aims to identify the legal owners of information items, which is essential to identify who has full control concerning information permissions; and (ii) *Goals-information relations modeling*: that model the different relations between goals and information (e.g., produces, reads, modifies and sends); (1.4) *Social interactions modeling*: aims to identify and model the different social dependencies (interactions) among actors of the system concerning information provision, and the delegation of both authorities and entitlements. More specifically, based on actors' capabilities some goals might be delegated to actors, who have the capabilities to achieve them; and based on actors' needs, information and permissions are provided/ delegated to

[2]We discuss each of these phases in details in sections *B*, *C*, and *D* respectively

them respectively. (1.5) *Trust modeling*: aims to identify and model trust/ distrust among actors of the system concerning goals and permissions delegation. After the modeling phase is finished (both step 1.3 and step 1.5), and if the model does not require any more refinements, we proceed to the next 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.

(3) **Specification phase**: aims to automatically derive the final IQ specifications based on the requirements model in terms of IQ policies that are represented in IQ policy specification language, which clearly define the permitted, forbidden and obligated actions to be carried out by actors of the system toward information.

B. Extended Modeling Language

Our previous framework [14] introduces concepts for modeling actors of the system along with their objectives, entitlements and capabilities, and it adopts the notion of delegation and trust. Moreover, it proposes concepts for capturing IQ requirements in terms of their different dimensions such as accuracy, timeliness, consistency, etc. For instance, it introduce *trusted production/ provision* concepts that enable for capturing information accuracy, and it provides *information volatility*, *read timeliness* and *send timeliness* concepts for capturing information timeliness. Moreover, it proposes *Purpose Of Use* (POU), *interdependent readers* and *read time* concepts for capturing information consistency, where the first captures the intended purpose of information usage, which enables to identify *interdependent readers* that can be defined as actors who read the same information for the same *purpose of use*. While the last concept represents the currency (age) of information that an actor has. We say that information is consistent among its *interdependent reader*, if all of them have the same *read time*, and it is inconsistent otherwise[3].

However, some of the proposed IQ concepts are at high abstraction level, and seem to be inappropriate for identifying detailed IQ specifications, i.e., we cannot rely on them for specifying the permitted, forbidden and obligated activities toward information. For instance, it proposes *trusted /distrusted provision* that helps in analyzing the accuracy of the transferred information, yet we cannot rely on such concepts to derive detailed IQ specifications, i.e., such concept cannot help in determining information transfer types, or identifying the allowed/ forbidden activities on the transferred information. In other words, our previous framework does not support 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 [30].

Figure 2 shows a portion of a goal model concerning the stock market system represented in the refined modeling language. For instance, `Pro trading` is an agent that plays `Small trader` role, which is specialized (*is_a*) from the role `Stock trader` that has a main goal of `Make profit by trading security`, which is And-decomposed into `Analyze the market for targeted securities` and `Make profit by producing the right orders`. Moreover, the goal `Make profit by producing the right orders` `(R)eads` and `(M)odifies Investor's orders`, and it `(P)roduces Trader's orders`, and it needs to `(S)end` both `Investor's orders` and `Trader's orders` to the *stock market*. A `stock investor` provides (Integrity Provision (IP)) `investor's orders` to `trader`, and since it is the `owner` of such information, it delegates (R)ead, (M)odify and (S)end permissions concerning it to the `trader`. Finally, a *stock investor* delegates the goal `Make profit from trading securities` to *stock trader*, and trusts it for its achievement.

C. Analyzing the Requirements Model

In this section, we describe the automated reasoning support that our approach proposes to guarantee the correctness and consistency of the requirements model, which is essential to guarantee the correctness and consistency of the derived IQ policies. In order to verify the correctness and consistency of the requirements model, we provide a Datalog [31] formalization of all the concepts that have been introduced in the paper, along with the reasoning axioms[4]. Moreover, we define a set of properties of the design (shown in Table I) that can be used to verify the correctness and consistency of the requirements model; in what follows we discuss each of these properties:

Pro1-3 verify goals related properties, where **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. 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 holds between the delegator and the delegatee, since delegation with no trust leaves the delegator with no guarantee about the achievement of the delegated goal. **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.

[3]For more information about the concepts and the formal framework, you can refer to [13], [14]

[4]The formalization of the concepts and axioms is omitted due to space limitation, yet they can be found at https://mohamadgharib.wordpress.com/iqp/

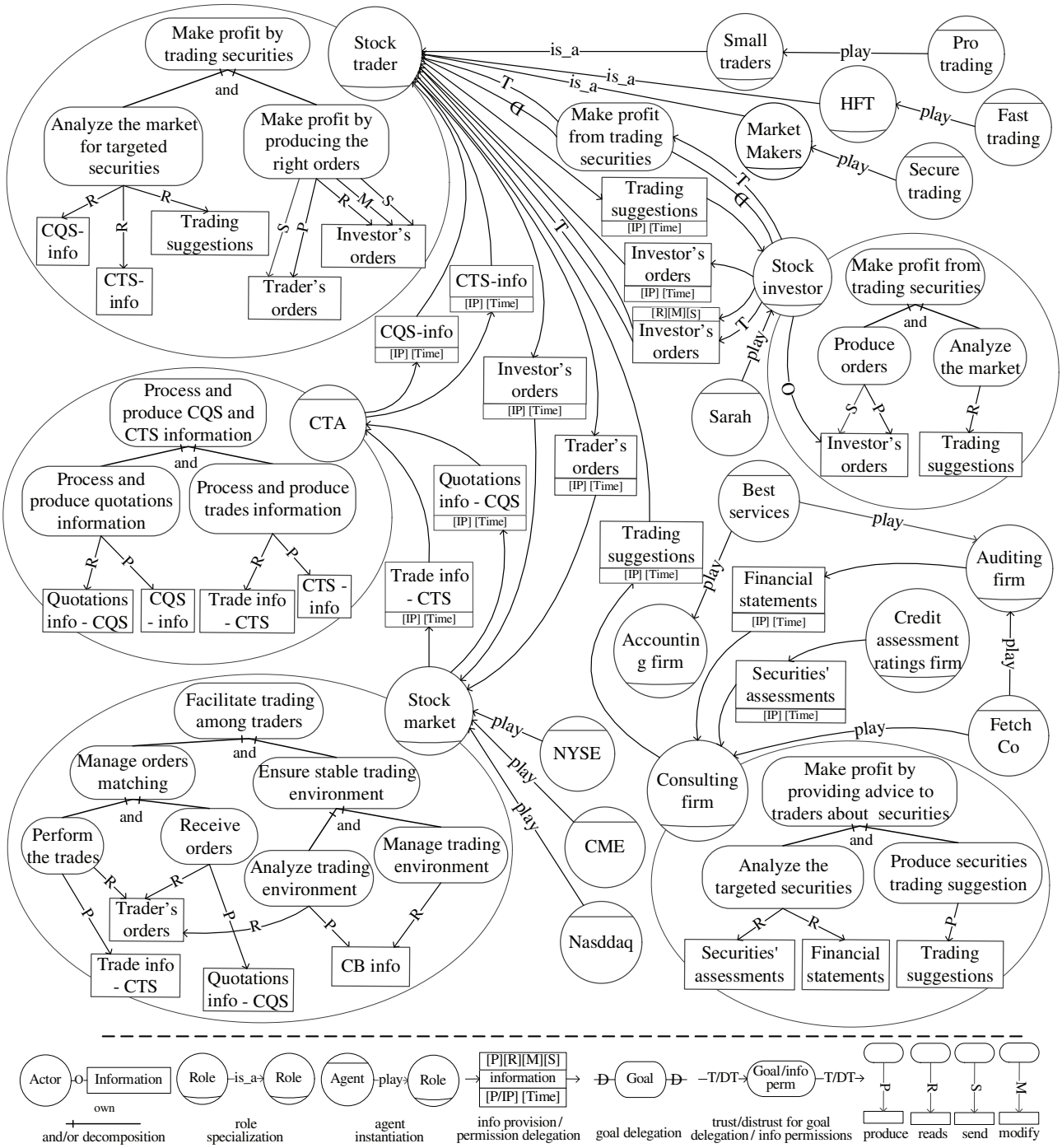

Fig. 2. A partial goal model concerning the U.S stock market structure

For example, in Figure 2 if the *stock investor* delegates its goal "Make profit from trading securities" to *stock trader* with no trust relation, *Pro2* is able to detect such situation and notify the designer. While if the designer mistakenly delegate the *investor* goal to any other actor but the *trader*, and there was no valid delegation to the *trader*, *Pro3* will be able to detect such violation.

Pro4-5 are used to verify information related properties. For instance, **Pro4** states that actors should have all information that is required for the achievement of the goals they are responsible of. While **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.

For example, in Figure 2, *Pro4* is able to detect if the *investor* did not receive "trading suggestions" from the *trader*. While if such information was provided to *stock market*, *Pro5* will be able to detect such unrequited provision, since infor-

TABLE I. PROPERTIES OF THE DESIGN

Pro1	:- aims(A,G), not achieved(A,G)
Pro2	:- deleChain(A, B, G), not trustChain(A, B, achieve, G)
Pro3	:- deleChain(_,A,G), not can_achieve(A,G)
Pro4	:- needs(A,I), information(I, T), not has(A,I, T)
Pro5	:- provideChain(_,A,I), needs(B,I), not needs(A,I), not provideChain(A,B,I)
Pro6	:- need_perm(P, A, I), not has_perm(P, A, I)
Pro7	:- has_perm(P, A, I), not need_perm(P, A, I), not own(A, I)
Pro8	:- dele_perm(P, A, B, I), not has_perm(P, A, I)
Pro9	:- has_perm(P, B, I), owner(A,I), not trust_perm_chain(P, A, B, I)
Pro10	:- read(G, I), not fits_read(G, I)
Pro11	:- send(T, G, B, I), not fits_send(T, G, B, I)
Pro12	:- play(A, R1), play(A, R2), conflict_roles(R1, R2, produce, I), producer(A, I, T)
Pro13	:- play(A, R1), play(A, R2), conflict_roles(R1, R2, read, I), reader(A, I)

mation is not required by the *market*, and no valid provision to actors who require them exist.

Pro6-9 are used to verify information permissions related properties. For instance, **Pro6** states that actors should have all the permission they require 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. While **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.

For example, in Figure 2, *Pro6* will be able to detect if the *investor* did not delegate (R)ead, (M)odify, or (S)end permissions to the *trader* concerning its *orders*, since such permissions are needed by the *trader* to achieve its objectives. While, if *consulting firm* delegates (M)odify permission to a *trader*, *Pro7* will be able to detect such violation, since the *trader* does not require such permissions. Moreover, if the *investor* delegates any permission related to its *orders* to a *trader*, *Pro8* will notify the designer, if no trust holds between the *investor* and the *trader* concerning the delegated permissions. While if a *trader* delegates any permission related to the *investor's orders* to any other *trader*, *Pro9* will notify the designer, if no trust holds between the delegtee and the *investor* (information owner) concerning the delegated permissions.

Pro10-11 are used to verify IQ related properties in the model. For instance, **Pro10** states that the model should not include any information that does not fits for the purpose of read (appropriate for read) from the perspectives of their readers, where information should be accessible, accurate, valid and consistent to be considered appropriate for read. In particular, *Pro10* is able to detect: (1) information accuracy, where information is inaccurate if it is produced with no permissions, modified intentionally/ intentionally during its transfer; (2) information accessibility, information is inaccessible to an actor, if the actor does not has read permissions; (3) information timeliness (validity), where timeliness can be analyzed depending on information *currency* that is the time interval between information creation (or update) to its usage time [22], [25], and information *volatility* that is the change rate of information value [25], i.e., information is not valid, if its currency is bigger than its volatility interval, otherwise it is valid; (4) information consistency, where information is inconsistent if there exist at least two *interdependent readers* (actors read information for the same purpose of use), who have different *read times* that indicates the actual read time by information *reader*. While **Pro11** states that the model should not include any information that does not fits for send from the perspectives of their senders, where information should be accurate and valid at its intended destination to be considered appropriate for send.

Pro12-13 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. Defining the conflicting roles in terms of producing or reading information is not an easy task. Usually, it is done with the help of domain experts, and conflicting roles for produce or read are added to the model as logical constraints, which enable the automated analysis to detect such situations at the instance (agents) level. For example, in Figure 2 *Best services* is playing two conflicting roles ("Accounting firm" and "Auditing firm") concern producing "Financial statement", i.e., it may provide accounting and auditing services for the same company, since we cannot trust a firm for *producing* auditing information ("Financial statement") for a company that it gets paid from to perform their accounting services (e.g., The Enron scandal [32]), *Best services* should be prevented from producing such information. Similarly, *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. If any of the previously discussed cases happened in the model properties *Pro12-13* will be able to detect such situation and notify the designer.

D. Automated Specification of Information Quality Policies

After verifying the correctness and consistency of the requirements model, we are able to derive correct and consistent IQ specification. In order to do that, we need to define IQ policy specification language to represent such policies (section 1), and we need to define the required rules that enable for the automatic derivation of such policies from the requirements model (section 2).

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 polices, namely: permit, forbid, and obligate policies that are used to control four different types of activities over information, namely, produce, read, modify, and send. Moreover, the IQ policy specification language can be represented in BNF notation [33] 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*[5] 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_the_right_orders`**)**

permitted_modify(`trader, investors_order,` **for**, `make_profit_by_producing_the_right_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)]`**)**

For example, if a *trader* 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`**)**

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 covers 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`**)**

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. In particular, we define three sets of rules[6] (shown in Table II), 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

[5]We assume it has the required permissions

[6]Derivation rules (axioms) that are also represented in Datalog language

TABLE II. RULES FOR AUTOMATED DERIVATION OF IQ SPECIFICATIONS

```
P1 permitted_produce(A,I):- has_perm(p,A,I), not forbidden_produce(A,I).
P2 permitted_read(A,I,for,G):- own(A,I), is_responsible(A,G), not forbidden_read(A,I,for,G).
P3 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).
P4 permitted_send(A,I,to,B):- has_perm(s,A,I), is_responsible(A,G), send(T,G,B,I).
P5 permitted_modify(A,I,for,G):- has_perm(m,A,I), is_responsible(A,G), modify(G,I).
F1 forbidden_produce(A,I):- play(A, R1), play(A, R2), conflict_roles_produce(R1, R2, produce, I).
F2 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).
F3 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.
F4 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.
F5 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.
O1 obligated_read(A,I,in,0):- interdependent_readers(A,B,I).
O2 obligated_provide(A,B,I,in,T):- provideChain(_,T,A,B,I).
O3 obligated_send(A,I,to,B,in,T):- sender(T,A,B,I).
```

into consideration the activities that might be prevented from performing (e.g., conflict of interests).

P1 states that an actor is permitted to produce an information item, if it has the related 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 responsible of its achievement, if it is the owner of such information, and there is no reason forbidding it from reading such information. While **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. Finally, **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.

For example, in Figure 2 *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. While **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. Moreover, **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. Finally, **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.

For example, in Figure 2 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*).

Our previous framework hardly recognizes the obligation concept, which is essential for addressing several IQ related vulnerabilities by defining the activities that actors must perform toward information. For example, in Figure 2 if a *stock investor* depends on a *stock trader* for sending some trading orders to a *stock market*, our previous framework 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 framework. Similarly, there is no guarantee that a *stock market* will provide information concerning 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 [34] 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

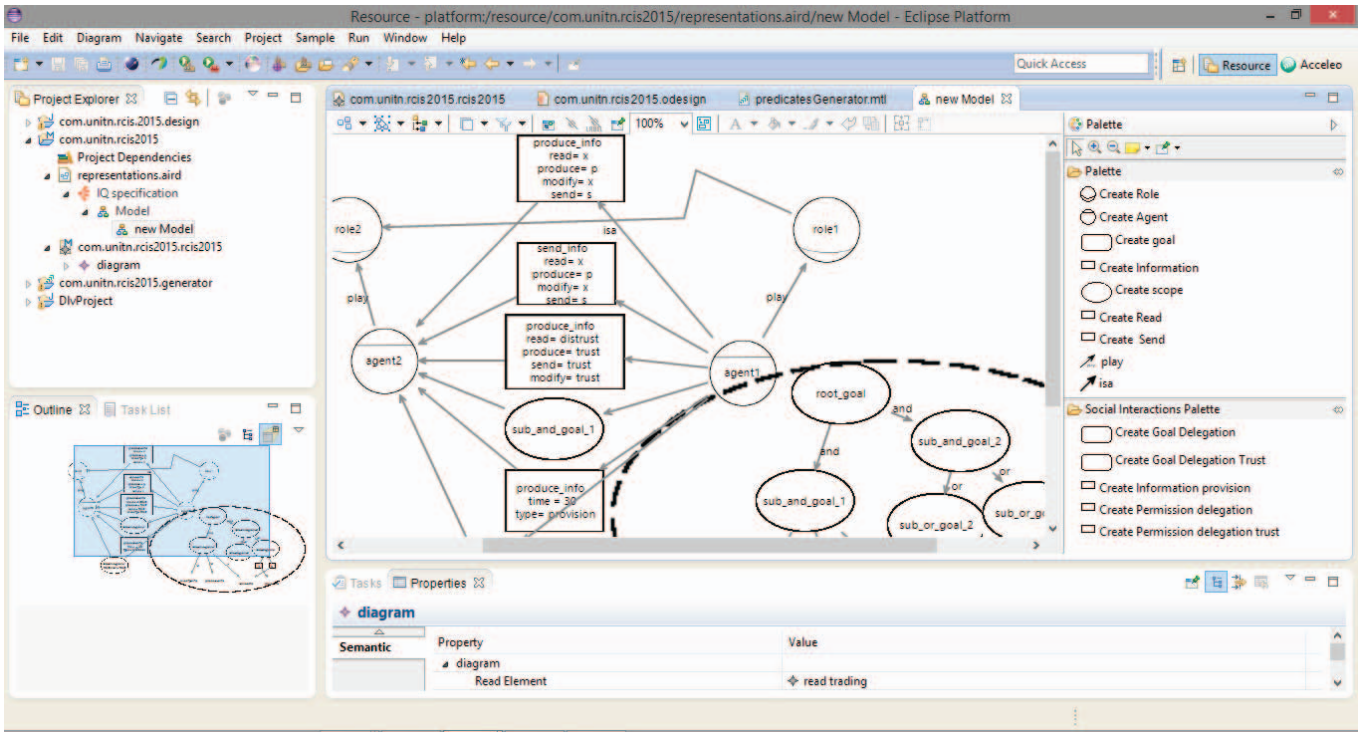

Fig. 3. Screenshot of the IQ Policy Specification Tool

address information consistency. However, these concepts do not guarantee that all *interdependent readers* will use 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 identify the activities that actors must perform over information.

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

V. Implementation and Evaluation

We evaluated our approach on a simulation basis following [35], namely developing a prototype tool and test its applicability with artificial data. In particular, by means of our prototype implementation[7], we evaluated the automated derivation of IQ policies from the requirements model, i.e., the ability of capturing IQ requirements in their social and organizational context, and then automatically derive the final IQ specifications in terms of clearly defined IQ policies. In what follows, we briefly describe the tool (a snapshot of the tool is shown in Figure 3), discuss its applicability over the Flash Crash scenario[8], and then we test the scalability of its reasoning support.

Prototype implementation: our tool consist of three main parts: (1) A graphical user interface (GUI)[9]: that help designers while drawing the requirements model by drag-and-drop modeling elements from palettes, and enables them for specifying the different properties of these elements along with the different relations among these elements. Moreover, it is able to prevent modeling any relation among the elements that violates the modeling language semantics; (2) Model-to-text transformation: supports the transformation of the graphical requirements model into Datalog formal specifications depending on Acceleo[10]; (3) automated reasoning support (DLV system[11]) that takes the Datalog specifications, which

[7]The prototype tool is available at https://mohamadgharib.wordpress.com/iqp/

[8]For more information about the case study refer to [13]

[9]Developed using Sirius https://projects.eclipse.org/projects/modeling.sirius

[10]https://projects.eclipse.org/projects/modeling.m2t.acceleo

[11]http://www.dlvsystem.com/dlv/

has been resulted from translating the graphical model along with an already defined reasoning axioms, and then verifies the correctness and completeness of the requirements model against the properties of the design. Finally, it enables for the automated derivation of the final IQ specification in terms of clearly defined IQ policies.

Applicability: we tested the applicability of our approach by applying it to the Flash Crash scenario. In particular, we modeled the scenario, transform it into Datalog specification, and then we run the automated analysis to test its ability in discovering violations to the properties of the design. The analysis detects several violations of the properties of the design, including:

- 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.
- Inconsistent information: both of *NYSE* and *NASDAQ* are *interdependent readers* concerning "CME CB info". However, "CME CB info" is provided to them with different provision times. According to [36], 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 these two markets, and resulted in inconsistency between them.

These violations were addressed one-by-one until we reached a point that no violations were returned when we run the analysis, i.e., the model is correct and consistent. Finally, we used the tool to derive the final IQ specifications of the requirements model.

Experiments on scalability: we have investigated how the reasoning execution time is affected by the size of the model. In particular, we expanded the model shown in Figure 2 by increasing the number of its modeling elements (adding more actors, goals, information, social dependencies, etc.), and then we run the reasoning and calculated the execution time. More specifically, the model have been gradually expanded from 140 modeling elements to 1335 elements through 10 steps, and we investigated the reasoning execution time at each of these step by repeating the reasoning execution seven times, discarding the fastest and slowest ones, and then computed the average execution time of the rest. We have performed the scalability experiment on laptop computer, Intel(R) core(TM) i3- 3227U CPU@ 190 GHz, 8GB RAM, OS Window 8.1, 64-bit. The result is shown in Figure 4, and the relation between the size of the model and the execution time is still linear (not exponential) for the considered model sizes.

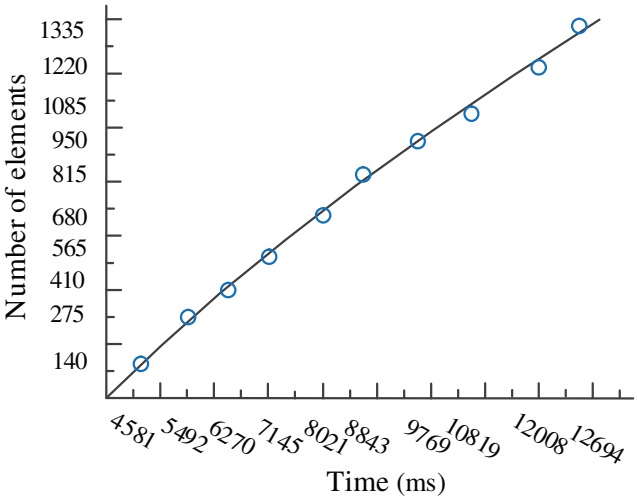

Fig. 4. Scalability results with increasing the number of modeling elements

VI. Related Work

Most RE approaches were not designed to capture IQ requirements, i.e., they either ignore or loosely define such requirements. However, we can find several RE approaches that rely on the top-level goals of the system to define different kinds of policies that can be used to control the actors' behavior. For example, Antón and Earp [37] depend on GBRAM method [38] to operationalize security goals into security policies. While Fontaine [39] propose a process for mapping KAOS [15] goal models into security policies that can be represented in ponder policy specification language [40]. In [41], security concerns were represented as security goals that can be refined and made precise as security requirements on the software, and security policies on the operational environment. Giorgini et al. [9] propose a methodology for modeling security requirements as constraints that can be imposed to the different actors activities. While Massacci et al. [10] introduce a framework that extends SI* [11] with the required mechanisms to derive access control policies from security requirements. Finally, Paja et al. [12] propose a framework that supports the identification and management of conflicts in security requirements. Even their framework supports modeling of authorizations over information entities in terms of permissions, but offers no concepts for modeling or reasoning about IQ related aspects.

At the other hand, several organizational and security policy languages have been proposed. For instance, Hoagland et al. [19] 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 [21] 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) [42] 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. Security Policy Language (SPL) [43] 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 [40] is able to support a number of basic policies, including: obligation, information filtering, delegation, and refrain policies.

Finally, we can find various approaches for improving IQ by design that have been proposed in literature. For example, Wang and Strong [44] propose the Total Data Quality Management (TDQM) methodology. While, in [45] they develop IP-MAP that is a formal modeling method for creating high quality Information Product (IP). Relaying on IP-MAP framework, Scannapieco et al. [46] propose IP-UML approach that combines both data analysis and process analysis in order to improve IQ. Moreover, Guerra-García et al. [47] introduce a Model Driven Architecture (MDA) for the management of Data Quality (DQ) during the design and development of Web applications. In particular, they propose a meta-model and a UML profile for capturing and specifying DQ requirements (DQ_WebRE). Finally, Cappiello et al. [48] propose a methodology that is composed of four main steps to support business process designers in identifying DQ requirements, and selecting the required actions (improvements) during the design of business processes in order to satisfy the DQ requirements.

VII. Conclusions and Future Work

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. 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 be implemented. Furthermore, we advocate that such needs should be considered from the early system requirements analysis phase, which prevent revising the entire system to accommodate such requirements when they are needed, which might be very costly.

In particular, we proposed a goal-based approach based on an extended version of Secure Tropos modeling language [14], and extends its modeling language with more refined concepts to capture IQ requirements from a socio-technical perspective. Moreover, we discussed the analysis techniques that the framework offers to verify the correctness and consistency of the requirements model. At the other hand, we provide mechanisms for the automatic derivation of the final IQ specification from the requirements model in terms of IQ policies that define the permitted, forbidden and obligated activities toward information. More specifically, we defined IQ specification language, and we describe several rules that can be used to derive these policies from the requirements model.

As highlighted in [48], specifying IQ requirements is not a trivial task. However, one main advantage of our approach is its ability to specify correct and consistent IQ policies, since it enables to derive 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, the approach 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.

For the future work, we plan to extend the IQ dimensions that we considered in this paper (e.g., credibility, trustworthiness, etc.), and we plan to investigate IQ dimensions along with their interrelations more deeply. Also we plan to extend our IQ Policy specification language to accommodate such extensions. Moreover, we aim to enrich our IQ policy specification language 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. Furthermore, we aim to evaluate our framework with end users (designers), i.e., we are planning to perform a set of experiments to assess the adequacy of our proposed framework. Finally, we aim to better validate our approach by applying it to several other case studies that belong to different domains.

Acknowledgment

This research was partially supported by the ERC advanced grant 267856, “Lucretius: Foundations for Software Evolution”, http://www.lucretius.eu/.

References

[1] T. Redman, “Improve data quality for competitive advantage,” *Sloan Management Review*, vol. 36, pp. 99–99, 1995.

[2] R. S. Sandhu and P. Samarati, “Access control: principle and practice,” *Communications Magazine, IEEE*, vol. 32, no. 9, pp. 40–48, 1994.

[3] E. Simon and P. Valduriez, “Design and implementation of an extendible integrity subsystem,” in *ACM SIGMOD Record*, vol. 14, no. 2. ACM, 1984, pp. 9–17.

[4] A. Motro, “Integrity= validity+ completeness,” *ACM Transactions on Database Systems (TODS)*, vol. 14, no. 4, pp. 480–502, 1989.

[5] F. Emery and E. Trist, “Socio-technical systems. management sciences, models and techniques. churchman cw et al,” 1960.

[6] I. Sommerville, D. Cliff, R. Calinescu, J. Keen, T. Kelly, M. Kwiatkowska, J. Mcdermid, and R. Paige, “Large-scale complex it systems,” *Communications of the ACM*, vol. 55, no. 7, pp. 71–77, 2012.

[7] E. S.-K. Yu, “Modelling strategic relationships for process reengineering,” Ph.D. dissertation, University of Toronto, 1995.

[8] H. Mouratidis and P. Giorgini, “Secure tropos: A security-oriented extension of the tropos methodology,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 2, pp. 285–309, 2007.

[9] P. Giorgini, H. Mouratidis, and N. Zannone, “Modelling security and trust with secure tropos,” *Integrating Security and Software Engineering: Advances and Future Visions*, 2006.

[10] F. Massacci and N. Zannone, “A model-driven approach for the specification and analysis of access control policies,” in *On the Move to Meaningful Internet Systems: OTM 2008*. Springer, 2008, pp. 1087–1103.

[11] N. Zannone, “A requirements engineering methodology for trust, security, and privacy,” Ph.D. dissertation, PhD thesis, University of Trento, 2006.

[12] E. Paja, F. Dalpiaz, and P. Giorgini, “Managing security requirements conflicts in socio-technical systems,” in *Conceptual Modeling*. Springer, 2013, pp. 270–283.

[13] M. Gharib and P. Giorgini, “Detecting conflicts in information quality requirements: the may 6, 2010 flash crash,” Universitá degli studi di Trento, Tech. Rep., 2014.

[14] ——, "Modeling and reasoning about information quality requirements," in *Requirements Engineering: Foundation for Software Quality - 21st International Working Conference, REFSQ 2015, Essen, Germany, March 23-26, 2015. Proceedings*, 2015, pp. 49–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-16101-3_4

[15] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of computer programming*, vol. 20, no. 1-2, pp. 3–50, 1993.

[16] M. Steen and J. Derrick, "Formalising odp enterprise policies," in *Enterprise Distributed Object Computing Conference, 1999. EDOC'99. Proceedings. Third International*. IEEE, 1999, pp. 84–93.

[17] O. E. Dictionary, "Oxford: Oxford university press," 1989.

[18] M. Sloman and E. Lupu, "Security and management policy specification," *Network, IEEE*, vol. 16, no. 2, pp. 10–19, 2002.

[19] J. A. Hoagland, R. Pandey, and K. N. Levitt, "Security policy specification using a graphical approach," *arXiv preprint cs/9809124*, 1998.

[20] C. P. Pfleeger and S. L. Pfleeger, *Security in computing*. Prentice Hall Professional, 2003.

[21] M. W. Steen and J. Derrick, "Odp enterprise viewpoint specification," *Computer Standards & Interfaces*, vol. 22, no. 3, pp. 165–189, 2000.

[22] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," *Communications of the ACM*, vol. 45, no. 4, pp. 211–218, 2002.

[23] M. Bovee, R. P. Srivastava, and B. Mak, "A conceptual framework and belief-function approach to assessing overall information quality," *International journal of intelligent systems*, vol. 18, no. 1, pp. 51–74, 2003.

[24] Y. Wand and R. Wang, "Anchoring data quality dimensions in ontological foundations," *Communications of the ACM*, vol. 39, no. 11, pp. 86–95, 1996.

[25] R. Wang and D. Strong, "Beyond accuracy: What data quality means to data consumers," *Journal of management information systems*, pp. 5–33, 1996.

[26] A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun, "The flash crash: The impact of high frequency trading on an electronic market," *Manuscript, U of Maryland*, 2011.

[27] F. S. Mishkin, "Policy remedies for conflicts of interest in the financial system," in *Macroeconomics, Monetary Policy and Financial Stability Conference*. Citeseer, 2003.

[28] P. Gomber, M. Haferkorn, M. Lutat, and K. Zimmermann, "The effect of single-stock circuit breakers on the quality of fragmented markets," in *FinanceCom*, 2012, pp. 71–87.

[29] C. W. Holden and S. Jacobsen, "The breakdown of standard microstructure techniques: And what to do about it," *Available at SSRN 1911491*, 2011.

[30] M. Gharib and P. Giorgini, "Modeling and analyzing information integrity in safety critical systems," in *Advanced Information Systems Engineering Workshops - CAiSE 2013 International Workshops, Valencia, Spain, June 17-21, 2013. Proceedings*, 2013, pp. 524–529. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-38490-5_48

[31] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of databases*. Citeseer, 1995.

[32] J. A. Petrick and R. F. Scherer, "The enron scandal and the neglect of management integrity capacity," *American Journal of Business*, vol. 18, no. 1, pp. 37–50, 2003.

[33] D. D. McCracken and E. D. Reilly, "Backus-naur form (bnf)," *Encyclopedia of Computer Science*, pp. 129–131, 2003.

[34] "Nanex flash crash summary report," http://www.nanex.net/FlashCrashFinal/FlashCrashSummary.html, 2010, accessed: 2014-05-30.

[35] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[36] M. Lewis, *Flash boys: a Wall Street revolt*. WW Norton & Company, 2014.

[37] A. I. Antón and J. B. Earp, "Strategies for developing policies and requirements for secure electronic commerce systems," in *E-commerce security and privacy*, vol. 2. Citeseer, 2000, pp. 29–46.

[38] A. Anton, "Goal-based requirements analysis," in *Requirements Engineering, 1996., Proceedings of the Second International Conference on*. IEEE, 1996, pp. 136–144.

[39] P.-J. Fontaine, "Goal-oriented elaboration of security requirements," Master's thesis, Dept. Computing Science, University of Louvain, 2001.

[40] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *Policies for Distributed Systems and Networks*. Springer, 2001, pp. 18–38.

[41] A. Van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, 2004, pp. 148–157.

[42] J. Lobo, R. Bhatia, and S. Naqvi, "Apolicy description language," in *Proceedings of AAAI*, 1999, pp. 291–298.

[43] C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes, "Spl: An access control language for security policies and complex constraints." in *NDSS*, vol. 1, 2001.

[44] R. Wang, "A product perspective on total data quality management," *Communications of the ACM*, vol. 41, no. 2, pp. 58–65, 1998.

[45] G. Shankaranarayanan, R. Wang, and M. Ziad, "Ip-map: Representing the manufacture of an information product," in *Proceedings of the 2000 Conference on Information Quality*, 2000, pp. 1–16.

[46] M. Scannapieco, B. Pernici, and E. Pierce, "Ip-uml: Towards a methodology for quality improvement based on the ip-map framework," in *7th Intl Conf. on Information Quality (ICIQ-02)*, 2002, pp. 8–10.

[47] C. Guerra-García, I. Caballero, and M. Piattini, "Capturing data quality requirements for web applications by means of dq_webre," *Information Systems Frontiers*, vol. 15, no. 3, pp. 433–445, 2013.

[48] C. Cappiello, A. Caro, A. Rodriguez, and I. Caballero, "An approach to design business processes addressing data quality issues." in *ECIS*, 2013, p. 216.