Special issue: The future of software engineering for security and privacy

Education Paper

Curriculum design and methodologies for security requirements analysis

Kenji TAGUCHI[1] and Yasuyuki TAHARA[2]
[1,2]National Institute of Informatics

ABSTRACT
Lack of security is one of the most widespread problems affecting information systems. Security breaches at companies are reported almost everyday and users of computer systems are busy updating security of their computer system against vulnerabilities. While some of these problems are caused by human errors and faults of physical devices but the majority of them are due to the defects in software systems. The best way to reduce them is to find and fix them in an early stage of the software development, especially in the requirements elicitation and analysis phases. In this paper, we will present how we designed a security requirements analysis course to address this issue. We will also present security requirements elicitation methodologies based on the agent- and goal-oriented requirements analysis methodologies of KAOS and i*.

KEYWORDS
Security, requirements analysis, KAOS, i*, RBAC, CC

1 Introduction

Lack of security is one of the most widespread problems affecting information systems. Security breaches at companies are reported almost everyday and users of computer systems are busy patching the vulnerabilities of their computer systems. While some of these problems are caused by human errors and faults of physical devices but the majority of them are due to the defects in software systems.

Aside from security issues, one of the main issues in software engineering is how to reduce software defects. One way to reduce them is *finding and fixing them during the requirements and design phases* [1]. The requirements elicitation and analysis phases are especially important in this regard.

There are a good number of computer security courses that deal with security problems such as buffer overflow at the code level and how to protect servers from attackers at the network device level. However, we have been unable to find the topic of elicitation and analysis of security requirements in existing software engineering courses. To help fill this vacuum in security education, we have developed a course teaching practical skills for eliciting and analyzing security requirements. In this paper, we discuss how we designed this course in our education program for IT specialists called Top SE [3] and describe what methodologies are taught in it.

This paper is organized as follows. The next section presents an overview the Top SE program. Section 3 outlines the Requirements Analysis course series. Section 4 presents the Security Requirements Analysis course and its methodologies. Section 5 briefly touches on future directions of this work and Section 6 is a summary of the paper .

2 Top SE

Top SE is a non-accredited course at the masters level; it is fully funded by the Japanese government and is operated through a close collaboration between industry and academia at the National Institute of Informatics. Table 1 lists the whole curriculum, which consists of five lecture series (Requirements Analysis, System Architecture, Formal Specifications, Model Checking and Implementation). The overview and the curriculum design of the program are discussed in [2].

The program has the following distinguishing features:

Received September 28, 2007; Revised November 14, 2007; Accepted December 12, 2007.
[1] *ktaguchi@nii.ac.jp, http://honiden-lab.ex.nii.ac.jp/˜taguchi/*
[2] *tahara@nii.ac.jp, http://honiden-lab.ex.nii.ac.jp/˜tahara/*
DOI: 10.2201/NiiPi.2008.5.4

Table 1 Curriculum.

Series	Courses
Requirements Analysis	Requirements Analysis Security Requirements Analysis
System Architecture	Component-based Development Software Patterns Aspect-Oriented Development
Formal Specifications	Formal Specifications (Foundation) Formal Specifications (Advanced) Formal Specifications (Security)
Model Checking	Design Verification (Foundation) Design Verification (Advanced) Software Model Checking Real-time Model Checking Modelling and Verifying Concurrent Systems
Implementation	Testing Program Analysis

- Wide coverage of cutting-edge software engineering technologies
- Real case studies from industry
- Tool first
- Emphasis on engineering practices rather than theory

The program covers a wide range of cutting-edge software engineering technologies, e.g., software patterns, aspect-oriented development and model checking. We regard Top SE as a medium to transfer these new technologies to industry. To do so, we tailor our program to meet the educational needs of industry and create case studies for laboratory work with our industrial partners (Hitachi, NEC, Toshiba, NTT Data, Fujitsu, Nihon Unisys, among others). We also have a close collaborative relationships with our academic partners (Shinshu Univ., Tsukuba Univ., etc.) to develop and deliver the courses.

As software engineering is a practical engineering discipline, we teach good practices rather than theory in all courses. We specifically emphasize the use of tools; all courses are based on laboratory work in which several tools are used.

3 Requirements analysis course series

The Requirements Analysis course series consists of *Requirements Analysis* and *Security Requirements Analysis*. The former is a pre-requisite of the latter.

The key learning objective of the Requirements Analysis course is to teach skilled software engineers how to apply the agent- and goal-oriented requirements analysis methodologies to practical software development. KAOS [7]–[9] and i* [24] are mainly taught in the course. The details of the course and the student error models are discussed in [5].

3.1 KAOS

KAOS is a goal-oriented requirements engineering methodology developed by van Lamsweerde and his associates [7]–[9]. KAOS covers the whole requirements elicitation process. KAOS has a two-layered architecture in which the graphical layer is used for conceptual modeling, and the assertion layer is optional and supports a first-order real-time logic to specify constraints associated with goals for formal reasoning [10]. In our course, we only teach the graphical layer of KAOS. We use [11] as the reference for graphical representation of that layer.

The KAOS graphical layer consists of a goal hierarchy and an object model. The requirements elicitation process consists of elaboration of goals in a hierarchy and associated objects and operations in the object model. The elaboration of goals step may begin with identified high-level goals, which are then refined into sub-goals until terminal goals, which cannot be further refined are reached. A terminal goal is either an *expectation* or a *requirement*. An *expectation* is a terminal goal to be achieved by an agent in the system environment and a *requirement* is a terminal goal to be achieved by a software agent. Terminal goals are *operationalized* to incorporate an agent with its operation and entities that serve as input and output objects of the operation.

In [11],

> The KAOS object model is compliant with UML class diagrams in that KAOS entities correspond to UML classes; and KAOS associations correspond to UML binary association links or n-ary association classes.

This compliance with UML gives us freedom to use the object model in KAOS as UML class diagrams including extension mechanisms such as stereotypes, which we will use to create an access control policy model in KAOS.

3.2 i*

The i* goal-oriented requirements engineering methodology was originally proposed by Yu [24]. Its main feature is that it handles the social aspects of system requirements such as the relationships of stakeholders. The methodology is as follows.

- Before conducting a requirements analysis of the system to be implemented, i* analyzes the as-is status, that is, the social environment in which the system is to be put.

- The as-is status analysis identifies dependency relationships between actors; that is, the i* model is a representation of the stakeholders and the systems that already exist or are going to be developed. Such dependency relationships are used to identify trade-offs in the requirements of the future systems.

- Two kinds of models are used in i*. *Strategic Dependency* (SD) models focus on the organizational aspects of the system environments; they consist of dependency relationships between actors. *Strategic Rationale* (SR) models detail the concerns of each actor.

i* models have hard goals and soft goals. These goals are distinguished by their definition of satisfaction. A hard goal is totally satisfied or not at all. A soft goal has a degree of satisfaction represented by a number between 0 and 1. Two other constructs are included in i* models: tasks and resources. A task represents a routine to satisfy goals. A resource is a physical or informational entity used to satisfy goals or carry out tasks. A dependency relationship between actors is involved in a goal, a task, or a resource. i* models can incorporate other relationships, such as AND-OR decompositions and contributions representing the influence of constructs in satisfying a soft goal. Examples of i* models appear in the latter parts of this paper.

4 Security requirements course

The course is concerned with security issues in the requirements analysis and elicitation phases. It introduces several methodologies for security-related topics with the aim of exposing students to more than one security methodology.

There are so many security issues and it is not wise to assume that a single methodology can be applied to all of them. For this reason, we chose to teach several other methodologies such as misusecases, abusecases and security usecases in addition to KAOS and i*.

The course consists of twelve lectures as shown in Table 2. Most of the lecture time is devoted to laboratory works using tools. We use a UML CASE tool called JUDE for constructing usecases and K-Tool, our original tool for KAOS and ST-Tool [4] for i*. The intended learning outcomes are as follows:

- The student demonstrates an understanding of security issues

- He or she demonstrates an understanding of the methodologies used in eliciting and analyzing security requirements

- He or she can explain and construct security-related models in usecase variants (misusecases, abusecases and security usecases), KAOS and i*

- He or she can compare different approaches to security requirements analysis and elicitation

Security issues in the requirements elicitation and analysis phases are largely ignored in industry so we carefully chose the topics we would teach in the course. We chose the following topics:

- Security Control Policy

- Copyright

- Common Criteria (CC)

We chose the Role-Based Access Control (RBAC) policy, since designing and implementing secure and robust access control policies for organizations and computing systems has become a critical aspect of protecting the information and resources available to them. In particular, access control policies will become very complex in large and complex organizations such as the Department of Homeland Security [14] and Dresdner Bank [13]. In the latter case, it is reported that there are over one thousand roles. To control complexity and ease the elicitation of requirements process for the access control policies, we should adopt a well-established requirements engineering methodology at an early stage of the software development process; hence we chose KAOS for this purpose.

There is a vast amount of research on models for access control policies in general and role-based access control (RBAC) in particular. A reference model in this area is the proposal on RBAC by NIST [6] in the form of ANSI INCITS 359-2004. We chose NIST RBAC as our teaching materials on this problem.

Our course uses an example of copyright protection for hard disk (HD)/DVD recorders. Although the copyright issue is not exactly a security problem, it is closely related to various security issues and functionalities. For example, many software products treating text or multimedia data include access control mechanisms to prevent the data from being copied. Most digital rights management (DRM) mechanisms use security-related technologies such as cryptography and digital watermarks. For this reason, we treat this copyright problem as a topic of security requirements analysis.

The problem is as follows.

Table 2 Course Structure.

Number	Description
1	Introduction
2	Usecases for Security (1)
3	Usecases for Security (2)
4	Access Control Policies (1)
5	Access Control Policies (2)
6	Interim Presentation
7	Copyright (1)
8	Copyright (2)
9	Common Criteria (CC)
10	Lab Work on CC (1)
11	Lab Work on CC (2)
12	Final Presentation

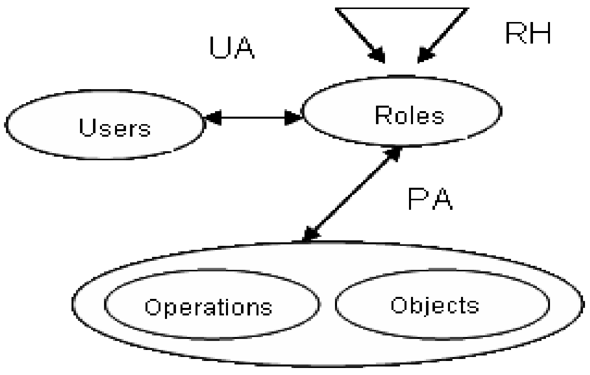

Fig. 1 Hierarchical RBAC.

- If a user tries to copy multimedia data recorded in an HD recorder to a DVD recorder, she must not violate the copyright of the TV station for the data.
- The students of this course need to identify functionalities of the HD and DVD recorders and issues raised by the functionalities.

Our security requirements analysis course aims at enabling students to apply the advanced requirements analysis techniques to their workplace. This course, in the same way as the other courses of Top SE, treats an international standard ISO/IEC 15408 as the framework of security requirements specifications and the catalog of security functionality requirements specifications. This standard is called "Common Criteria for Information Technology Security Evaluation" or *Common Criteria* or *CC* for short. It is for evaluating the security of IT systems. However, the objective of evaluation is not the security strength but the "level of confidence that the security functionality and its assurance measure meet the specified requirements" [20]. Therefore, the requirements specifications are as important as the system itself for the CC evaluation. CC specifies the format of the requirements specifications called the *Security Target* or *ST* for short. The course teaches students how to write an ST efficiently through an i*-based security requirements analysis.

4.1 Role-Based Access Control (RBAC)

Organizations and systems use Role-Based Access Control (RBAC) to protect resources from illegal access. Its applications include database systems, enterprise security administration software and programming languages [15]. NIST proposed a standard in which a basic model called Core RBAC is defined [6]. Core RBAC consists of users, roles, and permissions, which can be further divided into operations and objects. An operation is an executable command or instruction which uses objects as its resources. Core RBAC is further extended to include hierarchies in roles and role constraints.

A role plays a central part in RBAC, which is an intermediary between users and permissions. A user is not granted permissions but is instead assigned to a role that is associated with permissions. These concepts and relations only give a flat structure to roles; most organizations and systems have hierarchically structured roles to control inheritance of permissions. Fig. 1 illustrates their relationships, which are a many to many mapping of a role onto a set of users (UA), a many to many mapping of a permission to a set of roles (PA), and the role hierarchies (RH):

Role hierarchies control the inheritance of permissions. Senior roles inherit permissions from their junior roles in the senior-junior relation between roles.

Constrained RBAC is Core RBAC with an additional constraint called Separation of Duties (SOD). SOD is a basic principle on how to assign permissions to users. Any critical operation which might breach security cannot be assigned to a single individual.

For more details regarding RBAC, the reader is referred to references [6], [15].[1)]

4.1.1 RBAC-KAOS

Here, we will present how an RBAC model can be derived from KAOS. We use the existing model elements in KAOS to model the RBAC elements of *Users*, *Roles* and *Permission* (*Operations* and *Objects*) illustrated in the figure A·1 in Appendix A.

Permissions are not represented; instead, *Operations* and *Objects* are represented by KAOS operations and

1) This paper only deals with core RBAC with general role hierarchies and without *sessions*.

entities. Generalization/specialization relationships between agents are used for role hierarchies (RH). A mapping of a user to roles (UA) is represented by a directed association. A mapping of a role to permission (PA) is represented by a performance relation between the role and an operation and an input relation between the operation and an object.

This interpretation is very natural and does not diverge from their original meanings in KAOS.

In the diagram goals are represented by parallelograms (e.g., *Handle Payroll*). A yellow line with a yellow circle represents a refinement of a parent goal to child goals. An agent is a yellow box with angled corners with a stereotype ⟨⟨role⟩⟩ (e.g., *Payroll clerk*) to represent a role. The agent (in fact a role) responsible for a goal is linked by a red arrow with a red circle in the middle of the line[2]. An operation is a pink oval (e.g., *Update*) that is associated with a role by a performance link and with an entity (object) in yellow rectangle by an input link and with a requirement by a link with a blue circle.

This diagram exemplifies how payrolls are handled and how exclusive tasks are separately assigned to different roles. It should be noted that the RBAC model only occupies the bottom half of the diagram, but the traceability of KAOS gives a direct link between the requirements of security policies represented by goals and the RBAC model. Hence, it clearly shows how the RBAC model is derived from abstract descriptions of goals.

In this modeling method, the agents in KAOS are simply divided into *users* and *roles* with specific stereotypes ⟨⟨user⟩⟩ and ⟨⟨role⟩⟩ to carry additional meanings.

One of the difficulties of requirements elicitation and analysis is how to find and resolve conflicts between stakeholders [12]. Conflicting goals are ones that cannot be satisfied simultaneously. KAOS provides a graphical representation for conflicting goals (a line with a zig-zag symbol) so that if two agents (i.e., stakeholders) are assigned conflicting goals, it shows that their requirements conflict. A different kind of conflict appears in eliciting access control policies, i.e., conflict of interests. Separation of Duties (SOD) is used to eliminate conflict of interests. For example, in the Fig. A·1, the goal *Generate paychecks* and *Manipulate register table* should be assigned to different roles, *Payroll clerk* and *Cashier*. These goals have conflicting interests in a sense that giving permissions to a role to achieve those two goals enables any user assigned to that role to compromise security. This is a situation in which the traditional conflict resolution method cannot be applied. Our analysis for SOD is not a definitive way of eliciting access control policies, but rather it combines the use of general guidelines as to which positions should be separated in organizations and possible scenarios to check whether there are conflicts of interests in goals.

[2] An agent with the stereotype ⟨⟨user⟩⟩ should not be linked with a requirement.

4.2 i*-Based security requirements analysis approach

Here, we describe the security requirements analysis approach based on i* and risk analysis approaches [22]. Our approach combines the following techniques.

- Liu et al [21] proposed a security RE technique based on i* that analyzes extensive constructs of a security requirements model including vulnerability, threats, attacks, and countermeasures. Our approach is mainly based on this technique (we call it Liu's method hereafter).
- HazOp (HAZard and OPerability study) [23] is a well-known risk analysis technique. Recently, it has been applied to security risk analysis for software. We found that HazOp can be effectively combined with Liu's method.

We first explain our approach and illustrate an exercise using the example of HD/DVD recorders. To improve readability, all the figures of i* models are put in Appendix A.

Our approach here is to enhance Liu's method with risk analysis techniques. Liu's method is as follows (see also Fig. 2).

Usual i∗ Process The i∗ process included in Liu's method (called *domain requirements analysis process* in [21]) is divided into the three steps: (1.1) Actor Identification, (1.2) Goal/Task Identification, and (1.3) Dependency Identification. (1.1)

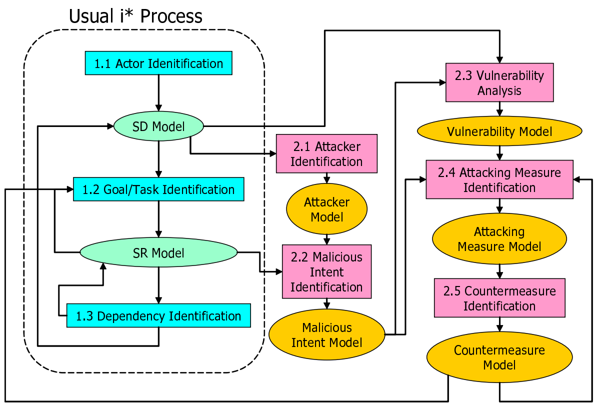
Usual i* Process
1.1 Actor Idenitification
SD Model
1.2 Goal/Task Identification
SR Model
1.3 Dependency Identification
2.1 Attacker Identification
Attacker Model
2.2 Malicious Intent Identification
Malicious Intent Model
2.3 Vulnerability Analysis
Vulnerability Model
2.4 Attacking Measure Identification
Attacking Measure Model
2.5 Countermeasure Identification
Countermeasure Model

Fig. 2 Liu's Method. [21]

and (1.3) transform the SD model and (1.2) transforms the SR model.

2.1 Attacker Identification The main idea of this step is to imagine that any actor could behave as an attacker. This not only means that an unknown attacker could behave as if he is a legitimate actor through the masquerading attack, but also includes the case in which the attacker makes use of intent represented by goals, system functionalities or human behaviors represented by tasks, and resources that are available to the attacker for carrying out the attack.

The resulting model includes the attacker actors and is called an *attacker model*. In this paper, we distinguish model constructs involved in the attacker actors such as malicious intent goals and attacking measure tasks from others. We call a model including such constructs an *attack model*, while we call a model with no such constructs a *domain model*.

2.2 Malicious Intent Identification Identify the malicious intent of the attackers as goals of the attacker actors. The resulting model includes such goals and is called a *malicious intent model*.

2.3 Vulnerability Analysis Identify a part of the SD model that has the possibility of being attacked as a vulnerability. The resulting model is called a *vulnerability model*.

2.4 Attacking Measure Identification Identify the actions realizing the malicious intent as attacking measure tasks. The resulting model is called an *attacking measure model*. HazOp is applied to this step for systematic identification.

2.5 Countermeasure Identification Identify the countermeasures against the attacking measures as tasks. The resulting model is called a *countermeasure model*. Note that a countermeasure model is a domain model because the attacking measure tasks are transformed into softgoals representing the degree of avoidance of the attacks and other constructs about the attacks are removed.

Our course uses the following copyright protection problem for HD/DVD recorders:

- If a user tries to copy multimedia data recorded in an HD recorder to a DVD recorder, she must not violate the copyright of the TV station for the data.
- The students need to identify the functionalities of the HD recorder and the DVD recorder as well as the issues raised by these functionalities.

The taught process is described from 4.2.1 to 4.2.5 below.

4.2.1 Domain requirements analysis

The scenario below is the usual i* process and create a model of environments and requirements other than security. Because the entire model for HD/DVD recorders would be very large, we handle only the part of it involved in the security RE process. The scenario is as follows.

1. Identify actors. They include not only the user and the TV station that can be easily identified from the problem descriptions, but also the devices such as the HD recorder, DVD recorder, and TV set.
2. Identify the goals. Here, we identify the hard goal “Copy Contents” of the user and the soft goal “Protect Copyright” of the TV station.
3. Analyze the “Copy Contents” goal as follows.

    - Delegate the “Copy Contents” goal to the DVD recorder; that is, identify the dependency of the user to the recorder about the goal, because the recorder is expected to achieve the goal completely.
    - Divide the goal of the DVD recorder into two sub-goals “Receive Content” and “Record Content to DVD”.
    - Identify the goal “View Contents Later” of the user as the super goal of the “Copy Contents” goal.
    - Proceed with analyses for each goal.

As a result, we can create the SD model of Fig. A·2 and the SR model of Fig. A·3.

4.2.2 Attacker and malicious intent identification

We show an example case of treating the user as an attacker. The attacker identification and malicious intent identification are carried out as follows.

1. Identify the attacker actor “Attacker” and the malicious goal “Violate Copyright” as the intent of the attack.
2. Change the type of “User” actor to “Role” and create a “Play” association between them (Fig. A·4). The purpose of these operations is to indicate that the user is considered to be an attacker.
3. Create an actor “User As Attacker” by putting “User” and “Attacker” together and put their goals into a new actor to clarify the situation in which “Attacker” “Plays” the “User” role and proceed with the analyses. The results of the model analyses are shown in Fig. A·5.

4.2.3 Vulnerability identification

This step identifies the range of influence of the identified malicious goals, their subgoals, and their subtasks. Fig. A·6 is an example of the result of this step in which the "Protect Copyright" softgoal delegated from the TV station to the user is vulnerable.

4.2.4 Attacking measure identification

This step identifies attacking measures as tasks that must be completed to achieve the malicious goals. It is possible to use (or *misuse* in this case) the domain model constructs to carry out the attacking measures. As the original Liu's method does not provide specific techniques to identify the measures, our course teaches a technique using HazOp shown in 4.2.5. For example, the attacking measure model shown in Fig. A·7 is created.

4.2.5 Application of HazOp

HazOp is a risk analysis technique that helps analysts to identify potential risks in systems. This technique has started to be used in security analysis. Our course applies it to increase the applicability of Liu's method.

HazOp is based on a typical risk analysis procedure in which (1) the targets of the analysis are identified first, (2) then the potential risks are identified, and (3) finally the risk reducing measures are identified. HazOp mainly focus on step (2). The security requirements analysis described before lacks the detailed techniques for the task corresponding to this step. Thus, we consider the combination of these approaches to be very useful.

The main feature of HazOp in step (2) is application of guide words. A *guide word* is "a short word to create the imagination of a deviation of the design/process intent" [22]. A deviation and a design/process intent correspond to a malicious intent goal and a domain goal of i* models respectively. Thus HazOp is used to identify malicious intent goals and attacking measures. Table 3 shows a typical set of guidewords and their usage in our approach.

Table 4 shows an example of the application of HazOp to our example. Thus the initially identified malicious intent goals are obtained through application of HazOp.

4.3 CC and ST

We explain the details of CC and how we can use our approach to create efficiently an ST document prescribed in CC.

4.3.1 Organization of CC

CC consists of the following three parts.

Part 1: Introduction and general model describes the background and the fundamental ideas of security evaluation and specifies the general model of assets, countermeasures, and evaluation. In particular, the format of ST is specified here. In addition, another type of document, called *Protection Profile*, or *PP* for short, is specified. A PP is a subset of ST and is used as a reusable module for writing STs.

Part 2: Security functional components is a catalog of security functionalities. The functionalities are divided into eleven classes such as security audit, communication, cryptographic support, and user data protection. It is recommended to choose security functionalities from here when writing an ST or a PP.

Part 3: Security assurance components is a catalog of inspection tasks to make sure that the security functionalities are correctly implemented. Such tasks are divided into ten classes. This part also specifies *Evaluation Assurance Level*s, or *EAL*s for short, which is a set of assurance requirements. There are seven levels of EALs (from EAL1 to EAL7). These levels do not represent the security strength but the ranges of targets of inspection and the degrees of inspection.

4.3.2 ST Organization

In CC, an ST is defined as an implementation-dependent statement of security needs for a specific identified TOE (Target Of Evaluation). The organization of ST is as follows.

1. ST introduction: describing the TOE in a narrative way on three levels of abstraction: ST and TOE references, TOE overview and TOE description.
2. Conformance claims: describing how the ST conforms with the Common Criteria itself, Protection Profiles (if any), and Packages.
3. Security problem definition: defining the security problem that is to be addressed.
4. Security objectives: a concise and abstract statement of the intended solution to the problem defined by the security problem definition.
5. Extended components definition: defining components that are not based on those in CC Part 2 or CC Part 3.
6. Security requirements: consists of security functional requirements (SFRs) and security assurance requirements (SARs).

   SFRs are translations of the security objectives for the TOE into a standardized language.

   SARs are descriptions of how assurance is to be gained that the TOE meets the SFRs

Table 3 Guidewords and Their Usage.

No	Do not carry out a task or satisfy a goal
More	Carry out a task or satisfy a goal too much by repetition, concurrent execution, or using extreme values
Less	Carry out a task or satisfy a goal too little
As well as	Carry out an additional task or satisfy an additional goal
Part of	Carry out a task or satisfy a goal only partially
Other than	Carry out a task to satisfy an inappropriate goal or satisfy a goal to satisfy an inappropriate supergoal

Table 4 Example of Application of HazOp.

Domain goal	Guideword	Malicious intent goal
Copy Contents	More	Copy Contents Many Times
	Other than	Copy Copy-Protected Contents
Record Contents	Other than	Record Record-Protected Contents

7. TOE summary specification: describing the general technical mechanisms that the TOE uses to satisfy all the SFRs.

When creating an ST using security requirements analysis approaches, we use the correspondence between an ST and the security requirements model shown in Fig. 3. As this table shows, the majority of an ST can be written using security requirements analysis approaches.

To apply Liu's method to writing an ST, we establish the mapping from the models created using Liu's method according to the relationships between i* and ST. The mapping consists of the relationships between security requirements analysis in general and ST shown in Fig. 3 and the following ones.

- i* goals and tasks to various requirements
- In writing "3.1. Threats",
  - i* actors (attackers) as threat agents
  - i* resources (to be protected) as assets
  - i* tasks (of attacking) as adverse actions
- In writing Rationales (4.3., 6.3.), i* goal models are mapped to table representations of relationships between requirement items.

We should note that when writing an ST, goals or tasks corresponding to the following sections should be refined to the level of CC Part 2: 5. Extended components definition and 6.1. Security functional requirements.

As an exercise of writing an ST, we use the following example of a document management system.

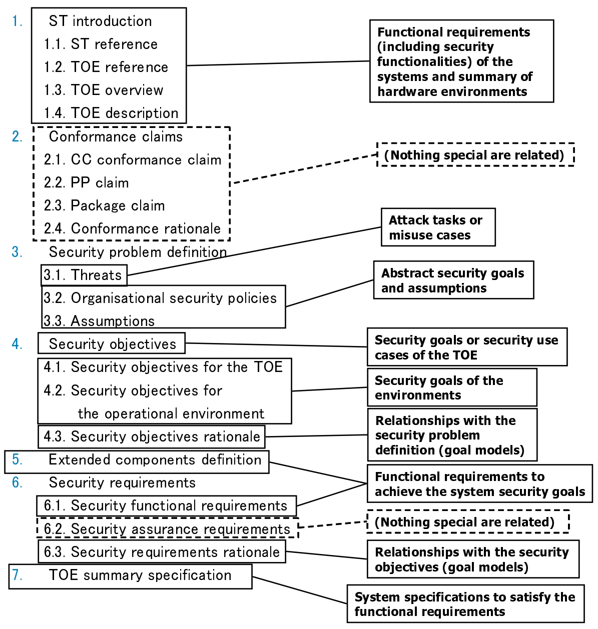
1. ST introduction
1.1. ST reference
1.2. TOE reference
1.3. TOE overview
1.4. TOE description
2. Conformance claims
2.1. CC conformance claim
2.2. PP claim
2.3. Package claim
2.4. Conformance rationale
3. Security problem definition
3.1. Threats
3.2. Organisational security policies
3.3. Assumptions
4. Security objectives
4.1. Security objectives for the TOE
4.2. Security objectives for the operational environment
4.3. Security objectives rationale
5. Extended components definition
6. Security requirements
6.1. Security functional requirements
6.2. Security assurance requirements
6.3. Security requirements rationale
7. TOE summary specification
Functional requirements (including security functionalities) of the systems and summary of hardware environments
(Nothing special are related)
Attack tasks or misuse cases
Abstract security goals and assumptions
Security goals or security use cases of the TOE
Security goals of the environments
Relationships with the security problem definition (goal models)
Functional requirements to achieve the system security goals
(Nothing special are related)
Relationships with the security objectives (goal models)
System specifications to satisfy the functional requirements

Fig. 3 Correspondence between ST and security requirements model.

- Users can access the documents
- The extent of users is specified for each document
- Each user can access the documents permitted to the user
- TOE is the document management system
  Other entities are considered environments.

Say we create the attack model shown in Fig. A·8 by using Liu's method.

From this model, we can write "3. Security problem definition" of the ST as follows.

3.1. Threats

- T.UNAUTHORIZED_ACCESS_TO_DOCUMENTS

  Attacker may access documents with no access permissions

Note that a prefix "T.", in which "T" is the first character of "threat", is attached to each item of this section to indicate that the item represents a threat. In similar ways, each prefix below consists of an abbreviation of the corresponding concept followed by a dot.

3.2. Organizational security policies: None

3.3. Assumptions: None

After that, we create a domain requirements model including security countermeasures (Figure A·9). The symbols such as *FMT_MSA_dot_3* represents SFRs adopted from CC Part 2. Although this SFR is denoted as FMT_MSA.3 in CC, we use "_dot_" to represent "." because ST-Tool cannot handle a dot.

From this model, we can write the following corresponding part of the ST.

4.1. Security objectives for the TOE

- O.ACCESS_CONTROL

  The extent of users is specified for each document

  Each user can access the documents permitted to her

4.3. Security objectives rationale

	T.UNAUTHORIZED _ACCESS_TO _DOCUMENTS
O.ACCESS_CONTROL	x

6.1. Security functional requirements

- FDP_ACC.2 Complete access control

  FDP_ACC.2.1: The TSF shall enforce the *ACL SFP (or any other access policy)* on *the users and the documents* and all operations among subjects and objects covered by the SFP.

  - The emphasized words are concrete assignments to the original CC descriptions.

  FDP_ACC.2.2: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

- FDP_ACF.1 Security attribute based access control

- FMT_MSA.3 Static attribute initialization

6.3. Security requirements rationale

	O.ACCESS_CONTROL
FDP_ACC.2	x
FDP_ACF.1	x
FMT_MSA.3	x

The students also work on a larger exercize using an example of cellular phones containing contactless IC chips with the electronic money facilities.

5 Future work

Aside from research on security policies in requirements engineering, there have been a number of attempts at integrating RBAC into the software modelling language UML. [16], [18], [19] A notable example is SecureUML [16], [17] by Basin, Doser and Lodderstedt.

Eliciting requirements on and modeling of access control policies are completely different issues. Class diagrams which model security policies will lose information on early requirements and how they are derived. Because of this deficiency it is very hard to maintain policies and to adapt them to even a small change in an organization or system. Hence it is essential to link the requirements and design phases.

Unfortunately, like any other software engineering program, our requirements analysis courses and system architecture courses are taught separately so that how they are linked is only dealt with in the final projects. We are planning to bridge this gap by developing a methodology that will enable us to transfer a model obtained in KAOS to a UML model.

There are issues regarding writing CC documents using security requirements analyses. The first is the treatment of PPs (protection profiles). As a PP is created for a somewhat general purpose in an application domain, writing one requires domain analysis. We also need to think about how to use PPs efficiently when writing the ST. The second issue is how to analyze SARs (security assurance requirements). Because an SAR is not a requirement for a system but instead is related to security assurance activities, we need a software process approach. Finally, considering that CC documents are formal ones and are targets of official inspection, their quality should be improved. A tool for supporting the writing of such documents would thus be useful.

6 Conclusion

As far as we are aware, there is no other educational course besides ours on how to elicit and analyze security requirements; hence it has been a challenge to design such a course. In this paper we presented how we designed such a course in the Top SE program and explained our security requirements analysis methodologies used in the course. We are still in the early stage of course development, and we have not fully assessed the effectiveness of the course yet. However, we must mention that we have received positive feedback from our students. We are improving the content of the course based on this feedback, and our future work will include a full assessment of the course.

Acknowledgements

We would like to thank Shinichi Honiden for his guidance on the course design. We would also like to thank the reviewers of the paper for their suggestions and comments on improving the draft.

The Top SE program is fully supported by Special Coordination Funds for Promoting Science and Technology, Fostering talent in an emerging research field by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] B. Boehm and V. R. Basili, "Software Defect Reduction Top 10 List," *IEEE Computer*, vol. 34, no. 1, pp. 135–137, 2001.

[2] S. Honiden, Y. Tahara, N. Yoshioka, K. Taguchi and H. Washizaki, "Top SE: Educating Superarchitects Who Can Apply Software Engineering Tools to Practical Development in Japan," *Proceedings of International Conference on Software Engineering* (*ICSE 2007*), pp. 708–718, IEEE, 2007.

[3] Top SE website, http://www.topse.jp.

[4] P. Giorgini, F. Massacci, J. Mylopoulos, A. Siena and N. Zanneno, "ST-Tool: A CASE Tool for Modeling and Analyisng Trust Requirements," *Proceedings of the Third International Conference on Trust Management* (*iTrust 2005*), LNCS, Spinger-Verlag, pp. 415–419, 2005.

[5] T. Tsumaki, H. Kaiya, Y. Tahara, N. Yoshioka, K. Taguchi and S. Honiden, "Errors and Misconceptions in Learning i*," *Proceedings of the 2nd International Workshop on Requirements Engineering Education and Training* (*REET 2007*), 2007.

[6] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn and R. Chandramouli, "Proposed NIST standard for role-based access control." *Proceeding of the NIST-NSA National Computer Security Conference*, pp. 554–563, 1992.

[7] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-Directed Requirements Acquisition." *Science of Computer Programming*, vol. 20, pp. 3–50, 1993.

[8] A. van Lamsweerde, R. Darimont and P. Massonet, "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt." *Proceedings of International Conference on Requirements Engineering* (*RE 1995*), pp. 194–203, IEEE, 1995.

[9] R. Darimont, E. Delor, P. Massonet and A. van Lamsweerde, "GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering," *Proceedings of International Conference on Software Engineering* (*ICSE 1998*), pp. 612–613, IEEE, 1998.

[10] A. van Lamsweerde, "Elaborating Security Requirements by Construction of Intentional Anti-Models," Proceedings of *International Conference on Software Engineering* (*ICSE 2004*), pp. 148–157, IEEE, 2004.

[11] A KAOS Tutorial, http://www.objectiver.com/. CEDITI sa, 2003.

[12] I. Sommerville, *Software Engineering*, Addision-Wesley, 2005.

[13] A. Schaad, J. Moffett and J. Jacob, "The Role-Based Access Control System of a European Bank: A Case Study and Discussion." *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies* (*SACMAT 2001*), pp. 3–9, ACM, 2001.

[14] J. S. Park, K. P. Costello, T. M. Neven and J. A. Diosomite, "A Composite RBAC Approach for Large, Complex Organizations," *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies* (*SACMAT 2004*), ACM, pp. 163–172, 2004.

[15] D. F. Ferraiolo, D. R. Kuhn and R. Chandramouli, *Role-Based Access Control.* Artech House 2003.

[16] T. Lodderstedt, D. Basin and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," *Proceedings of 5th International Conference on the Unified Modeling Language* (*UML 2002*), Springer, pp. 426–441, LNCS 2460, 2002.

[17] D. Basin, J. Doser and T. Lodderstedt, *Proceedings of 8th ACM Symposium on Access Control Models and Technologies* (*SACMAT 2003*), pp. 100-109, ACM, 2003.

[18] D-K. Kim, I. Ray, R. France and N. Li, "Modeling Role-Based Access Control Using Parameterized UML Models," *Proceedings of Fundamental Approaches to Software Engineering* (*FASE 2004*), pp. 180–193, LNCS 2984, 2004.

[19] P. Epstein and R. Sandhu, "Towards A UML Based Approach to Role Engineering," *Proceedings of 4th ACM Symposium on Role-Based Access Control* (*RBAC 1999*), ACM, pp. 135–143, 1999.

[20] Common criteria for information technology security evaluation part 1: Introduction and general model.

http://www.commoncriteriaportal.org/public/developer/index.php?menu=2, Sep. 2006.

[21] L. Liu, E. Yu, and J. Mylopoulos. "Security and privacy requirements analysis within a social setting" *Proceedings of International Conference on Requirements Engineering* (*RE 2003*), pp. 151–161, 2003.

[22] M. Rausand and A. Hϕyland. *System Reliability Theory; Models, Statistical Methods and Applications* (Second Ed.) Wiley, 2004.

[23] F. Redmill, M. Chudleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP*. John Wiley & Sons, 1999.

[24] E. Yu. "Towards modeling and reasoning support for early-phase requirements engineering." *Proceedings of the 3ed IEEE International Symposium on Requirements Engineering* (*RE 1997*), pp. 226–235, 1997.

Kenji TAGUCHI

Kenji TAGUCHI is a professor (by special appointment) at NII. He received his PhD in Computer Science from Uppsala University in 2001. His has been mainly working on integrations of different formal methods such as process algebras and state-based formal specification languages, Z and Object-Z. He jointly founded an international conference series IFM (Integrated Formal Methods) in 1999. He has been working for an education program called Top SE since he joined NII in 2005.

Yasuyuki TAHARA

Yasuyuki TAHARA is an associate professor in NII (by special appointment). He received his BSc and his MSc in Mathematics from the University of Tokyo, Japan, and his PhD in Information and Computer Science from Waseda University, Japan, in 1989, 1991, and 2003, respectively. He joined Toshiba Corporation in 1991. He was a visiting researcher in City University London, UK, from 1995 to 1996, and in Imperial College London, UK, from 1996 to 1997. He left Toshiba Corporation and joined NII in 2003. His research interests include formal verification of software and requirements engineering. He has been working for an education program called Top SE since 2004. Prof. Tahara is a member of the Information Processing Society of Japan and Japan Society for Software Science and Technology.

Appendix A: KAOS and i* models

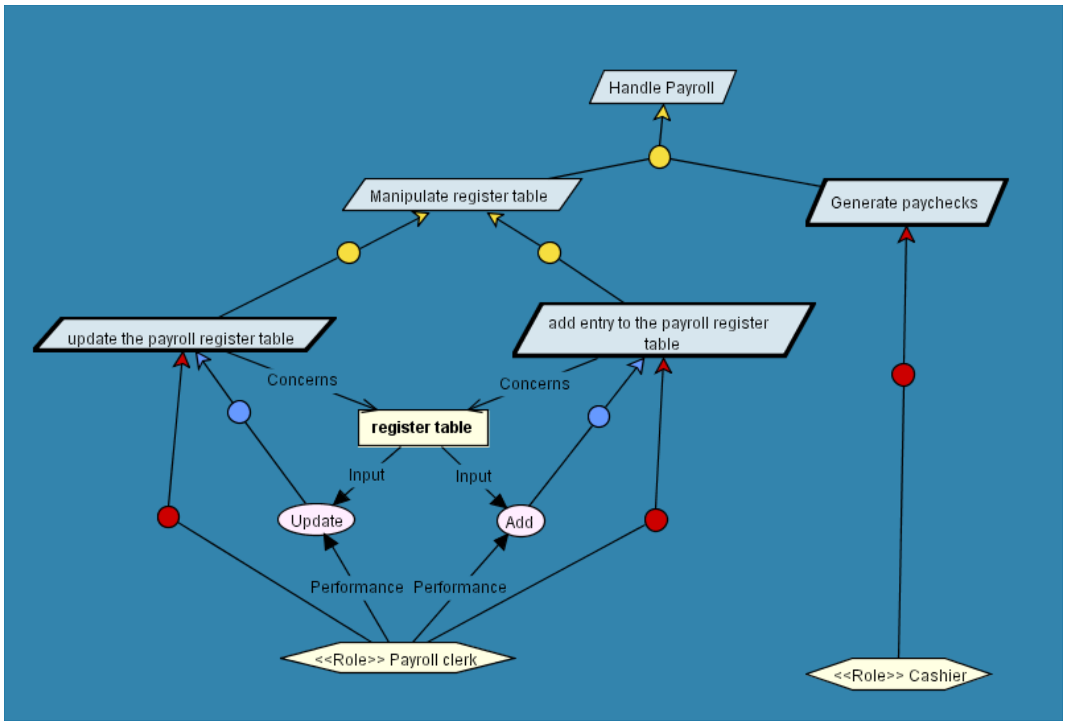

Fig. A·1 RBAC in KAOS.

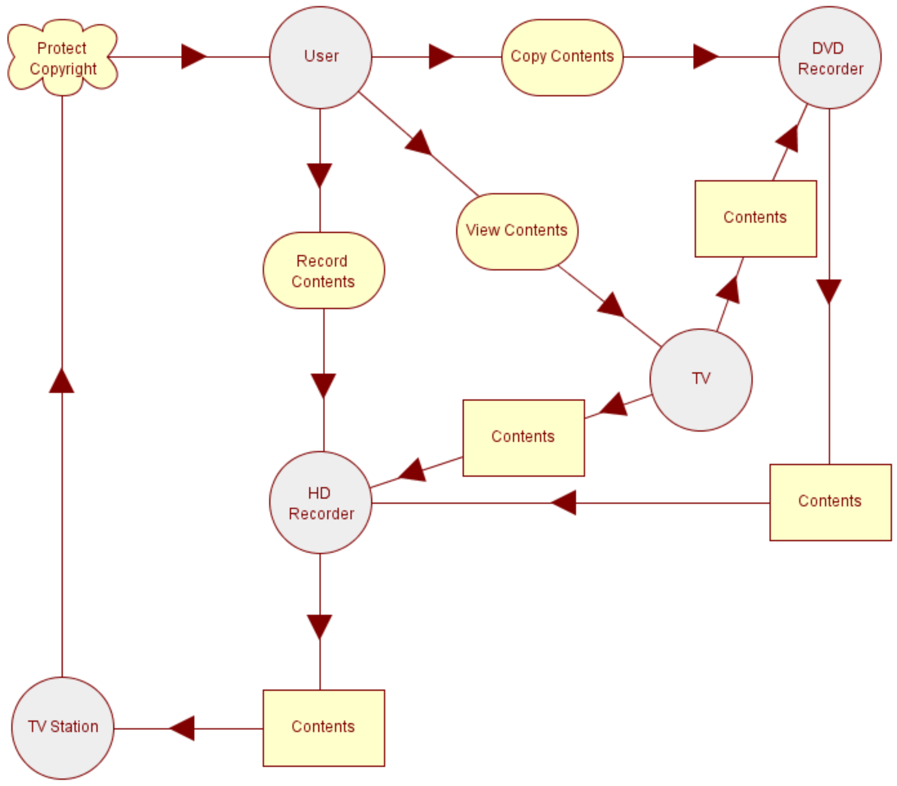

Fig. A·2 Domain SD model.

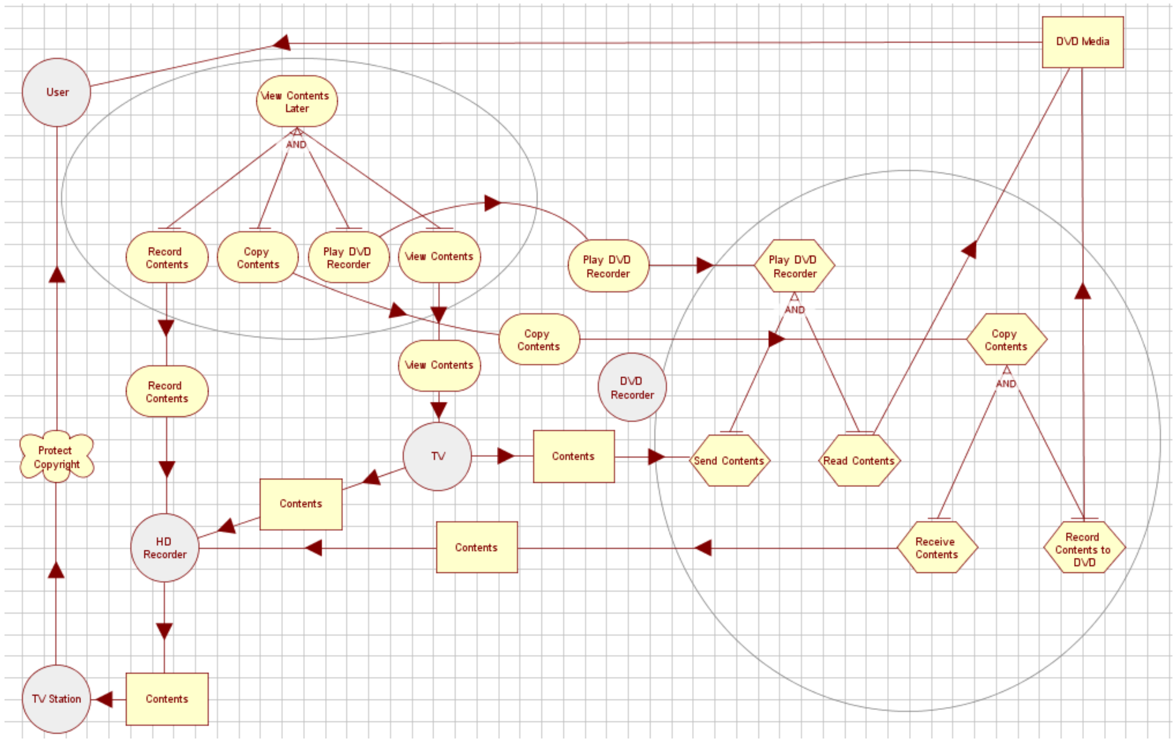

Fig. A·3 Domain SR model.

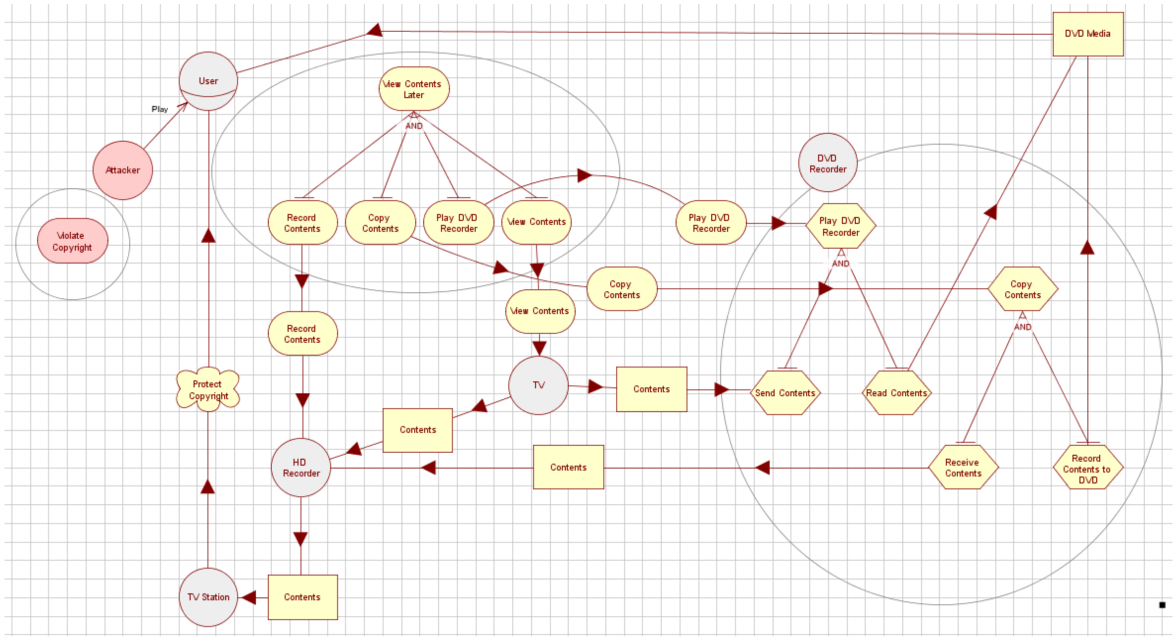

Fig. A·4 Introduction of attacker actor.

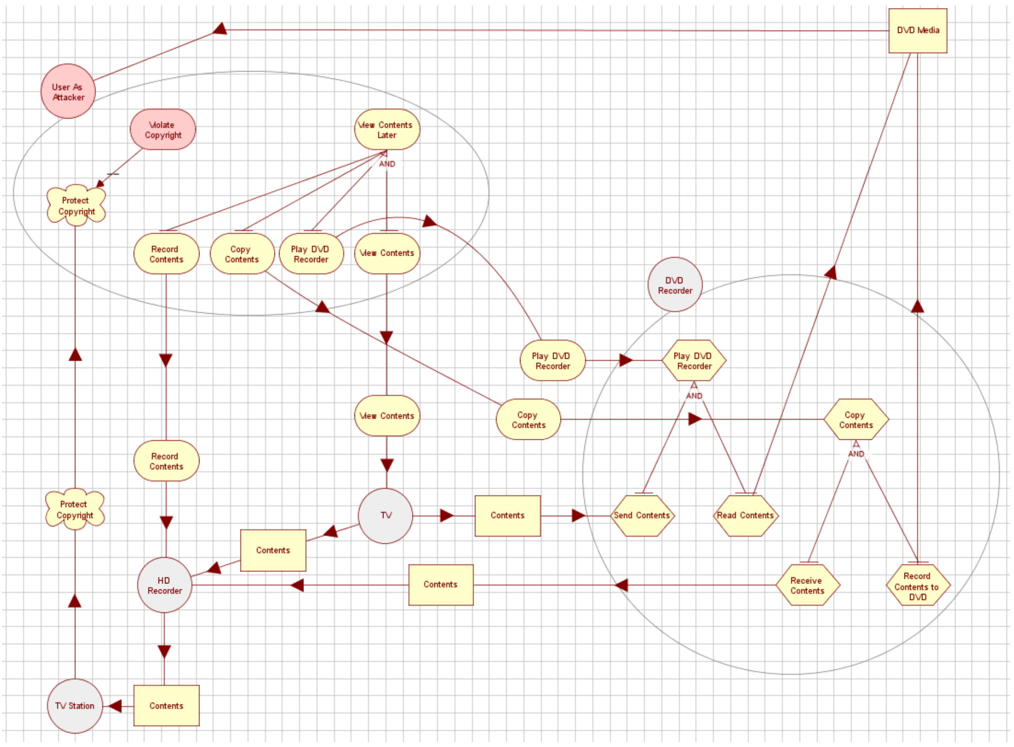

Fig. A·5 Attacker model.

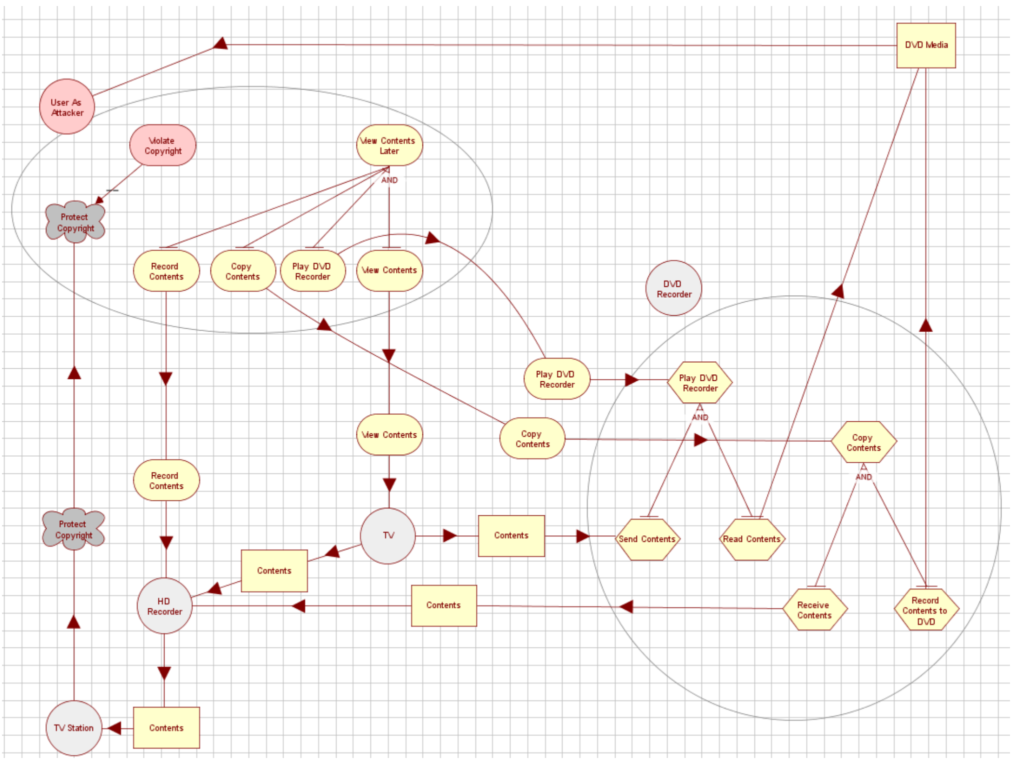

Fig. A·6 Vulnerability model.

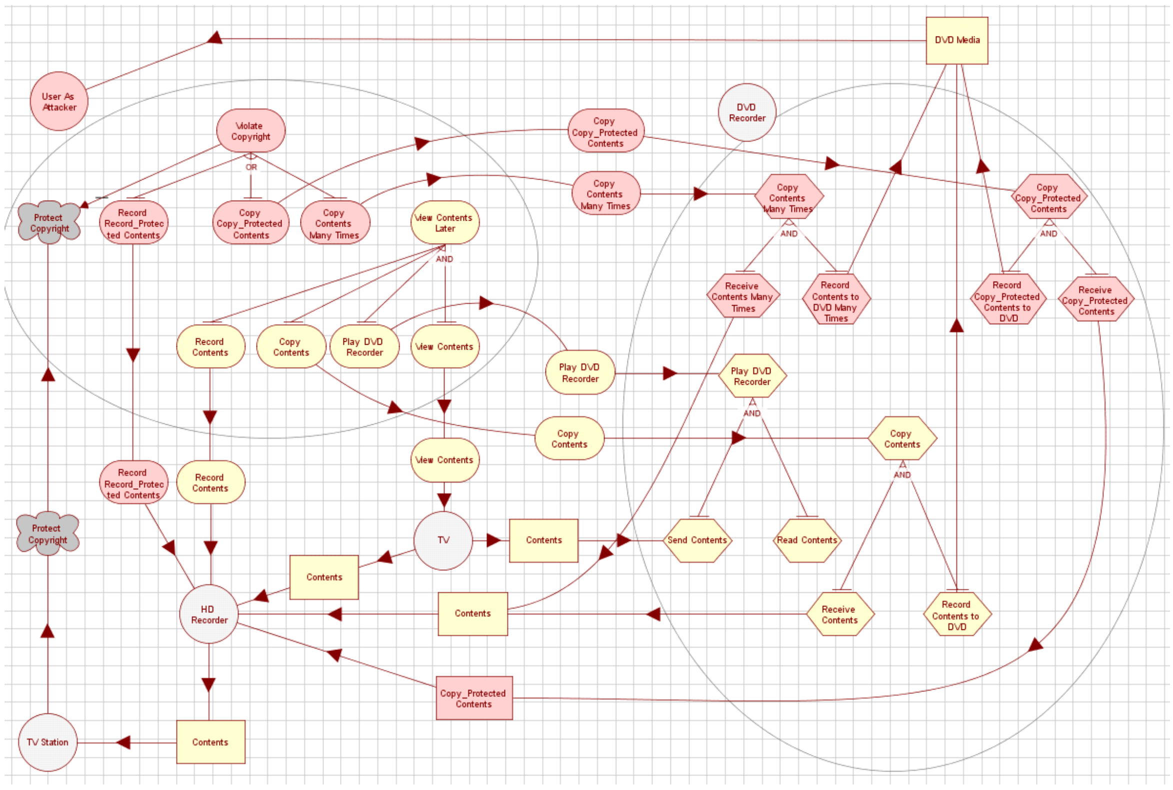

Fig. A·7 Attacking measure model.

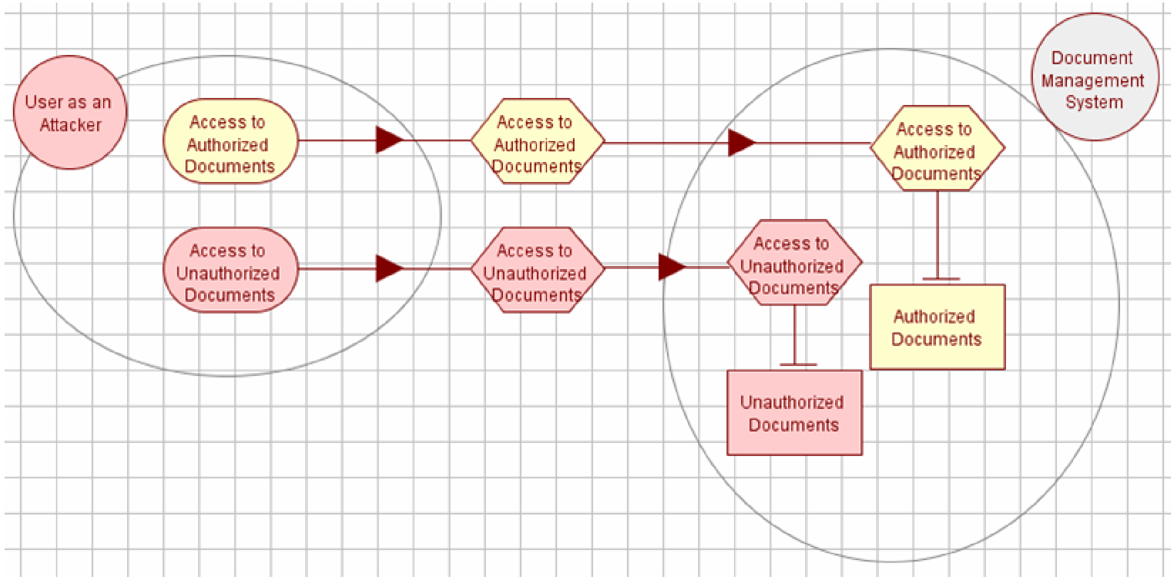
User as an Attacker
Access to Authorized Documents
Access to Authorized Documents
Access to Authorized Documents
Document Management System
Access to Unauthorized Documents
Access to Unauthorized Documents
Access to Unauthorized Documents
Authorized Documents
Unauthorized Documents

Fig. A·8 Attack model of document management system.

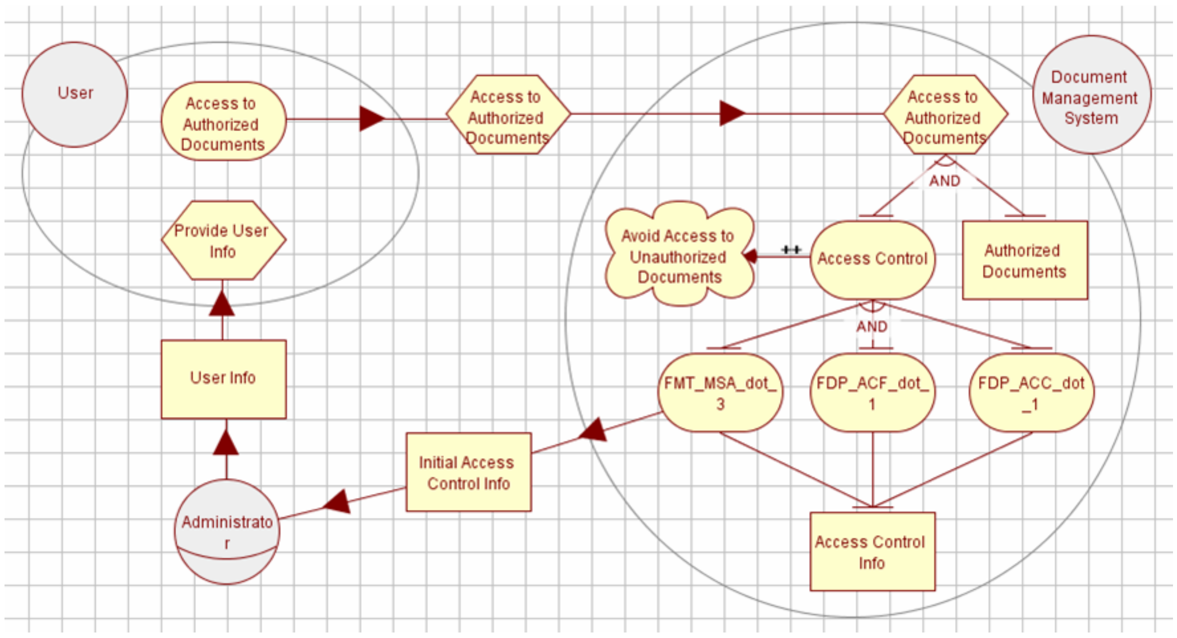
User
Access to Authorized Documents
Access to Authorized Documents
Access to Authorized Documents
Document Management System
AND
Provide User Info
Avoid Access to Unauthorized Documents
++
Access Control
Authorized Documents
AND
User Info
FMT_MSA_dot_3
FDP_ACF_dot_1
FDP_ACC_dot_1
Initial Access Control Info
Administrator
Access Control Info

Fig. A·9 Domain requirements model of document management system.