Using Common Criteria as Reusable Knowledge in Security Requirements Elicitation

Motoshi Saeki[1] and Haruhiko Kaiya[2]

[1] Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1-W8-83, Meguro-ku, Tokyo 152-8552, Japan
[2] Dept. of Computer Science, Shinshu University
Wakasato 4-17-1, Nagano 380-8553, Japan
saeki@se.cs.titech.ac.jp,kaiya@cs.shinshu-u.ac.jp

Abstract. The elicitation of security requirements (SRs) is a crucial issue to develop secure information systems of high quality. Although we have several methods mainly for functional requirements such as goal-oriented methods and use case modeling, most of them do not provide sufficient supports to identify threats, security objectives and security functions. Security functions are closely related to architectural design of the information system, i.e. solution space, and knowledge from the solution space is necessary to elicit appropriate SRs of higher quality. This paper proposes the usage of Common Criteria and related knowledge sources to identify SRs from functional requirements through eliciting threats and security objectives. Our proposed technique can be combined with and embedded into any existing functional requirements elicitation methods.

1 Introduction

Information systems deployed at different sites are being connected to each other through networks and their users can obtain various services anytime and anywhere. In this circumstance, it is very significant to protect assets in an information system from events and/or malicious actors that compromise their security and therefore to develop the information systems with functions that protect from so-called security threats.

In usual information system development like waterfall style, the requirements to an information system are elicited after a business process modeling stage. It is necessary to elicit the requirements to security functions (simply security requirements) as early as possible, in order to reduce the development cost and to develop the information system of higher quality [1]. Some techniques to elicit security have been proposed and put into practice, e.g. misuse case [2], abuse case [3], security use case [4], i* [5] and secure Tropos [6]. All of them are the extended versions of use case modeling and goal-oriented approaches, which are requirements elicitation ones originally for functional requirements, so that they can adapt to the elicitation of security requirements as one of the non-functional requirements. However, since security functions are closely related to system architecture design, i.e. artifacts on a solution space of the problems, thus it is difficult to elicit appropriate security requirements without considering the system architectures. For instance, let's consider the data base system that stores university students' grades and its functions for the students to access to their grades. There

is a potential of the threat that grade data of a student can be read by the others. The technique of password authentication can be adopted to mitigate the occurrences of this threat, so that the only student that is authenticated and identified can read her grade data from the data base system. Therefore, a file system of password data used for authentication and identification (password file) is newly adopted in the system and it stores pairs of student IDs and passwords. The malicious person illegally and furtively may read password data from the password file and impersonates other students to get their grade data when adopting such a technique. To mitigate this threat further, we can have a solution to encrypt the password data in the file. We can consider threats as concepts of a problem space, while the countermeasure techniques to mitigate the threats, e.g. password authentication, password file and cryptography are the concepts in a solution space of this problem domain. Thus, not only both of them are closely connected, but also a new threat (a problem) may be invented from the newly adopted solutions. This relation expresses just a twin-peak model that Bashar Nuseibeh mentioned [7]. Requirement elicitation activities both in a problem space and in a solution space is indispensable to appropriate security requirements elicitation, as shown in Figure 1. The existing studies are biased to requirements elicitation on the problem space side, and there are quite few studies that both sides are simultaneously considered.

In this paper, we propose an integrated security requirements elicitation method that uses Common Criteria Part 2 [8] as a guideline or knowledge source while requirements elicitation on the solution space side is being performed, because the Common Criteria consists of general concepts in the solution space and it can be considered as a kind of catalog that provides threats, security objectives and security functions that have generally appeared. For example, by using Common Criteria, we can select the objective "data encryption" from the catalog, to mitigate the threat "disclosure of password data". Furthermore, from the catalog we can select more concrete security functions, e.g. the procedures to generate encryption keys and to abandon them, in order to achieve the selected objectives. Our technique is for combining and embedding the usage of Common Criteria to any existing functional requirements elicitation methods such as goal-oriented approaches and use case modeling ones.

The rest of the paper is organized as follows. We explain Common Criteria and describe how it can be used for security requirements elicitation in a solution space in section 2. Section 3 shows the overview of the proposed method and section 4 depicts an example. The related work and future research agenda are presented in sections 5 and 6 respectively.

2 Basic Idea

2.1 Common Criteria

The Common Criteria (CC) is an international standard prescribing how to write the documents that are used for assessing security properties of the information system called TOE (target of evaluation). The produced document includes two types of documents; one document called *security target* is to specify security properties of the TOE, and another is to describe security assurance requirements used for verifying the compliance of the TOE product with the security properties. In this paper, we consider the

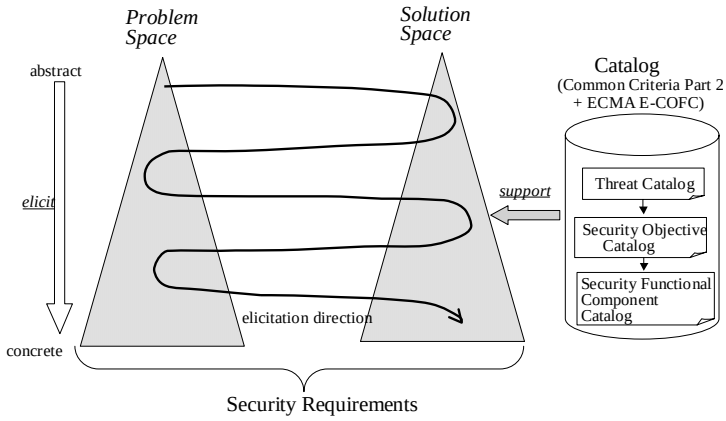


Fig. 1. Twin Peak Model in Security Requirements Elicitation

former document, i.e. a security target. The CC itself consists of three parts, and we use its Part 2 as knowledge source of requirement elicitation.

A security target basically consists of 6 chapters. It begins with the overview of the TOE in chapter 1 and then describes its functional requirements in chapter 2. The description of functional requirements includes the information on the assets to be protected. In the example of reading students' grades mentioned in the previous section, the assets to be protected are grade data of students. Potentials of threats and assumption, etc. are described in the successive chapter 3 "Security Problem Definition". In the example of the database system of students' grades, we can consider identity spoofing by impersonation as a potential of threat. The chapter 4 "Security Objectives" provides for the objectives to mitigate the threats listed in the chapter 3. To mitigate the threat "impersonate", authentication by password can be given as a security objective. Note that each security objective described in chapter 4 should be linked to the threats in chapter 3, to clarify which objective mitigate which threats. In the chapter 5 "Security Requirements", the components of security functions to represent *exactly* the security objectives listed up in chapter 4 are described. In CC Part 2, security functional (SF) components are catalogued beforehand in the form of template. This is the reason why we use CC Part 2, a collection of SF components and their relationships, as knowledge source in our approach. We can select appropriate templates from the CC Part 2 catalog and instantiate them by filling their slots with relevant information. We represent the security objective as the combination of the instantiations of the selected templates in chapter 5. For instance, the template class FIA is for representing the security objective "identification and authentication", and it has the templates of the functional requirements descriptions relevant to "identification and authentication" such as authentication failure, user authentication, user identification, user attribute definition, etc. In chapter 6, the last chapter, we specify the security functions by logically combining and summarizing the component descriptions selected and instantiated in the chapter 5. It is necessary to maintain traceability between threats, security objectives, SF components

and the specification statements of security functions, as will be mentioned in next sub section.

2.2 Catalog

Although CC Part 2 has about 120 SF components as a catalog, it has no catalogs of threats and security objective. Thus requirements analysts should freely write threats and security objectives by themselves. ECMA-271 E-COFC [9], which can be considered as a profile of CC in a certain problem domain, includes the catalogs of threats and security objectives. In this paper, we use them together with CC Part 2. As shown in the left hand side of Figure 1, we accumulate a threat catalog, a security objective catalog and a SF component catalog, and hold relationships between their catalog entries (i.e. security objective *mitigates* threat, SF component *represents* security objective). After a threat is identified, requirements analysts should perform the two tasks; 1) deriving the security objectives that the threats can mitigate, and 2) identifying the SF components that can represent the derived security objectives. These two tasks can be considered as those in a solution space and be supported by using our approach.

For example, suppose that the analyst selected a threat "Impersonation" for students (T.impersonate in the catalog) from the threat catalog. To mitigate it, the relationship between the threat catalog and the security objective catalog suggests that O.Authentication (authentication for students as authorized users) and O.Integrity (protection of integrity of authentication data) should be selected from the security objective catalog. Furthermore, the analyst can select SF components (templates) of FIA class (User Identification and Authentication) in order to refine and represent the O.Authentication, by using the relationship *represent* between the security objective catalog and the SF component's. She fills the slots with relevant information, e.g. the acceptable time of authentication failures and the actions to be performed when the failures exceed to an acceptable value (e.g. ringing an alarm etc.) in the templates FIA_AFL family. And then she completes the document of security functional requirements.

3 Overview of the Elicitation Method

Our proposed method is shown as follows, and Figure 2 also shows its task flow with a simple example.

Step1. Identify functional requirements (FRs) :
An analyst elicits functional requirements of the information system by using the existing elicitation methods such as goal-oriented methods and use case modeling. As an example, she uses a simplified version of goal-oriented method. As shown in the figure, she decomposes the root goal "Checking grades" for the database system of students' grades and decomposes it into two sub goals "Retrieving grades" (a student retrieves his grade data) and "Getting grades" (a student gets his grade data from the database as a result of retrieval) with AND decomposition.

Step2. Identify assets and their attributes :
The analyst identifies from the functional requirements the assets to be protected.

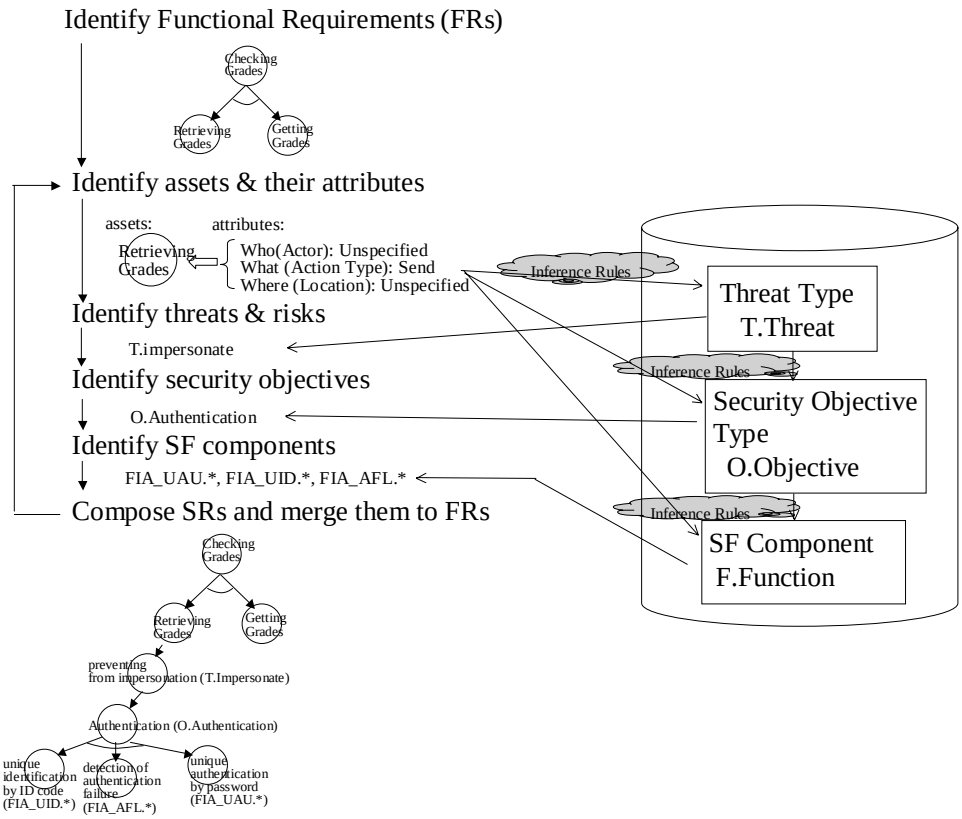


Fig. 2. Elicitation Method

In the case of using a goal-oriented method only, the assets can be extracted from goals and words appearing in the goal descriptions written in natural language. On the other hand, when using use case modeling method, assets can be elicited from constructs of use case diagrams, e.g. actors and use cases, and the words in use case descriptions. For each of the identified assets, the analyst attaches the attributes to it. Generally, we use 5W1H (Who, What, When, Where, Why and How) as the attributes to characterize the assets. The attributes to be used depend on a problem domain. In this example, the analyst focuses on the operationalized sub goal "Retrieving grades" as the asset to be protected, and its attributes are the actor who perform it (unspecified person), the action type of what the actor performs (sending retrieval information to the system) and the location where the actor does (a student can perform at unspecified place).

Step3. Identify threats & risks : The analyst infers and derives the potentials of threats from the attribute values of the assets and the threat catalog by means of inference rules prepared beforehand. In this example, since the actor of the asset (goal "Re-

trieving Grades") was unspecified (persons) and the location was also unspecified, she selects T.imperonate using the catalog and an inference rule. The inference rules specify the relationships among possible threats and the attribute values of the assets. For the above example, the analyst used the following rule of "if then" style to get T.impersonate.

if who(x) = unspecified ∧ where(actor(x)) = unspecified
then select T.impersonate(actor(x)), where x is an asset.

Step4. Identify security objectives : The analyst selects suitable security objectives from the security objective catalog, using the identified threats and the asset attribute values. In this example, she selected O.Authentication from the catalog, by applying the inference rule like:

if who(x) = unspecified ∧ T.impersonate **then** select O.Authentication(actor(x)).

This rule suggests that we could adopt the authentication technique in order to prevent some actors from retrieving the grade data of the other students by means of spoofing and impersonation.

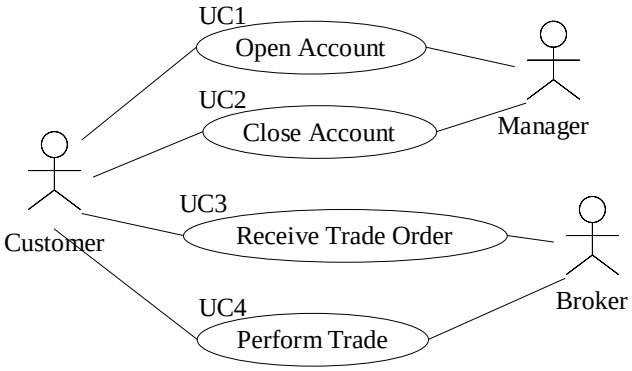
Step5. Identify security functional (SF) components :
The analyst selects the set of the candidate SF components using the identified security objectives and the asset attribute values. This step is similar to the former steps 3 and 4, i.e. the predefined inference rules support the selection of SF components. In this example, the analyst selects FIA_UAU, which is a family of the functional components for user authentication. Since these components have the dependencies to FIA_UID (the functions for user identification) and to FIA_AFL (the functions for measures to authentication failures), some of them are selected to refine the security objective appropriately. Dependency relationships among SF components are very helpful to avoid missing security functional requirements.

Step6. Compose security requirements (SRs) and merge them into FRs :
The analyst combines the identified SF components logically and makes the document of security functional requirements from them. This document explains for the customers how the described functions can mitigate the threats and achieve the security objectives. In this example, since the analyst uses the goal-oriented approach, she adds these identified elements (threats, security objectives and SF components) as the sub goals of the asset to be protected, i.e. "Retrieving Grades". As shown in the bottom of Figure 2, the sub goals corresponding to T.Impersonate, O.Authentication and FIA_UAU are successively added to "Retrieving Grades". FIA_UID and FIA_AFL are also added with AND decomposition in the same level because of their dependencies to FIA_UAU. To maintain traceability, these elements as sub goals and their relationships are kept in the goal graph.

For the newly added security requirements, the analyst iterates from step 2 to step 6 and elicits new requirements. In the example of the figure, after eliciting a sub goal "unique authentication by password" from FIA_UAU, she identifies passwords as the asset to be protected at the iterated step 2, then identifies the threat T.Data_Theft (password data may be stolen) in the same way, and continues her elicitation tasks.

4 Example

We illustrate another example where use case modeling is applied in order to show our approach can be combined to various existing requirements elicitation methods, not only goal-oriented approaches. The example in this section is the system of a financial company that provides investment services to its customers, which was included in [10]. Figure 3 shows the overall of the system with a use case diagram and a part of the use case descriptions.



UC1 : Open Account
1. Manager gets the personal information from Customer
e.g. name, telephone number, etc.
2. Manager inputs the personal information of Customer to System.
3. System issues Customer ID and Password to Manager.
4. Manager inform Customer ID and Password to Customer
5. ... (to be continued)

UC3 : Receive Trade Order
1. Customer inputs ID and password to System.
2. System authorizes right of Trade Order to Customer.
3. Customer specifies commodities and their amounts which he wants to trade.
4. ... (to be continued)

Fig. 3. Use Case Diagram and Descriptions

This system includes many potentials of threats and some of them were pointed out in [10]. In this example, out of them, we pick up and pay attention to the threat “the manager can create a spurious authorization to access the account”. More concretely, since the manager can know the identification & authentication information (ID and

Password) of the customer when the customer opens his account (with UC1), a malicious manager can be spuriously authorized using the customer's ID and password, and can trade orders by impersonating as the customer. We apply our approach to this security problem.

Step1. Identify functional requirements

As shown in Figure 3, we elicit the functions of the system with use case modeling. The security problem results from the case where the customer can ask the manager to open his account, not by himself. The action 3 of the UC1 suggests that the manager could obtain the ID and the password of the customer and then impersonate the customer using UC3 with the obtained ID and password.

Step2. Identify assets and their attributes

We identify the asset to be protected from the use case diagram as UC3 "Receive Trade Order", and find its attributes as follows.

who (who performs): Customer, Broker
what (what is performed): Input, Authorize, Specify, ...
how (how it is performed) : Identify and Authenticate by Customer's ID and password
where: (where it is performed) : unspecified
when : (when it is performed) : after opening an account
why: (why it is performed) : Customer gets trades

Step3. Identify threats and risks

In this example, we use a word-matching technique instead of rigorous "if-then" inference rules used in the previous example of section 3. The attribute values are set based on the words appearing in the use case description of UC3, e.g. authorize, input, specify, etc., and we match these words with the words appearing in the catalogs. More concretely, we look for the entries whose explanation sentences include the synonyms as the attributes, i.e. Input, Authorizes, etc. Concentrating on the word "Authorize" and its flections (we use a wild card to express the word and its inflection, like authoriz*) of the "what" attribute, we can obtain the following threats from the threat catalog of ECMA-271 (E-COFC) because the explanations of all of them have authoriz*.

T.Insider, T.Outsider, T.Secret_Disclose.

Step4. Identify security objectives

In the similar way, we can get OE.Access_Malicious from the security objective catalog. In addition, we can get the followings by using the "what" attribute authoriz* and "how" attribute values identify* and authentic*.

O.Authen_Address, O.Authen_Age, O.Authen_Indep, O.Authen_Protect.

Step5. Identify security functional components

Since OE.Access_Malicious is an environmental security objective and it means that the malicious activities of the manager cause the threat, the SF components included in CC cannot mitigate it. However, we can select the FIA_UAU.3.2 to mitigate the other security objectives by using the word matching with authentic*.

Step6. Compose security requirements and merge them into FRs

The use case corresponding to FIA_UAU 3.2 is newly added as a sub use case of UC3.

Note that we used the inference rules different from the goal-oriented approach example of section 3. The reason is that use case descriptions are written in natural

language and we consider that the word-match technique would be simpler and more suitable. It suggests that the rule styles, in addition to the assets, vary on the adopted requirements elicitation method.

5 Related Work

Although CC is used to assess whether the security properties of the IT products reach a certain standard level or not, it can provide reusable knowledge for security requirements elicitation such as SF component catalog. Our technique focuses just on this point. The approach proposed by Ware et. al. was the first one to use CC as a reusable catalog to support the elicitation of security requirements [11]. In their approach, an analyst constructs a profile for each actor after drawing a use case diagram and correlates the actors to threats based on the actors' profiles only. Although our approach can be considered as its more elaborated version, it can deal with wider requirements elicitation methods and their concepts, not only actors in use case modeling but also use case, goals in goal-oriented approach, etc. And it can elicit security objectives and SF components besides threats. Furthermore, we adopt an iterative process based on a twin peak model and our method is more sophisticated rather than Ware's approach.

Liu et al. and Mouratidis et al. proposed the usage of the concept of soft goals in goal-oriented approaches such as i* and Secure Tropos. However, their approaches did not include the guidelines or the methodologies to assist in identifying threats and security objectives, and in refining security-related goals in a graph. Thus their approaches can be considered only as the detailed version of the step 6 of our approach.

The approaches using misuse cases and anti-goals may be helpful to identify threats and can correspond to the steps 2 and 3 of our approach. However, their supporting techniques of refining and decomposing use cases or goals are not so powerful to derive countermeasures after identifying threats. That is to say, any of them did not consider the powerful supports for the activities of the right-hand side of Figure 1.

Industries have tried to extract and structure knowledge included in CC so as to assist in eliciting security requirements. For example, a STF (specialist task force) in ETSI has worked on this aim since 2003 [12]. The structured and classified security knowledge that it has produced can be helpful to make our method more sophisticated, and our approach can be considered as a bridge between its outcomes and existing requirements elicitation methods.

6 Research Agenda

This paper proposes the technique to use CC as reusable knowledge within the existing requirements elicitation methods in order to elicit security requirements. And it helps the harmonization of requirements elicitation between a problem space and a solution one, following a twin-peak model. However, there are several unsolved research agenda to be tackled. Firstly, the attributes attached to assets and the inference rules should be explored and elaborated. They may depend on the adopted functional requirements elicitation methods. Secondly, as mentioned in section 4, the usage of the words appearing in CC would be promising to select the catalogued threats, security objectives

and SF components. That is to say, we will investigate the application of ontological approaches, e.g. words as ontological concepts and the selections as ontological inference respectively. Thirdly, we also consider that the other types of reusable resources for security requirements, e.g. various levels of security patterns [13], attack patterns of SAFE-T [14] and Microsoft's STRIDE catalog could be integrated with our approach. Lastly, we should have more practical case studies to evaluate our technique.

Acknowledgements

The authors gratefully thank anonymous reviewers for their valuable and helpful comments.

References

1. C. Haley, J. Moffett, and B. Nuseibeh. Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Trans. on Software Engineering*, 34(1):133–153, 2008.
2. G. Sindre and A. Opdahl. Eliciting Security Requirements with Misuse Cases. *Requirements Engineering*, 10(1):34–44, 2005.
3. J. McDermott and C. Fox. Using Abuse Case Models for Security Requirements Analysis. In *Proc. of the 15th Annual Computer Security Applications Conference*, pages 55–64, 1999.
4. D. Firesmith. Security Use Cases. *Journal of Object Technology*, 2(3):53–64, 2003.
5. L. Liu, E. Yu, and J. Mylopoulos. Security and Privacy Requirements Analysis with a Social Setting. In *Proc. of the 11th IEEE International Requirements Engineering Conference*, pages 151–161, 2003.
6. H. Mouratidis and P. Giorgini. Secure Tropos: A Security-oriented Extension of the Tropos Methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):285–309, 2007.
7. B. Nuseibeh. Weaving Together Requirements and Architectures. *IEEE Computer*, 34(3):115–117, 2001.
8. Official CC/CEM Versions - The Common Criteria Portal. http://www.commoncriteriaportal.org/thecc.html, 2007.
9. ECMA-271: Extended Commercially Oriented Functionality Class for Security Evaluation (E-COFC). http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-271.pdf, 1999.
10. E. Fernandez, M. Larrondo-Petrie, T. Sorgente, and N. Vanhilst. Methodology to Develop Secure Systems Using Patterns. In H. Mouratidis and P. Giorgini, editors, *Integrating Security And Software Engineering: Advances And Future Vision*, pages 107–126, 2006.
11. M. Ware, J. Bowles, and C. Eastman. Using the Common Criteria to Elicit Security Requirements with Use Cases. In *Proc. of the IEEE Southeast Conference*, pages 273–278, 2005.
12. TISPAN security: Adoption of Common Criteria in security evaluation. http://portal.etsi.org, 2003.
13. T. Heyman, K. Yskout, R. Scandariato, and W. Joosen. An Analysis of the Security Patterns Landscape. In *Proc. of the 3rd International Workshop on Software Engineering for Secure Systems*, 2007.
14. M. Gegick and L. Williams. Matching Attack Patterns to Security Vulnerabilities in Software-intensive Designs. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7, 2005.