Goal-Oriented Security Requirements Analysis for a System Used in Several Different Activities

Haruhiko Kaiya[1], Takao Okubo[2], Nobuyuki Kanaya[2], Yuji Suzuki[1], Shinpei Ogata[1], Kenji Kaijiri[1], and Nobukazu Yoshioka[3]

[1] Dept. of Computer Science, Shinshu University, Nagano 380-8553, Japan
kaiya@shinshu-u.ac.jp
[2] Fujitsu Laboratory, Kawasaki, Kanagawa, 211–8588, Japan
[3] National Institue of Informatics (NII), Tokyo 101-8430, Japan

Abstract. Because an information system is used in different activities simultaneously today, we have to analyze usages of the system in the existing activities and to-be usages in an intended activity together. Especially, security aspects should be carefully analyzed because existing activities are not always secure. We propose a security requirements analysis method for resolving this problem. To take both existing and intended activities into account together, we integrate them on the basis of the unification of common actors. To explore possible attacks under integrated activities, we enumerate achievable attacks on the basis of the possible means in each actor with the help of security knowledge. To avoid or mitigate the attacks and to achieve fundamental goals, we disable some means or narrow down the means to be monitored with the help of propositional logic formulae. Through case studies on insurance business, we illustrated our idea.

Keywords: Goal-Oriented Requirements Analysis, Security Requirements Analysis, Strategic Dependency, Logic.

1 Introduction

When an information system is intended to be introduced into some activity such as business or amusements, we have to analyze the activity and define what kinds of functionalities and qualities including security needs are required for the system. In many cases, such a system or a part of it has already used in other activities. For example, we are using PC's, Web browsers, emails, social network services (SNS) and free software for our daily activities. Some business person cannot perform his/her business without own smart phones, and he/she also uses SNS for fun via the smart phones. We thus have to take such existing activities into account when an information system is intended to be introduced into an activity. Especially, we have to take them into account carefully for analyzing security requirements because not all existing activities are secure.

In this paper, we propose a method for analyzing security requirements for an information system used in several different activities. When actors and their dependencies in each activity are specified, we can clarify the common parts among different activities. Through such common parts, good or bad impacts are conveyed. We thus use i* notation [1,2] for specifying each activity. In i*, the needs of actors are represented as goals. Although some goals are fundamental for some actor and others are just subcontracts, there is no explicit distinction between fundamental goals and others in i*. We

X. Franch and P. Soffer (Eds.): CAiSE 2013 Workshops, LNBIP 148, pp. 478–489, 2013.

thus introduce this distinction because fundamental goals cannot be easily modified or given up but others can. Instead, we don't use original four goals types in i*. In usual goal-oriented modeling, a goal hierarchy is represented in a goal tree or a goal DAG (directed acyclic graph) so that lower goals satisfy their upper goals. In i*, such a goal hierarchy is used for specifying the rationale of each actor, and a proper actor will have his/her rationale so that lower goals satisfy their upper goals. However, this does not go for malicious actors. For example, some free software or web site sometimes contains a virus against its official functionalities. We thus introduce the concept "incorrect goal decomposition". To explore avoiding or mitigating security problems, we simply use propositional logic formulae because the rationale for each actor is represented in goal hierarchies and each of them can be represented in a propositional logic formula. Although existing i* variations could be used for analyzing security requirements for a system used in several activities, our method is simpler than others and even practitioners can accept it.

The rest of this paper is organized as follows. In the next section, we review related works including i* modeling language. In section 3, we explain our method in detail. In section 4, we apply our method to an insurance business to evaluate our method. Finally, we conclude our current results and show the future issues.

2 Related Work

A modeling language i* [1,2] is one of well-known goal-oriented requirements notations, and there already exists a lot of its security extensions [3], [4] [5], [6], [7]. When we develop and deploy a new information system, several human, organizations and existing systems will be related the news system. In i*, such human, organizations, existing systems and the new system are represented as actors. Because each actor has its goals and not all goals can be achieved by itself, an actor depends on another. Such dependencies are represented as strategic dependencies in i*. The strategic dependency is the most novel idea in i*, and most of all its variations such as Tropos [8] contain the idea. In i*, four types of goals such as (hard) goals, soft-goals, tasks and resources are used. Although these types are useful for some purposes, distinguishing them correctly is not easy. Especially, tasks are very useful for specifying a means for achieving a goal, but the other usages are not always effective. Although a soft-goal is not the same as a quality requirement, soft-goals are usually used for representing quality requirements. Although quality requirements are used for specifying how well a function is established [9], the relationship between quality requirements and a functional requirement is not clear in i* SD (Strategic Dependency) model.

We review several security extensions of i*. In secure tropos by Mouratidis [4], security concerns can be attached to both want- and can-relations in i* strategic dependencies. These concerns can be used for developing strategic rationale model, i.e., goal hierarchy in each actor However, how to write such concerns is out of scope of this study. In Secure i* [6], [7], each actor is assumed as an attacker, and attacks are assumed on the basis of the means of each actor. However, assumed attacks are limited because each actor will belong to several different business or activities in the real life. There is another secure tropos by Giorgini et al. [3]. In their research, additional relationship types such as ownership, trust and delegation is used in addition to dependencies. They use logic based formal techniques for analyzing the model. Because the model becomes more complex than normal i*, it seems to be difficult to write a model in such extension. Sutcliffe also proposed an extension of i* using trust [5].

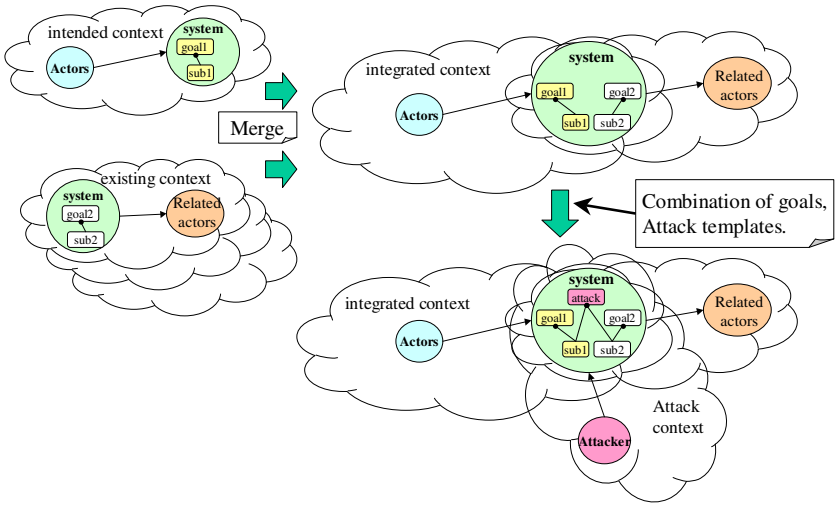

Fig. 1. Overview of the method

3 Method

3.1 Overview

The method contributes to elicit security requirements for a system, which is used in several different activities. We call a model of each activity a context. Figure 1 shows the overview in the method.

1. We temporarily introduce an information system to business or activities in a model. We call such a model "intended context". We may use patterns we proposed [10] for this introduction.
2. We also develop a model of existing usage of the system in the same way. We call the model "existing context". The existing context can be developed on the basis of observing the actual activities. Especially in the private usage, insecure applications tend to be used. Incorrect decomposition mentioned below should be embedded in each insecure application.
3. We merge the existing context with the intended context by unifying the system and their related actors respectively. We call the model "integrated context".
4. We explore suspected attacks on the basis of the combination of achievable goals in the system. To facilitate such exploration, STRIDE patterns [11] are used as templates for concrete attacks.
5. We explore how to mitigate or avoid the attacks. Because goals of both attackers and proper users are represented in goal trees, each goal can be regarded as a logic formula. We try to choose achievable sub-goals so that users' goals are satisfied and attackers' goals are not satisfied. We have to reconsider the goal trees or accept the existence of insecure goals when such achievable goals do not exist.

If more than one existing usages already exist, we have to repeat the steps above.

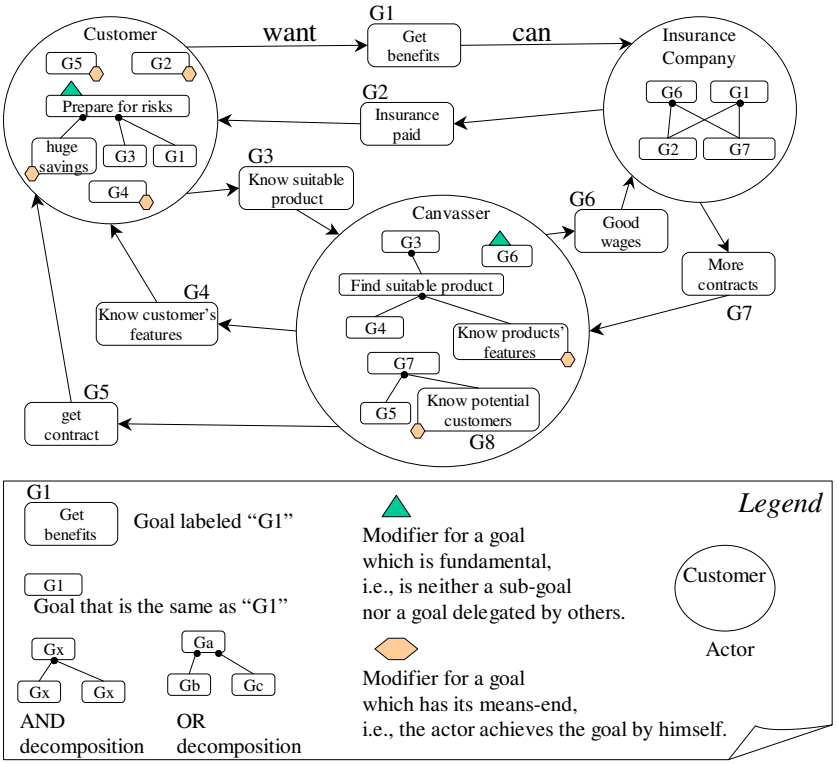

Fig. 2. An example of our simplified i* model about an insurance business. Incorrect goal decompositions are not contained in this example.

3.2 Simplified i* Model

For our security requirements analysis, we use simplified i* notation because we use strategic dependencies and goal hierarchies in each actor. Although most variations of i* contains them, they are not the original idea of the variations but the variations simply adapt them. We simplified original i* as follows. First, we do not distinguish four different types of goals (intentions) such as goals, soft-goals, tasks and resources because there are several constrains to construct goal graphs among such types [12]. For example, a goal cannot be decomposed into other goals directly, but the goal should be refined into other task(s). Second, we do not explicitly specify a task, which is a means for achieving a goal. Instead, we specify an actor has the means to achieve a goal. Third, we specify a goal is fundamental or not. We think a goal in an actor can be categorized into three types: a sub-goal of another goal, a goal delegated by another actor, and the rest. The rest goal can be regarded as a goal, which is initially and fundamentally wanted by the actor. For example, a goal of human "Want to survive" is really fundamental. We think we have to give priority to fundamental goals because the other types' goals are just subcontracts. Fourth, we introduce incorrect goal decompositions to specify a kind of malicious intentions. Although i* is based on the delegation of goal achievement, dependee does not always achieve a delegated goal properly in real life. For example, a

meeting participant does not report his vacant schedule correctly if he wants to disturb the meeting initiator.

By using an example in Figure 2, we explain our simplified i* model in detail. An insurance business is modeled in the example. In the model, three actors "Customer", "Canvasser" and "Insurance Company" are represented in circle. Goals are represented in round-rectangles. In i*, the actor A is called depender and the actor B is called dependee. Such dependency is represented as an actor-goal-actor relationship. For example, an actor "Customer" has a goal G1 "Get benefits", and another actor "Insurance Company" has a responsibility to achieve the goal. Such dependency is represented in the top of Figure 2. For convenience, we call a relationship between depender and its goal want-relation, and another relationship between the goal and dependee can-relation as shown in the figure. An actor who becomes a dependee of some goals has to achieve the goals. There are two typical ways to achieve them. The first way is to achieve it directly. The second way is to refine and to decompose the goal into several sub-goals, and the actor delegates other actors to achieve each sub-goal. An example of the first way is G2 in the actor "Customer". The actor "Insurance Company" wants "Customer" to pay insurance every month or year until the contract ends. The goal G2 represents such goal of the "Insurance Company". The "Customer" will achieve the goal G2 by get a salary for example, but we don't mention how to achieve G2 in this model. We simply attach a modifier (its shape is hexagon in our model), which clarifies the goal is achieved by the actor. An example of the second way is G1 in the actor "Insurance Company". The actor "Customer" wants "Insurance Company" to give benefits when he encounters some problem such as a sick or an injury. To achieve the goal G1, the company has to gather money in advance. The company thus has to have enough amounts of contracts (G7) and insurances are paid by each customer according to each contract (G2). Because these goals G2 and G7 cannot be directly achieved by the company, they are delegated to customer and canvasser respectively in this example. We then review the examples of fundamental goals. In our simplified model, a fundamental goal has a modifier which shape is triangle. In the model in figure 2, there are two fundamental goals: "Prepare for risks" of customer and G6 "Good wages" of canvasser. At least in this model, the goal is neither sub-goal of another nor a goal delegated by another actor. The actors fundamentally have such goals because of the reasons outside the model. The fundamental goals are the origin of the goals and their dependencies in the model. Therefore, the model is dramatically changed when some fundamental goals are withdrawn or modified. On the other hand, the other goals and dependencies are variable because there are several different alternatives to achieve the fundamental goals.

In original i*, a goal of a depender is assumed to be properly achieved by a dependee. For example in Figure 2, G3 "know suitable product" of Customer is properly achieved by Canvasser as follows. To achieve G3, the canvasser tries to find suitable product for the customer. He then asks the customer to tell her features such as age and income, and he investigate existing insurance products on the basis of the features. However, a goal of a depender is not always achieved by a dependee when malicious or incapable actors participate in the social dependencies. If a dependee is a malicious human or a system tampered by the human, the dependee intentionally decomposes the goal of a depender incorrectly. Tampering programs executed by users such as cross-site request forgery (CSRF) is typical case.

To represent such incorrect goal-decomposition, we introduce two modifiers to decomposition: one is cut-modifier and another is malicious-modifier. Each modifier is

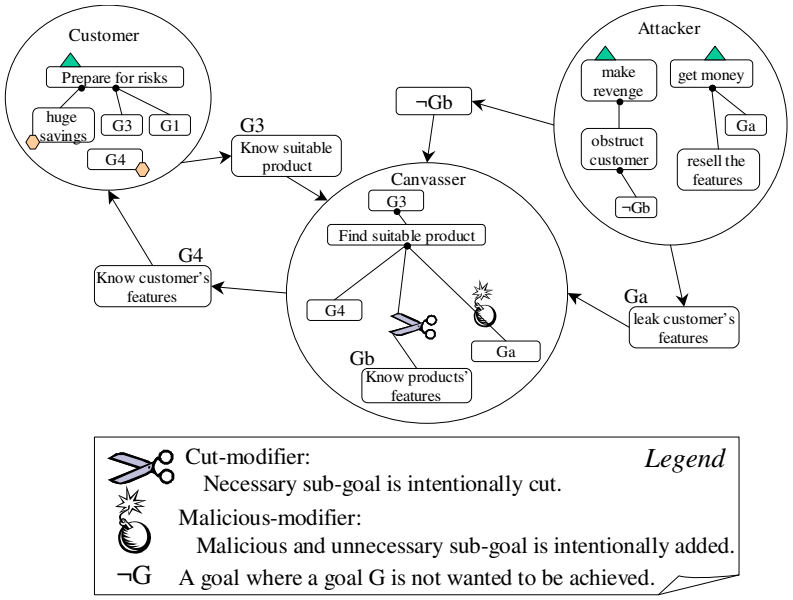

Fig. 3. Another example of our simplified i* model for explaining incorrect goal decompositions

represented in "scissors" and "bomb" icons respectively as shown in Figure 3. We explain how to use the modifiers by using Figure 3. The example in the figure is a modified version of the sub-model in Figure 2. To achieve G3, the canvasser has to achieve both G4 and Gb "Know products' features". However in this example, Gb is not achieved because the decomposition between "Find suitable product" and Gb is cut (by scissors). To realize this situation, inexperienced canvasser is dispatched to this customer. This cutting itself is represented as a goal of "Attacker", and the goal is represented as "¬Gb". The goal "¬Gb" is used as sub-goal to satisfy attacker's goal "make revenge" on the customer. In this example, we assume the attacker personally hate the customer. To achieve G3, the canvasser of course does not have to achieve Ga "leak customer's features". The attacker's threats can terrify the canvasser into achieving Ga when the attacker can take advantage of canvasser's weakness for example. The goal Ga then becomes one of sub-goals of "Find suitable product". This unnecessary goal Ga is used as means for an attack, and the attacker can earn money as shown in this example.

Cut-modifiers are not used so frequently because its super-goals become unachievable and depender can easily know incorrect decomposition in dependee. For example, G3 in Figure 3 becomes unachievable and Customer can know it. However, it is not easy to know sub-goals introduced by malicious-modifiers because its super-goals are still achievable even with the malicious sub-goals.

3.3 Deriving Attacker Context

As mentioned in overview of this method, an actor corresponding to a system contains various goals that do not normally coexist. In addition, systems sometimes contain incorrect goal decomposition where necessary sub-goals are omitted or malicious and

unnecessary sub-goals are added. An attacker can thus achieve his malicious goals (attacks) by using such goals. This is an extended idea by Liu et al. [6], [7]. In Liu's idea, a proper actor becomes an attacker, and he uses his achievable goals for his attacks. Our idea is almost the same as the Liu's idea, but attackers' goals can be predicted more than ever because the number of goal-combinations enabling attacks increases.

It is very difficult to predict the malicious intentions of attackers. We think the fundamental goals of attackers will be getting money, harassment, revenge, boasting his power and so on. However, they are too abstract to make the traceability between achievable goals in a system and such attackers' goals. We thus use STRIDE [11] as templates of temporary goals of attackers. STRIDE is a classification of the effects of realizing a threat and is an acronym of the following.

> Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege.

To realize a threat, each effect above requires several means. For example, an attacker can successfully tamper some information when he can interrupt and update it. The attacker can also successfully disclose some information when he can access and send it to somewhere. We find a combination of goals in an actor so that the means of the goals can realize an effect in STRIDE.

3.4 Mitigating or Avoiding Attackers' Goals

To mitigate or to avoid the attacks, we basically disable some of such goals in a system, and find alternatives not to disable goals of proper actors. Even before integrating contexts, actors including a system achieve several different fundamental and delegated goals at the same time. An actor normally hopes such goals in his strategic rationale will be achieved. On the other hand, an actor hopes that goals delegated by attackers, i.e., attacks are unachievable. Because each fundamental and delegated goal in an actor is a goal tree or a goal DAG, it can be represented in a propositional logic formula. The problem for satisfying goals and avoiding attackers' goals can be regarded as satisfiability problem of corresponding formulae. We will introduce the concrete examples of such formulae in the next section.

4 Case Study

In this section, our method is applied to the business for canvassing for the insurance contract to facilitate BYOD (Bring Your Own Device). The idea for using private smart devices in the context of business is called BYOD, and some companies try to facilitate the idea because of its merits related to economizing and usability. The goal of this case study is to illustrate whether our method works well.

Figure 2 shows the as-is model of the business before systems are introduced. In Japan, this type of business is popular in insurance business. An actor "Customer" is a potential customer of the insurance contract. An actor "Canvasser" is a salesperson who explains the insurance products and invites customers to make the contract. Normally in Japan, canvassers are members of specific insurance company. An actor "Insurance company" is the company of insurance products. Because of the fundamental goals "Prepare for risks" of Customer and G6 "Good Wages" of Canvasser, the model in Figure 2 is constructed. Because G6 is delegated to Insurance Company, it decomposes

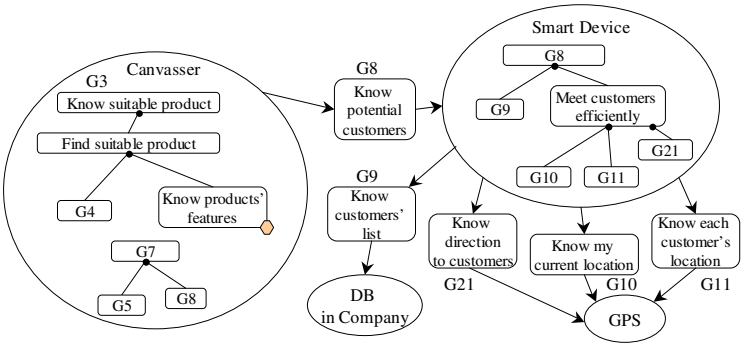

Fig. 4. A part of intended context in section 4.1

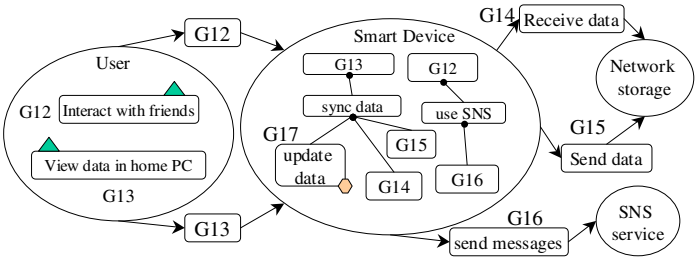

Fig. 5. Existing context in section 4.1

G6 into two sub-goals: G2 "insurance paid" and G7 "More contracts". Each sub-goal is delegated to Customer and Canvasser respectively. G7 is also decomposed into two sub-goals: G5 "get contract" and G8 "Know potential customers" in Canvasser. Because whether G5 is achieved or not depends on Customer, G5 is delegated to Customer. Canvasser achieves G8 by himself and means-end modifier is attached to G8 in Figure 2. For example about achieving G8, he has the knowledge about people in a specific area.

4.1 Supporting Knowledge Acquisition about Potential Customers

Figure 4 shows an intended context after introducing a smart device in Figure 2. Because the device interacts with external servers such as "DB in company" and "GPS" (Global Positioning System), actors corresponding to them are added in the model in Figure 4.

We also ask a canvasser to let us know his private usage of the smart device. In this case, he uses his private device at least for two goals. One is about interacting with his friends via the device, i.e., G12 in Figure 5. Another is about viewing his data in his home PC, i.e., G13 in the figure. Because the former goal is achieved by SNS, an actor "SNS service" and its related functionality are contained in this model as shown in the figure. Because the latter goal is achieved by network storage such as dropbox [1], "Network storage" and its related functionalities are contained in this model as shown in the figure.

[1] `https://www.dropbox.com`

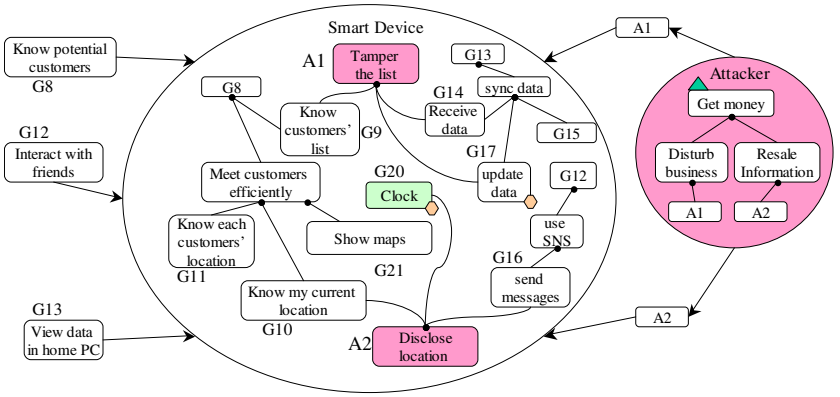

Fig. 6. A part of integrated context (including attack context) in section 4.1

Because "Smart Device" in Figure 4 is the same as one in Figure 5, we developed the integrated context by unifying them. We then explore suspected attacks on the basis of STRIDE mentioned in 3.3. We suspected two attacks as shown in Figure 6. One attack A1 "Tamper the list" can be achieved by G9, G17 and G14 because a program related to network storage can update data in the device and the customers' list is assumed to be stored as data. The contents for tampering the list can be also received by the G14 "Receive data". Another attack A2 "Disclose location" can be achieved by G10, "Clock" and G16 because of the similar reasons for A1. As shown in Figure 6, we suspect the gaps between a fundamental goal of an attacker "Get money" and these concrete attacks, e.g., A1 and A2. We suspect two intermediate goals of attackers: "Disturb business" and "Resale information".

To mitigate or avoid the attacks A1 and A2 in Figure 6, introducing the separation of duty is simple idea. However, required permissions for actual smart devices such as Android phones are too difficult for normal users and we normally have to accept their request for the permission [13]. It is thus hard to introduce clear separation of duty in smart device usage. Another idea for mitigating or avoiding the attacks is to explore alternatives to achieve super goals of goals that are used in the attacks. As mentioned in section 3.4, fundamental and delegated goals can be represented in logic formulas. At the smart device in this case, G8, G12 and G13 are wanted to be achieved, and A1 and A2 are wanted to be unachieved. These goals can be represented in the following formulas.

$$\mathrm{G8} \equiv \mathrm{G9} \wedge (\mathrm{G21} \vee (\mathrm{G11} \wedge \mathrm{G10}))$$
$$\mathrm{G12} \equiv \mathrm{G16}$$
$$\mathrm{G13} \equiv \mathrm{G14} \wedge \mathrm{G17} \wedge \mathrm{G15}$$
$$\mathrm{A1} \equiv \mathrm{G9} \wedge \mathrm{G14} \wedge \mathrm{G17}$$
$$\mathrm{A2} \equiv \mathrm{G10} \wedge \mathrm{G20} \wedge \mathrm{G16}$$

To choose an alternative G21 and to disable G10, G8 and G12 can be achieved and A2 can be also unachieved. However, there is no alternative to achieve G8 and G13 and to avoid A1 at the same time in this model. We thus have to find alternatives to G9 and/or G14 respectively for achieving G9 and G13 and for avoiding A1. To explore other attack possibilities, intermediate goals of the Attacker in Figure 6 are useful. For example, tampering the location data can disturb the efficient travel of the canvasser.

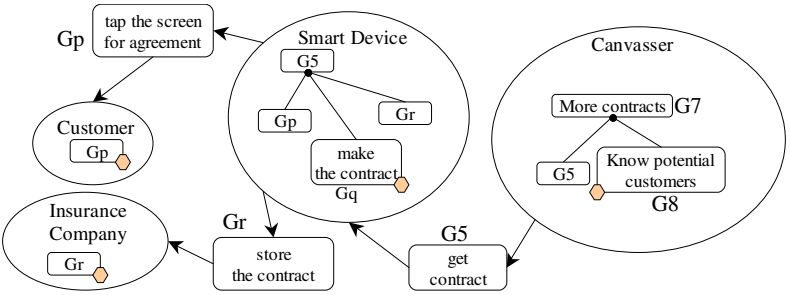

Fig. 7. A part of an intended context in section 4.2

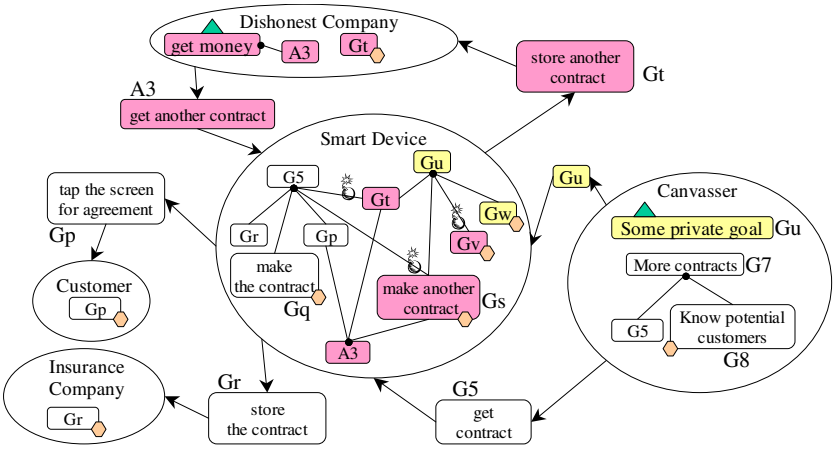

Fig. 8. A part of integrated context in section 4.2

4.2 Easing Reluctance to Make the Contract

Making contract of the insurance is highly reluctant because it traditionally requires a lot of paper works and customers have to sign the documents a lot. We thus try to ease the reluctance by using the smart device. Although the copyright transfer agreement of a technical paper even required written signature, some publishers such as IEEE recently provides the web sites for skipping written signature and we personally become happy without sending the fax to publishers. In the same way, we try to provide insurance customers to make contract easily. In Figure 2, the Customer has to directly satisfy G5 "get contract" by his paper works. To introduce a smart device such as a smart phone, we ease this reluctant works as shown in Figure 7. In the figure, the Customer simply taps the screen of the smart device for his agreement, and a smart device makes the contract document and sends it to the insurance company automatically.

We assume Canvasser satisfies some private goal by some free software on the smart device and the software contains some malicious codes. On the basis of this existing context, we depict the to-be integrated context in Figure 8. In the figure, Gu corresponds to the private usage of the smart device. To satisfy Gu, the device primarily achieves the sub-goal Gw. However, there is a possibility that Gu contains several malicious sub-goals: Gv corresponds to tampering G5, and Gt and Gs correspond to malicious codes

embedded in G5. The possibility is predicted on the basis of the permissions given to an application achieving Gu. In the figure, unintended and unnecessary contract (another contract in the figure) is made by the malicious codes. For example, unnecessary goods are bought and paid against the intention of the Customer. As a result, a Dishonest Company, i.e., an attacker can get money by spoofing the Customer, i.e., getting another contract. Goals in the smart device can be represented in the following formulas.

$$\mathrm{G5} \equiv \mathrm{Gr} \wedge \mathrm{Gq} \wedge \mathrm{Gp} \wedge \mathrm{Gt}$$
$$\mathrm{Gu} \equiv \mathrm{Gw} \wedge \mathrm{Gv} \wedge \mathrm{Gs} \wedge \mathrm{Gt}$$
$$\mathrm{A3} \equiv \mathrm{Gp} \wedge \mathrm{Gt} \wedge \mathrm{Gs}$$

Because the goal tree of Gu is predicted on the basis of the permission of its corresponding application, the tree contains malicious goals such as Gt, Gs and Gv. Because of the abilities of such malicious goals, G5 also contains Gt and Gs. The best way to avoid A3 is of course to uninstall the polluted software for achieving Gu, and to install more reliable software that can achieve Gu. However, it is not so easy because an application corresponding to Gu is private one. At least, the boss of the canvasser cannot enforce its uninstall.

4.3 Discussion

For suspecting attackers' goals, STRIDE in section 3.3 seem to works well. To finding the combination of achievable goals for each STRIDE such as tampering, ontological support [14] seems to be effective because the combination for each STRIDE has typical patterns. For example, tampering requires both receiving data from outside and writing data. We want to construct the ontology for such support.

For resolving the security problems, focusing on logic formulas corresponding to goal trees in each actor is simply and effective. However, the problem is not always resolved. For example, an attack A3 in section 4.2 cannot be resolved because of the private reason. One of the simple solutions for this problem is to introduce the separation of duty explicitly. Another solution is to introduce the monitoring mechanism based on our model, and to mitigate the security problems. It is more feasible than introducing rigorous separation of duty because we cannot live in the completely secure environment. We can minimize the activities to be monitored in the existing context based on our model. Each worker can accept such monitoring mechanism because the reasons and the range of monitoring are explained.

We had a workshop to explain our method to security practitioners. Although we did not compare existing i* extensions to our method, at least they understood our method well. From our other experiences [15], we found that practitioners and even students cannot correctly develop full i* model. We thus assume our simplification contributes to motivating practitioners to use our i* variation.

5 Conclusion

In this paper, we present a method for analyzing security requirements for a system, which is already used in several different activities. We use simplified i* model for this analysis. Through a case study of BYOD, we confirmed the method works well. Because the vocabularies in our simplified i* are so limited for security analysis, usual security experts can accept the notation.

In our previous work [16], we presented a method for evaluating a system introduction on the basis of the amount of responsibilities and satisfaction of human in average. Because we defined these metrics on the basis of the structural characteristics of an extended i* model, we can systematically compare an as-is model with its to-be model written in our extended i* model. However, such metrics only focus on positive aspects, and there is no metrics with respect to the security issues. We want to propose and evaluate the metrics for security as well as our current metrics.

References

1. Yu, E.S.K.: Towards modeling and reasoning support for early-phase requirements engineering. In: RE, pp. 226–235 (1997)
2. Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J.: Social Modeling for Requirements Engineering. The MIT Press (2010) ISBN 0262240556
3. Giorgini, P., Massacci, F., Mylopoulos, J., Zannone, N.: Modeling security requirements through ownership, permission and delegation. In: RE, pp. 167–176 (2005)
4. Mouratidis, H., Giorgini, P.: Secure tropos: a security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering 17(2), 285–309 (2007)
5. Sutcliffe, A.G.: Trust: From cognition to conceptual models and design. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 3–17. Springer, Heidelberg (2006)
6. Liu, L., Yu, E.S.K., Mylopoulos, J.: Security and privacy requirements analysis within a social setting. In: RE, pp. 151–161 (2003)
7. Liu, L., Yu, E.S.K., Mylopoulos, J.: Secure-i*: Engineering secure software systems through social analysis. Int. J. Software and Informatics 3(1), 89–120 (2009)
8. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
9. Blaine, J.D., Cleland-Huang, J.: Software Quality Requirements: How to Balance Competing Priorities. IEEE Software 25(2), 22–24 (2008)
10. Kaiya, H., Morita, S., Ogata, S., Kaijiri, K., Hayashi, S., Saeki, M.: Model Transformation Patterns for Introducing Suitable Information Systems. In: 19th Asia Pacific Software Engineering Conference, APSEC 2012, Hong Kong, pp. 434–439. IEEE CS (December 2012)
11. Frank Swiderski and Window Snyder. Threat Modeling. Microsoft Press (2004)
12. Yu, E., Castro, J., Perini, A.: Strategic Actors Modeling with i*. In: RE 2008, Tutorial (August 2008)
13. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: ACM Conference on Computer and Communications Security, pp. 627–638 (2011)
14. Shibaoka, M., Kaiya, H., Saeki, M.: Goore: Goal-oriented and ontology driven requirements elicitation method. In: ER Workshops, pp. 225–234 (2007)
15. Honiden, S., Tahara, Y., Yoshioka, N., Taguchi, K., Washizaki, H.: Top se: Educating super-architects who can apply software engineering tools to practical development in japan. In: ICSE, pp. 708–718 (2007)
16. Kaiya, H., Morita, S., Kaijiri, K., Hayashi, S., Saeki, M.: Facilitating business improvement by information systems using model transformation and metrics. In: CAiSE Forum, pp. 106–113 (2012)