Modeling and Analyzing Information Integrity in Safety Critical Systems

Mohamad Gharib and Paolo Giorgini

University of Trento, Italy

Abstract. In safety critical systems one important aspect of information quality is ensuring information integrity. Typically, information integrity is a problem handled at technical level through solutions such as checksumming, integrity constraints, and integrity correction techniques. However, information integrity cannot be considered only as a technical problem, but it has to be analyzed and studied in the social-technical context of the system. Business processes and relations among all involved actors may also affect correctness and consistency of the information. In this paper, we propose an extended version of i^*/Tropos modeling language to capture and analyze information integrity requirements. We illustrate the Datalog formalization of the proposed concepts and analysis techniques to support the analyst in the verification of integrity related properties. A case study concerning Air Traffic Management (ATM) is used throughout the paper.

Keywords

Information Integrity, Security, Modeling, Analysing

1 Introduction

The usage of low quality information may cause severe consequences in organizations and result in a loss of millions of dollars [18]. Business processes can be compromised during their execution and tactical and strategic decisions can be taken on the base of incorrect informations [19]. Integrity is surely one important aspect of Information Quality (IQ) and becomes an essential feature of any safety critical system, where incorrect or conflictual information may produce disaster and loss of humans lives (e.g., Air Traffic Control Management Systems).

Information integrity is traditionally considered in the way information are stored and transmitted and several techniques and solutions have been proposed in the literature such as, integrity violations avoidance (e.g. Readonly Storage [17]), integrity violations detection (e.g. Checksum [10]), integrity correction techniques (e.g. RAID [16]), and integrity constraints [14]. However, these solutions mainly focus on technical aspects of information integrity and do not solve problems that may rise at business and organizational level.

Consider for example an *Air Traffic Controller*, that has to provide the takeoff clearance to the *Airlines captain*. If the takeoff clearance is provided electronically its integrity can be preserved by a technical solution, such as checksum, which is a fixed-size datum computed from a block of digital data. The integrity of the data can be checked at any time by computing the checksum and comparing it with the stored one. If they match, the data is not altered. However, this is not always the case and very often, even in safety critical systems, incorrectness or inconsistency of information can be due to unsafe and "too flexible" procedures that pilots and controllers can adopt during their interaction. An example is the Charkhi Dadri mid-air collision, where, as described in the investigating commission report [11], the Kazakhstani pilot did not understood correctly the information transmitted by the controller and there were no procedures imposing the pilot to readback the received information.

Information integrity is always a socio-technical problem, where socio and organizational aspects have to be considered along technical solutions. So for example, in the Chicago-O'Hare runway collision [20], the ATC failed to notify the pilot that one of the runways was being used by another airplane and there was no technical support to monitor and alarm the pilot about the problem. This means that the design of a critical system has always to go through a socio-technical analysis, where needs about information integrity have to be identified and studied with respect to all involved social and technical actors, their local and global objectives and their mutual interactions.

Requirement engineering community suggests concepts and languages for capturing and analyzing the high-level security aspects of the system-to-be, but they did not appropriately support the modeling and analysis of information integrity (e.g. [22,12,3,13]). In this paper, we propose an extended version of i^*/Tropos [23,7] modeling language to capture and analyze information integrity requirements starting from a socio-technical perspective. We formalize the language in Datalog and we will propose formal analysis techniques to support the analyst in the verification of properties about information integrity over socio-technical models.

The paper is organized as follows. Section (§2) describes the ATM case study used throughout the paper. Section (§3) discusses about modeling information integrity, while Section (§4) presents the Datalog formalization of the extended version of the i^*/Tropos language. Section (§5) shows how the formalization is used to check information integrity related properties and, finally, we occlude the paper with related work and final consideration in Section (§6).

2 Case Study

Our case study concerns System Wide Information Management (SWIM), which is an advanced technology program designed to facilitate greater information sharing between Air Traffic Management (ATM) systems. SWIM enables information sharing between ATM services across the whole European ATM system segments (Civil and Military, Ground-Ground and Air-Ground) [21]. Informa-

tion handled by SWIM, includes but not limited to, AOC Data (flight plans), Capacity Data, Separation data, and Meteo Data. These information are produced, shared and consumed by different actors of the system. For instance, any airlines (AOC) wants to fly in Europe has to produce and send its flight plan to EUROCONTROL where it is checked and send out to all Air Traffic Control Units (ATCU) that is affected by the flight. Furthermore, each airport and ATCU produce and published maximum capacity, and CFMU uses this information to manage the air traffic flow.

Furthermore, Air Traffic Control (ATC) is a service provided by ground-based controllers (ATCs) working at different ATCU, with the main purpose of preventing aircraft collisions by providing airline captain with the required separation information. For instance, the ground controllers provide separation information at maneuvering area, while the local controllers provide separation information during take-off and landing. Moreover, departure and approach controllers provide separation information during the departure and approach phase respectively. Finally, en-route controllers provide separation information within its sector.

In such systems, information (e.g. AOC data, Separation data, and Capacity data) might be used by several actors, and the system should be able to maintain their integrity between all those actors at any time. One way to enhance the integrity of information is by controlling the delegation of modification permission, which can prevent unauthorized actors from modifying it. However, this does not ensure that the authorized actors will not misuse their permissions, which may be a main reason of compromising the integrity of information in the system. Thus, permissions should be delegated based on some criteria that can be captured by the social aspects of the system (trust), i.e., there should be a trust relation along with any permission delegation to ensure that such permessions will not be misused. Figure 1 shows the main actors of the ATM system along with their goals and information.

3 Modeling Information Integrity

Our modeling language revises and extends the i* modeling language [23] with new primitives for capturing information integrity requirements of the system-to-be. It provides primitives for modeling actors, their specializations, their goals, their relations with information, the notion of delegation and trust among them. In the following section, we discuss information and information integrity dimensions to get better understanding of what information integrity requirements are, and when they are needed.

3.1 Information and Information Integrity

Information [1] can be produced in many different ways. Following Buckland[8],four sources of information are identified: (1) information can be generated internally;

[1] In this paper data and information are used as synonyms

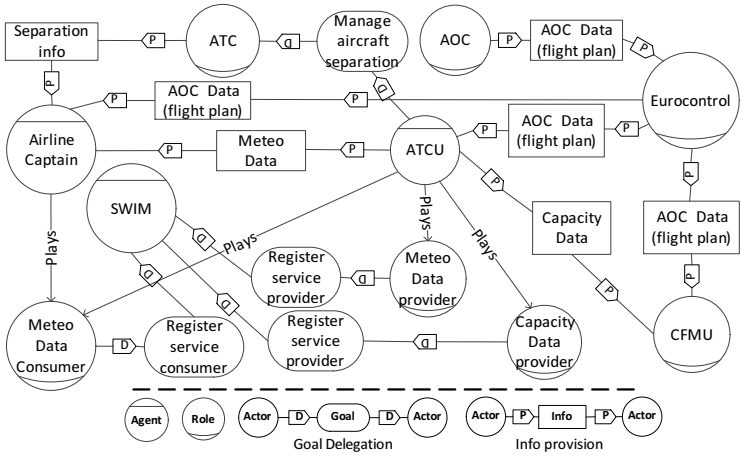

Fig. 1. The main actors, goals and information in the ATM system

(2) it can be acquired from objects (e.g., an ATCs is able to acquire information about the airplane position by visual observation); (3) it can be acquired from documents; and (4) it can be acquired by communication. We define information producer as following:

Definition 1 (Information Producer). *AOC is responsible for dispatching AOC data (flight plan). Thus, AOC is the producer of the AOC data.*

Example 1. AOC is responsible for dispatching AOC data (i.e. prepare flight plans). Thus, *AOC* is the producer of the AOC data.

Usually, information is produced for a certain reason, i.e., to be consumed for a certain purpose. We call the actor(s) who consume(s) information as information consumer(s), which can be defined as following:

Definition 2 (Information Consumer). *Information Consumer is the actor who consumes information, i.e., information is within its objective, and it is considered as the final destination of information.*

Example 2. The *Airplane captain* and *EUROCONTROL* both are AOC data (flight plan) consumers, since AOC data is within their objectives.

3.2 Information Integrity Dimensions

Information integrity is defined as the representational faithfulness of information to the true state of the object that the information represents [5]. Several studies have demonstrated that information integrity is a multi-dimensional concept [2,6,5] that can be characterized using multiple attributes, or dimensions. For instance, Bovee et al. [6] defined four attributes to evaluate information integrity: accuracy, completeness, consistency and existence. In the work of Boritz

[5], information integrity can be evaluated based on four attributes: completeness, currency/timeliness, accuracy/correctness and validity/authorization.

Following [6] the three core integrity dimensions are defined as follows: **(1) Information accuracy**: means that information should be true or error free with respect to some known, designated or measured value. **(2) Information completeness**: refers to having all required parts of an entity's information presented, and **(3) Information consistency**: requires that multiple recordings of the value(s) for an entity's attribute(s) be the same across time or space. For example, let's consider a flight plan that usually should include: 1- Type, 2- Aircraft Identification, 3- Aircraft Type, 4- Departure Point, etc. We say that the flight plan is accurate, if all information items included in it are accurate. We say it is complete, if it contains all information items composing it. Finally, we say it is consistent if all information items composing it are consistent.

It is well known that not all the goals have the same criticality to the organization performance, and preserving information integrity is not free. Thus, information integrity requirements will be determined based on the criticality of the consuming goal. In the next section, we introduce two concepts to determine when information integrity requirements is needed, namely, critical goal, and critical information (its integrity has been preserved).

3.3 Critical Information and Critical Goal

Within the organizational context, the failure of some goals may lead to serious problems for the organization, while the failure of other goals can be handled. Thus, not all goals have the same criticality to the organization. In this paper, we call these goals critical goals and they are defined as following:

Definition 3 (Critical Goal). *A critical goal is any goal that its failure may lead to major problems to the organization, and they can be used to represent the stakeholders critical objectives.*

Moreover, if the critical goal is decomposed into sub-goals (And/ Or decomposition) the criticality is propagated to its sub-goals, i.e., all its sub-goals are critical as well. Furthermore, compromised information might be a main reason of critical goals failure if they consume it. Thus, in ordered to guarantee that a critical goal will be satisfied properly, we should guarantee that the integrity of information consumed by it is preserved. To this end, we call information consumed by critical goals critical information and it is defined as follows:

Definition 4 (Critical Information). *Critical information is any information that its integrity should be preserved at any given time.*

Example 3. Airline captain has to satisfy the critical goal "manage gate-to-gate safe flight", which is decomposed into 8 sub-goals (And-decomposition), each one of these goals is a critical goal (the criticality is propagated from the parent goal), and consumes critical information.

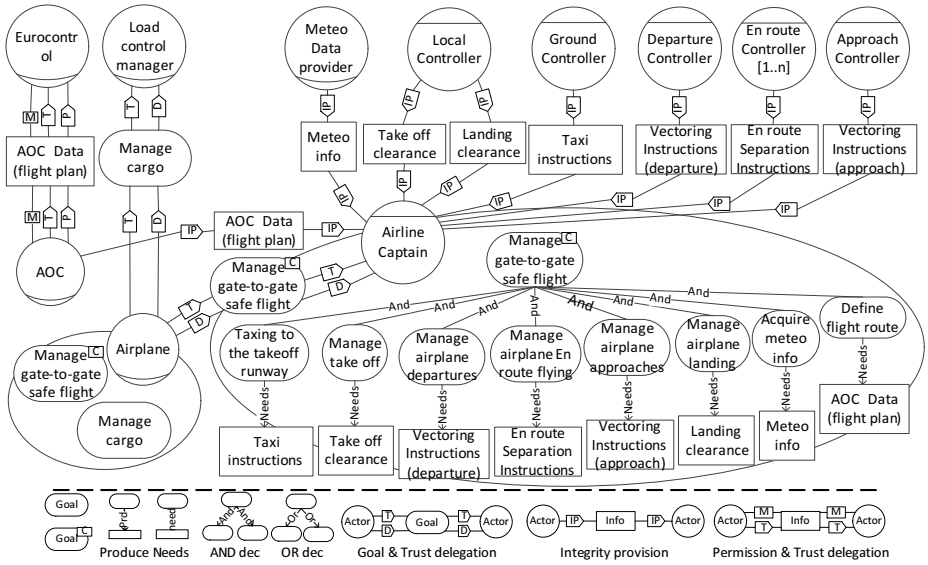

Fig. 2. A partial goal model concerning the ATM scenario

Information might lose its integrity during the provision process, thus, beside the normal provision that just consider information exchange between two actors, we define integrity provision that can preserve the integrity of the provided information, which will be used for the critical information provision and it can be defined as following:

Definition 5 (Integrity provision (I-Provision)). *Integrity provision describes information provision between two parties in which the integrity of the provided information is preserved.*

Example 4. The ground controller has to provide "taxi instructions" to the airline captain by integrity provision, since it is a critical information and its integrity has to be preserved during the provision.

Trust is a very important factor in almost every organization, since it may increase efficiency and save costs only if it is used efficiently (e.g. facilitate coordinated actions) [4]. Usually, trust is defined as a ternary relation between two parties (trustor, trustee) for the object of this trust (trustum). Trust is still a new research thread within the requirement engineering (RE) area. Several RE researchers suggested concepts for capturing trust relationships (e.g. [15]), but no previous work according to our knowledge, has studied the role of trust in enhancing information integrity. In this paper, we discuss the effect of trust over goal and permission delegation, since it represents the mental counter-part of delegation [9]. Therefore, trust plays an important role in deciding to delegate or not.

Table 1. General predicate

$needs(needer : n^1, info : i)$	$wants(actor : a, goal : g)$
$produce(goal : g, info : i)$	$modify(goal : g, info : i)$
$has(actor : a, info : i)$	$producer(actor : a, info : i)$
$consumer(actor : a, info : i)$	$provide(actor : a, actor : b, info : i)$
$prvChain(actor : a, actor : b, info : i)$	$ip\_provide(actor : a, actor : b, info : i)$
$ip\_prvChain(actor : a, actor : b, info : i)$	$can\_prv(actor : a, info : i)$
$can\_sat(actor : a, goal : g)$	$is\_responsible(actor : a, goal : g)$
$critical\_goal(goal : g)$	$critical\_info(info : i)$
$critical\_consumer(actor : a, info : i)$	
$and\_dec(goal : g, goal : g_1, goal : g_2)$	$or\_dec(goal : g, goal : g_1, goal : g_2)$
$delegate(actor : a, actor : b, goal : g)$	$delChain(actor : a, actor : b, goal : g)$
$have\_perm(type : t, actor : a, infoi)$	$grant\_perm(type : t, actor : a, actor : b, infoi)$
$have\_grant\_perm(type : t, actor : a, infoi)$	$grant\_grant\_perm(type : t, actor : a, actor : b, infoi)$
$need\_to\_have\_perm(type : t, actor : a, infoi)$	
$trust(actor : a, actor : b, goal : g)$	$trustChian(actor : a, actor : b, goal : g)$
$trust\_perm(type : t, actor : a, actor : b, info : i)$	$trust\_permChian(type : t, actor : a, actor : b, info : i)$
$trust\_grant\_perm(type : t, actor : a, actor : b, info : i)$	$trust\_grant\_permChain(type : t, actor : a, actor : b, info : i)$
$satisfied(actor : a, goal : g)$	$consumed(actor : a, info : i)$
$critical\_consumed(actor : a, info : i)$	$integrity\_preserved(actor : a, info : i)$
$integrity\_compromised(actor : a, info : i)$	

$^1 needer : n \in \{actor : a, goal : g\}$

Example 5. Airplane delegates "Manage gate-to-gate safe flight" to the *Airplane captain*, in this case, it is mandatory to have a trust relation along with the goal delegation to be sure that the goal well be satisfied. Furthermore, *AOC* delegates modification permission over "AOC data" to *EUROCONTROL*, similarly, a trust relation has to exist to guarantee that the permissions will not be misused.

Figure 2 shows a partial goal model concerning the ATM scenario, in which the *AOC* provide "AOC data" to both *EUROCONTROL* and the *Airline captain*, and delegates modification permission over it along with a trust relation concerning this delegation to *EUROCONTROL*. Furthermore, *Airplane* has 2 goals, it delegates them along with trust relation to *Load control manager*, and *Airline captain*. "Manage gate-to-gate safe flight" is a critical goal and it is decomposed into 8 sub-goals each one of them is critical goal as well, and consumes critical information that has to be provided through integrity provision (e.g. AOC data).

4 Formalizing Information Integrity

We use Datalog [1] as the underlying semantic framework. Table 1 introduces the model general predicates, for example predicates prvChain(a, b, i) and ip_prvChain(a, b, i) hold if there is a valid provision chain / integrity provision chain between actor a and actor b concerning information i.

Table 2 list the general axioms of a model. O1-6 deals with actor's objectives; for example, O1 express how a goal needs became an actors need. O2-5 are used to derive and assign new objectives (sub-goals) to the actor responsible

Table 2. General axioms

O1 $needs(A,I) \leftarrow is\_responsible(A,G) \wedge needs(G,I)$	O2 $wants(A,G_1) \leftarrow and\_dec(G,G_1,G_2) \wedge wants(A,G)$
O3 $wants(A,G_2) \leftarrow and\_dec(G,G_1,G_2) \wedge wants(A,G)$	O4 $wants(A,G_1) \leftarrow or\_dec(G,G_1,G_2) \wedge wants(A,G)$
O5 $wants(A,G_2) \leftarrow or\_dec(G,G_1,G_2) \wedge wants(A,G)$	O6 $wants(A,G) \leftarrow delChain(B,A,G)$
C1 $producer(A,I) \leftarrow can\_sat(A,G) \wedge produce(G,I)$	C2 $consumer(A,I) \leftarrow needs(A,I)$
C3 $has(A,I) \leftarrow producer(A,I)$	C4 $has(A,I) \leftarrow provide(B,A,I) \wedge can\_prv(B,I)$
C5 $has(A,I) \leftarrow ip\_provide(B,A,I) \wedge can\_prv(B,I)$	C6 $can\_prv(A,I) \leftarrow has(A,I)$
C7 $can\_prv(A,I) \leftarrow prvChain(A,B,I) \wedge has(B,I)$	C8 $can\_prv(A,I) \leftarrow ip\_prvChain(A,B,I) \wedge has(B,I)$
C9 $prvChain(A,B,I) \leftarrow provide(A,B,I)$	C10 $prvChain(A,B,I) \leftarrow \begin{cases} prvChain(A,C,I) \\ \wedge\, prvChain(C,B,I) \end{cases}$
C11 $ip\_prvChain(A,B,I) \leftarrow ip\_provide(A,B,I)$	C12 $ip\_prvChain(A,B,I) \leftarrow \begin{cases} ip\_prvChain(A,C,I) \\ \wedge\, ip\_prvChain(C,B,I) \end{cases}$
C13 $is\_responsible(A,G) \leftarrow wants(A,G) \wedge can\_sat(A,G)$	C14 $can\_sat(A,G) \leftarrow delChain(A,B,G) \wedge can\_sat(B,G)$
C15 $can\_sat(A,G) \leftarrow needs(G,I) \wedge has(A,I)$	C16 $can\_sat(A,G) \leftarrow \begin{cases} and\_dec(G,G_1,G_2) \wedge \\ can\_sat(A,G_1) \wedge can\_sat(A,G_2) \end{cases}$
C17 $can\_sat(A,G) \leftarrow or\_dec(G,G_1,G_2) \wedge can\_sat(A,G_1)$	C18 $can\_sat(A,G) \leftarrow or\_dec(G,G_1,G_2) \wedge can\_sat(A,G_2)$
D1 $delChain(A,B,G) \leftarrow delegate(A,B,G)$	D2 $delChain(A,B,G) \leftarrow \begin{cases} delChain(A,C,G) \\ \wedge\, delChain(C,B,G) \end{cases}$
CG1 $critical\_goal(G_1) \leftarrow \begin{cases} and\_decom(G,G_1,G_2) \\ \wedge\, critical\_goal(G) \end{cases}$	CG2 $critical\_goal(G_2) \leftarrow \begin{cases} and\_decom(G,G_1,G_2) \\ \wedge\, critical\_goal(G) \end{cases}$
CG3 $critical\_goal(G_1) \leftarrow \begin{cases} or\_decom(G,G_1,G_2) \\ \wedge\, critical\_goal(G) \end{cases}$	CG4 $critical\_goal(G_2) \leftarrow \begin{cases} or\_decom(G,G_1,G_2) \\ \wedge\, critical\_goal(G) \end{cases}$
CG5 $critical\_info(I) \leftarrow \begin{cases} need(G,I) \wedge \\ critical\_goal(G) \end{cases}$	CG6 $critical\_cunsumer(A,I) \leftarrow \begin{cases} is\_responsible(A,G) \wedge \\ critical\_goal(G) \wedge need(G,I) \end{cases}$

of satisfying the parent goal. O6 is used to show a goal delegation between two actors. Furthermore, C1-18 deals with actor's capabilities; for example, C3-5 represent the actor capability of having information, and C6-8 are used for actors' capabilities related to information provision; while C14-18 can be used to express the actors' capabilities concerning goals satisfying. Finally, CG1-4 show the criticality propagation from a goal to its sub-goals, and CG5 and CG6 are used to capture critical information and critical consumer respectively.

Example 6. Airplane delegates "Manage gate-to-gate safe flight" goal to the *Airline captain* (D1, D2), and it became an objective of the *Airline captain* (O6), along with each of its sub-goals (O2, O3). Furthermore, *Airline captain* is responsible of manage takeoff (C 13), and he needs "takeoff clearance" (O1). Thus, *Airline captain* depends on *local controller* to provide it since the *local controller* is the producer (C1), and he is able to provide it (C3). Finally, "Manage takeoff" is a critical goal, since it is a sub-goal of a critical goal (CG1, CG2), and "takeoff clearance" is critical information (CG5).

In table 3, P1 show how an actor can have permission T over information I, P2 and P3 show how an actor can have the grant permission, and P4 describes who need to have modification permission. Axioms T1 and T2 describe a trust chain between two actors concerning a goal satisfying, and T3 and T4 describe a trust chain between for a granted permission. T5 and T6 describe a valid trust chain between two actors concerning a grant permission type T over info I.

Example 7. AOC have grant permission over "AOC data" since it is the producer (P2, P3), and he can delegate this permission (P1) to *EUROCONTROL* since it needs such permission (P4). To ensure that *EUROCONTROL* will not

Table 3. Permission and Trust axioms

P1 $have\_perm(T,A,I) \leftarrow \begin{cases} grant\_perm(T,A,B,I) \\ \wedge\, have\_grant\_perm(T,A,I) \end{cases}$
P2 $have\_grant\_perm(T,A,I) \leftarrow producer(A,I)$
P3 $have\_grant\_perm(T,A,I) \leftarrow \begin{cases} grant\_grant\_perm(T,B,A,I) \\ \wedge\, have\_grant\_perm(T,B,I) \end{cases}$
P4 $need\_to\_have\_perm(m,A,I) \leftarrow is\_responsible(A,G) \wedge modify(G,I)$
T1 $trustChain(A,B,G) \leftarrow trust(A,B,G)$
T2 $trustChain(A,B,G) \leftarrow trustChain(A,C,G) \wedge trustChain(C,B,G)$
T3 $trust\_permChain(T,A,B,I) \leftarrow trust\_perm(T,A,B,I)$
T4 $trust\_permChain(T,A,B,I) \leftarrow \begin{cases} trust\_permChain(T,A,C,I) \\ \wedge\, trust\_permChain(T,C,B,I) \end{cases}$
T5 $trust\_chain\_grant\_perm(T,A,B,I) \leftarrow trust\_grant\_perm(T,A,B,I)$
T6 $trust\_grant\_permChain(T,A,B,I) \leftarrow \begin{cases} trust\_grant\_permChain(T,A,C,I) \\ \wedge\, trust\_grant\_permChain(T,C,B,I) \end{cases}$

misuse this permission a trust relation coercing this delegation exist (T3, T4). Moreover, a trust chain exist between *Airplane* and *Load control manager* concerning the goal "Manage cargo" (T1, T2).

In table 4 A1-5 list situations where an actor believes his goal will be satisfied, and A6-7 list situations where information consumer believes information it required will be provided. A8 shows when an actor believes that the integrity of critical information it consumes is preserved, and A9-11 list situations in which information consumer is not sure that the integrity of information he will consume is preserved.

Example 8. Airplane captain believes "define flight route" will be satisfied since he is responsible of it and he has the capability to satisfy it (A1), and *Airplane* believes "define flight route" will be satisfied since it is delegated to *Airplane captain* (has capability) and there is a trust relation concerning this delegation (A2). *Airplane captain* believes that "Manage gate-to-gate safe flight" will be satisfied because he is able to satisfy all its sub-goals (A3). *Airplane captain* believes that "AOC data integrity has been preserved (A8), since it was provided by integrity provision chain (A9), and no permission chain concerning this information is delegated without a trust relation chain to any actor (A10-11).

5 Analyzing Information integrity

We use the DLV system [2] to analyze information integrity requirements, we define a set of properties that will be used to verify the correctness of the model. These properties define constraints that designers should consider during the system design to satisfy the stakeholders requirements.

In table 5, Pro1 states that an actor cannot provide information unless he has the required provision capability; this property will allow the model to detect

[2] http://www.dbai.tuwien.ac.at/proj/dlv

Table 4. Achieved Actors Objectives

A1 $satisfied(A,G) \leftarrow is\_responsible(A,G)$
A2 $satisfied(A,G) \leftarrow \begin{cases} delChain(A,B,G) \wedge trust\_chain(A,B,G) \\ \wedge satisfied(B,G) \end{cases}$
A3 $satisfied(A,G) \leftarrow \begin{cases} and\_dec(G,G_1,G_2) \wedge \\ satisfied(A,G_1) \wedge satisfied(A,G_2) \end{cases}$
A4 $satisfied(A,G) \leftarrow or\_dec(G,G_1,G_2) \wedge satisfied(A,G_1)$
A5 $satisfied(A,G) \leftarrow or\_dec(G,G_1,G_2) \wedge satisfied(A,G_2)$
A6 $consumed(A,I) \leftarrow has(A,I)$
A7 $consumed(A,I) \leftarrow prvChain(A,B,I) \wedge can\_prv(B,I)$
A8 $critical\_consumed(A,I) \leftarrow ip\_prvChain(A,B,I) \wedge can\_prv(B,I)$
A9 $integrity\_preserved(A,I) \leftarrow \begin{cases} critical\_cunsumer(A,I) \\ \wedge not\, integrity\_compromised(A,I) \end{cases}$
A10 $integrity\_compromised(A,I) \leftarrow prvChain(B,A,I)$
A11 $integrity\_compromised(A,I) \leftarrow \begin{cases} have\_perm(modify,B,I) \\ \wedge trust\_perm(modify,A,B,I) \end{cases}$
A12 $integrity\_compromised(A,I) \leftarrow \begin{cases} have\_grant\_perm(modify,B,I) \\ \wedge trust\_grant\_perm(modify,A,B,I) \end{cases}$

any invalid information provision / information provision chain. Pro2 states that information should not be provided to any actor unless its entitlements require it; this property allows the model to detect any unrequired information provision that may threaten the integrity of information. For example, *AOC* does not have a provision capability concerning meteo information (Pro1), since it is not a meteo producer, and such information will not be provided to it because none of his entitlements requires it (Pro2). Thus, if such provision exists the model will mark it as invalid provision. Depending on Pro3 the model will detect any situation in which there is no valid trust chain along with a goal delegation chain, which may endanger the satisfaction of the delegated goal. For example, *Airplane* delegates "Manage gate-to-gate safe flight" to *Airplane captain*, the model is able to detect and notify the designer if there is no trust chain concerning this goal delegation.

Pro 4 states that permissions should be delegated to any actor require them. While Pro 5 states that no actor should have permissions unless its entitlements require these permissions. For example, *Airline captain* does not need to modify the "AOC data", thus, such permissions should not be delegated to him. While *EUROCONTROL* should have such permissions since it may need to modify the "AOC data". Similarly, Pro 6 and Pro 7 are specialized for grant permissions. Pro 4-7 enable the model to control the permission delegation properly by 1- detecting situations in which permission is not delegated to actors require them, which may prevent them from performing their duties properly; and 2- detecting situations in which permissions are delegated to actors even their entitlements does not require such permissions, which may threaten the integrity of information.

Finally, Pro 8 states that the integrity of critical information should be preserved from the perspective of its consumer. This enables the model to detect

Table 5. Properties of the Design

Pro 1 :- $provide(B, A, I), not\, can\_prv(B, I)$
Pro 2 :- $provide(B, A, I), not\, wants(A, I)$
Pro 3 :- $delChain(A, B, G), not\, trustChain(A, B, G)$
Pro 4 :- $need\_to\_have\_perm(T, A, I), not\, have\_perm(T, A, I)$
Pro 5 :- $have\_perm(T, A, I), not\, need\_to\_have\_perm(T, A, I)$
Pro 6 :- $need\_to\_have\_grant\_perm, not\, have\_grant\_perm(T, A, I)$
Pro 7 :- $have\_grant\_perm, not\, need\_to\_have\_grant\_perm(T, A, I)$
Pro 8 :- $critical\_cunsumer(A, I), not\, integrity\_preserved(A, I)$

any situation that may compromise the integrity of the critical information. For example, the *Airline captain* believes that the integrity of "AOC data" he consumes has been preserved, if it was not compromised by any of the situations listed in table 4 (A9-11).

6 Related Work and Conclusions

Requirement engineering community suggests concepts for capturing and analyzing the high-level security aspects of the system-to-be, but they did not appropriately support the modeling and analyzing of information integrity. For instance, misuse case [22] provide high level modeling constructs to capture threads to the system assets, but offer no special primitives for modeling nor reasoning about information integrity. In UMLsec [12], integrity was modeled as constraints that can restrict unwanted modification. SecureUML [3] was mainly developed to model access control policies, which can protect information integrity to a certain level. Abuse frame [13] addresses the integrity problem (modification) by preventing unauthorized actors from modifying information or prevent authorized actors from doing unauthorized modifications. The main limitation in the previously mentioned languages that they do not consider the social relation among the actors of the system-to-be (e.g. delegation and trust); even social relations play a major role in any socio-technical system.

Information integrity modeling is an extension of traditional information modeling, where information modeling captures the structure and semantics of information, while information integrity modeling captures structural and semantic issues underlying information integrity. In this paper, we argued that any solution to enhance information integrity should be first considered at the organizational level. We proposed an extension of i*/Tropos modeling language to capture information integrity requirements within the organizational environment, and provide a reasoning framework to check properties of the design over models built using this language.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*, volume 8. Citeseer, 1995.

2. D.P. Ballou and G.K. Tayi. Methodology for allocating resources for data quality enhancement. *Communications of the ACM*, 32(3):320–329, 1989.
3. 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.
4. K. Blomqvist and P. Ståhle. Building organizational trust. In *Proceedings of 16th Annual IMP Conference*. Citeseer, 2000.
5. J.E. Boritz. Is practitioners' views on core concepts of information integrity. *International Journal of Accounting Information Systems*, 6(4):260–279, 2005.
6. 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*, 18(1):51–74, 2003.
7. 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.
8. M.K. Buckland. Information as thing. *Journal of the American Society for information science*, 42(5):351–360, 1991.
9. 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.
10. F. Cohen. A cryptographic checksum for integrity protection. *Computers & Security*, 6(6):505–510, 1987.
11. Flight Safety Foundation. Accident description. http://aviation-safety.net/database/record.php?id=19961112-1 /, March 1996.
12. J. Jürjens. *Secure systems development with UML*. Springer Verlag, 2005.
13. L. Lin, B. Nuseibeh, D. Ince, M. Jackson, and J. Moffett. Introducing abuse frames for analysing security requirements. 2003.
14. A. Motro. Integrity= validity+ completeness. *ACM Transactions on Database Systems (TODS)*, 14(4):480–502, 1989.
15. 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.
16. D.A. Patterson, G. Gibson, and R.H. Katz. *A case for redundant arrays of inexpensive disks (RAID)*, volume 17. ACM, 1988.
17. 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.
18. T.C. Redman. Improve data quality for competitive advantage. *Sloan Management Review*, 36:99–99, 1995.
19. T.C. Redman. The impact of poor data quality on the typical enterprise. *Communications of the ACM*, 41(2):79–82, 1998.
20. National Transportation safety board. Aircraft accident report file no. 1-0017. http://libraryonline.erau.edu/online-full-text/ntsb/aircraft-accident-reports/AAR73-15.pdf /, July 1972.
21. SESAR. Swim concept of operations, deliverable a01, edition 00.02.00., 2011.
22. G. Sindre and A.L. Opdahl. Eliciting security requirements by misuse cases. In *Technology of Object-Oriented Languages and Systems, 2000. TOOLS-Pacific 2000. Proceedings. 37th International Conference on*, pages 120–131. IEEE, 2000.
23. Eric Siu-Kwong Yu. *Modelling strategic relationships for process reengineering*. Ph.d. thesis, Toronto, Ont., Canada, Canada, 1996. AAINN02887.