Modeling and Analyzing Information Integrity in Safety Critical Systems

Mohamad Gharib and Paolo Giorgini

DISI, University of Trento,
Italy, Trento, Via Sommarive 14
gharib@disi.unitn.it, paolo.giorgini@unitn.it

Abstract. Preserving information integrity represent an urgent need for safety critical systems, where depending on incorrect or inconsistent information may leads to disasters. Typically, information integrity is a problem handled at technical level (e.g., checksumming). However, information integrity has to be analyzed in the social-technical context of the system, since information integrity related problems might manifest themselves in the business processes and actors interactions. In this paper, we propose an extended version of *i**/ secure Tropos modeling languages to capture 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. Air Traffic Management (ATM) case study is used throughout the paper.

Keywords: Integrity, Security, Modeling, Analysing.

1 Introduction

Information integrity is surely one essential feature of any safety critical system, where incorrect or inconsistent information may produce disaster and loss of humans lives. Information integrity is traditionally considered in the way information are stored and transmitted, several techniques and solutions have been proposed (e.g., checksum [6]). However, such solutions do not solve problems that may rise at business and organizational level. So for example, in the Chicago-O'Hare runway collision [10], the *Air Traffic Controller* (ATCs) failed to notify the *Airplane Captain* that one of the runways was being used by another airplane, which results in inconsistent information within the system that lead to the runway collision, such problem cannot be handled by the available technical solutions since the problem (inconsistency) manifested itself in the loss business process.

Information integrity should be always considered as a socio-technical problem, where socio and organizational aspects of the system have to be considered along with the technical solutions, i.e., the critical system design has always to go through a socio-technical analysis, where needs about information integrity have to be identified with respect to all involved social and technical actors, their local and global objectives and their mutual interactions. In this paper, we extend *i*/* secure Tropos modeling languages with the required concepts and primitives that enable us to capture and analyze the information integrity requirements. The paper is organized as follows, ATM case study in section (§2). Section (§3) discusses modeling information integrity requirements, while section (§4) presents the Datalog formalization. Section (§5) discuss the properties of the design, related work in section (§6), finally, conclusion and future work are presented at section (§7).

2 Case Study

Our case study concerns Air Traffic Management (ATM), which is a safety-critical system since its failures may result in loss of humans lives. An ATM system is managed by Air Traffic Control Center (ACC) that provides air traffic control (ATC) services, which are provided by ground-based controllers (ATCs) working at different Air Traffic Control Unit (ATCU) with the main purpose of preventing aircraft collisions by providing airplanes with the required separation information. Information handled by ATM, includes but not limited to, flight plans, separation data, and meteo data. For instance, any airlines company want to fly in Europe has to send its flight plan to EUROCONTROL where it is checked (might be modified) and send out to all ATCU that is affected by the flight. Furthermore ATCs, exchange information concerning the traffic flow in their sectors, and they have to provide airplanes with separation data during the different phases of the flight.

In such systems, information (e.g., separation data) might be used by several actors (e.g., ATCs) and some of those actors might modify it for different reasons. The system should be able to maintain (preserve) the integrity of information among all its users at any time to prevent any disasters that might happen due to information integrity related problems. For example we show how modification permissions should be delegated based on some criteria that can be captured only by the social aspects of the system (e.g., trust).

3 Modeling Information Integrity Requirements

In this section, we discuss information[1] and its integrity dimensions to get better understanding of what information integrity requirements are and when they are needed. Information can be produced in several ways by information producers that represent its initial source, and we call the actor(s) who consume(s) information as information consumer(s). For example, *Airline company* is the producer of "flight plan", and *Airplane captain* is the "flight plan" consumer. While information integrity is defined as the representational faithfulness of information to the true state of the object that the information represents [3]. Information integrity is a multi-dimensional concept [4,3] that can be characterized using multiple dimensions: accuracy, completeness and consistency. While integrity requirements means preserving these 3 dimensions.

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

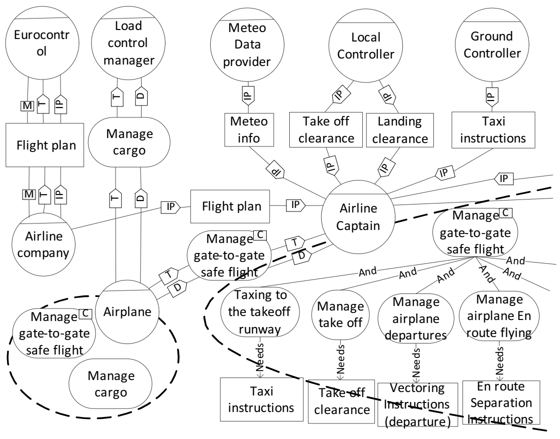

Fig. 1. A partial goal model concerning the ATM scenario

Within the organizational context not all the goals have the same criticality, compromised information might be a main reason for a goal failure and preserving information integrity is not free. The information integrity requirements will be determined based on the criticality of the consuming goal. Thus, we introduce two new concepts, namely, critical goal and critical information. The first is used to represent the stakeholders critical goals, while the second is information consumed by critical goals and its integrity should be preserved at any given time. For example, "manage gate-to-gate safe flight" is a critical goal and it is decomposed into several sub-goals, each one of them is a critical goal and consumes critical information.

Information might lose its integrity during the provision process. Thus, beside the normal provision (P), we define integrity provision (I-Provision) that can preserve the integrity of the provided information. Furthermore, we discuss the effect of trust over goals and permissions delegation, since it represents the mental counter-part of delegation [5]. For example, *Airlines company* delegates modification permission concerning "flight plan" to *EUROCONTROL* a trust relation has to exist to guarantee that the integrity of "flight plan" will be preserve. Figure 1 shows a partial goal model concerning the ATM scenario, in which the *Airline company* provide (I-Provision) " flight plan" to both *EUROCONTROL* and the *Airline captain*, and delegates modification permission over it along with a trust relation to *EUROCONTROL*. *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 critical and it is decomposed into several sub-goals each of them is critical as well.

Table 1. General predicate

$needs(needer : n, info : i)$, $needer \in \{actor, goal\}$	$wants(actor : a, goal : g)$
$produce(goal : g, info : i)$	$modify(goal : g, info : i)$
$has(actor : a, info : i)$	$producer(actor : a, info : i)$
$consumer(actor : a, info : i)$	$provide(actor : a, actor : b, info : i)$
$prvChain(actor : a, actor : b, info : i)$	$ip\_provide(actor : a, actor : b, info : i)$
$ip\_prvChain(actor : a, actor : b, info : i)$	$can\_prv(actor : a, info : i)$
$can\_sat(actor : a, goal : g)$	$is\_responsible(actor : a, goal : g)$
$and\_dec(goal : g, goal : \mathrm{g}_1, goal : \mathrm{g}_2)$	$or\_dec(goal : g, goal : \mathrm{g}_1, goal : \mathrm{g}_2)$
$delegate(actor : a, actor : b, goal : g)$	$delChain(actor : a, actor : b, goal : g)$
$critical\_goal(goal : g)$	$critical\_info(info : i)$
$critical\_consumer(actor : a, info : i)$	$need\_perm(type : t, actor : a, info : i)$
$have\_perm(type : t, actor : a, info : i)$	$grant\_perm(type : t, actor : a, actor : b, info : i)$
$have\_grant\_perm(type : t, actor : a, info : i)$	$grant\_grant\_perm(type : t, actor : a, actor : b, info : i)$
$trust(actor : a, actor : b, goal : g)$	$tChian(actor : a, actor : b, goal : g)$
$trust\_perm(type : t, actor : a, actor : b, info : i)$	$trust\_pChian(type : t, actor : a, actor : b, info : i)$
$trust\_grant\_perm(type : t, actor : a, actor : b, info : i)$	$trust\_grant\_pChain(type : t, actor : a, actor : b, info : i)$
$satisfied(actor : a, goal : g)$	$consumed(actor : a, info : i)$
$critical\_consumed(actor : a, info : i)$	$integ\_preserved(actor : a, info : i)$
$integ\_comp(actor : a, info : i)$	

4 Formalizing Information Integrity Requirements

We use Datalog [1] as the underlying semantic framework. Table 1 introduces the general predicates. Table 2 lists the general axioms. For example, O1 expresses how a goal needs became an actors need, D1-2 for goal delegation, C3-5 an actor capabilities of having information, and C6-8 information provision capabilities, while C14-18 goals satisfying capabilities, CG1-2 criticality propagation and CG3-4 critical information and consumer. In table 3, P1 for having permission, P2-3 for granting permission. T1-2 trust concerning a goal satisfying. A1-4 actor believes its goal will be satisfied, and A5-6 consumer believes its information will be provided. A7 actor believes integrity is preserved, A8-10 where integrity might be compromised.

Table 2. General axioms

O1 $needs(A, I) \leftarrow is\_responsible(A, G) \wedge needs(G, I)$	O2 $wants(A, G) \leftarrow delChain(B, A, G)$
O3 $wants(A, G_{1[2]}) \leftarrow and\_dec(G, G_1, G_2) \wedge wants(A, G)$	O4 $wants(A, G_{1[2]}) \leftarrow or\_dec(G, G_1, G_2) \wedge wants(A, G)$
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 $critical\_goal(G_{1[2]}) \leftarrow \begin{cases} and\_decom(G, G_1, G_2) \\ \wedge\, critical\_goal(G) \end{cases}$	CG2 $critical\_goal(G_{1[2]}) \leftarrow \begin{cases} or\_decom(G, G_1, G_2) \\ \wedge\, critical\_goal(G) \end{cases}$
CG3 $critical\_info(I) \leftarrow \begin{cases} needs(G, I) \wedge \\ critical\_goal(G) \end{cases}$	CG4 $critical\_cunsumer(A, I) \leftarrow \begin{cases} is\_responsible(A, G) \wedge \\ critical\_goal(G) \wedge needs(G, I) \end{cases}$

Table 3. Permission, Trust axioms and Achieved Actors Objectives

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 $need\_perm(T,A,I) \leftarrow \begin{cases} is\_responsible(A,G) \\ \wedge\, modify(G,I) \end{cases}$
T1 $tChain(A,B,G) \leftarrow trust(A,B,G)$	T2 $tChain(A,B,G) \leftarrow \begin{cases} tChain(A,C,G) \\ \wedge\, tChain(C,B,G) \end{cases}$
T3 $trust\_pChain(T,A,B,I) \leftarrow trust\_perm(T,A,B,I)$	T4 $trust\_pChain(T,A,B,I) \leftarrow \begin{cases} trust\_pChain(T,A,C,I) \\ \wedge\, trust\_pChain(T,C,B,I) \end{cases}$
T5 $trust\_grant\_pChain(T,A,B,I) \leftarrow trust\_grant\_perm(T,A,B,I)$	T6 $trust\_grant\_pChain(T,A,B,I) \leftarrow \begin{cases} trust\_grant\_pChain(T,A,C,I) \\ \wedge\, trust\_grant\_pChain(T,C,B,I) \end{cases}$
A1 $satisfied(A,G) \leftarrow is\_responsible(A,G)$	A2 $satisfied(A,G) \leftarrow \begin{cases} delChain(A,B,G) \wedge tChain(A,B,G) \\ \wedge\, satisfied(B,G) \end{cases}$
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 $satisfied(A,G) \leftarrow or\_dec(G,G_1,G_2) \wedge satisfied(A,G_{1[2]})$
A5 $consumed(A,I) \leftarrow has(A,I)$	A6 $consumed(A,I) \leftarrow prvChain(A,B,I) \wedge can\_prv(B,I)$
A7 $integ\_preserved(A,I) \leftarrow \begin{cases} critical\_cunsumer(A,I) \\ \wedge\, not\, integ\_comp(A,I) \end{cases}$	A8 $integ\_comp(A,I) \leftarrow prvChain(B,A,I)$
A9 $integ\_comp(A,I) \leftarrow \begin{cases} have\_perm(modify,B,I) \\ \wedge\, trust\_perm(modify,A,B,I) \end{cases}$	A10 $integ\_comp(A,I) \leftarrow \begin{cases} have\_grant\_perm(modify,B,I) \\ \wedge\, trust\_grant\_perm(modify,A,B,I) \end{cases}$

Table 4. 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 :- $delChain(A,B,G), not\, tChain(A,B,G)$	Pro 4 :- $need\_perm(T,A,I), not\, have\_perm(T,A,I)$
Pro 5 :- $have\_perm(T,A,I), not\, need\_perm(T,A,I)$	Pro 6 :- $need\_perm, not\, have\_grant\_perm(T,A,I)$
Pro 7 :- $have\_grant\_perm, not\, need\_perm(T,A,I)$	Pro 8 :- $critical\_cunsumer(A,I), not\, integ\_preserved(A,I)$

5 Analyzing Information Integrity Requirements

We use the DLV system[2] to analyze information integrity requirements, we define a set of properties that is used to verify the correctness of the model. In table 4, Pro1 allows the model to detect any invalid information provision / chain, Pro2 allows the model to detect any unneeded information provision that may threaten information integrity. For example, *Airlines company* 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 (Pro2). 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. According to Pro 4 permissions should be delegated only to actors require them, and they should not be delegated unless the actors require them (Pro 5). For example, *Airline captain* should not have modify permission over "flight plan", and *EUROCONTROL* should have such permission. Similarly, Pro 6-7 are specialized for grant permissions. Finally, Pro 8 enables the model to detect situations that might compromise critical information integrity. For example, the *Airline captain* believes that "flight plan" integrity is preserved, if it was not compromised by situations listed in table 3 (A8-10).

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

6 Related Work

Requirements engineering community did not appropriately support the modeling and analyzing of information integrity requirements. For instance, abuse frames [8] addresses the integrity problem by preventing unauthorized actors from modifying information. In both UMLsec and SecureUML [7,2] integrity was modeled as constraints that can restrict unwanted modification. The main limitation in these languages is that they do not consider the social relation among actors of the system (e.g. delegation, trust). Secure Tropos [9] provides concepts for capturing security requirements (mainly privacy) but offers no primitives neither to capture nor analyze information integrity requirements.

7 Conclusions and Future Work

We extend i */ secure Tropos [11,9] with several concepts including critical information, critical goal, information producer and consumer. The first two concepts are used to determine were integrity requirements are needed; the third is used to control modification permission properly by information producer, while consumer is used to determine if integrity has been preserved at its final destination. The extended framework with all new concepts presented in this paper will be supported by RE-Tool. Furthermore, we will extend the language with concepts and primitives for capturing the related information integrity dimensions (accuracy, completeness and consistency).

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of databases, Citeseer, vol. 8 (1995)
2. Basin, D., Doser, J., Lodderstedt, T.: Model driven security: From UML models to access control infrastructures (TOSEM) 15, 39–91 (2006)
3. Boritz, J.: IS practitioners' views on core concepts of information integrity. International Journal of Accounting Information Systems 6, 260–279 (2005)
4. Bovee, M., Srivastava, R., Mak, B.: A conceptual framework and belief-function approach to assessing overall information quality. International Journal of Intelligent Systems, Wiley Online Library 18, 51–74 (2003)
5. Castelfranchi, C., Falcone, R.: Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. Multi Agent Systems (1998)
6. Cohen, F.: A cryptographic checksum for integrity protection Computers & Security. Elsevier 6, 505–510 (1987)
7. Jrjens, J.: Secure systems development with UML. Springer (2005)
8. Lin, L., Nuseibeh, B., Ince, D., Jackson, M., Moffett, J.: Introducing abuse frames for analysing security requirements. IEEE Computer Society (2003)
9. Mouratidis, H., Giorgini, P.: Secure Tropos: A security-oriented extension of the Tropos methodology. World Scientific Publishing, Singapore (2007)
10. safety board null, N. T. Aircraft accident report File No. 1-0017 (1972)
11. Yu, E.S.-K.: Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto (1996)