Requirements Refinement and Exploration of Architecture for Security and Other NFRs

Takao Okubo[1], Nobukazu Yoshioka[2], and Haruhiko Kaiya[3]

[1] Institute of Information Security, 2-14-1 Tsuruyamachi, Kanagawa-ku, Yokohama, Japan
[2] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
[3] Kanagawa University, 2946 Tsuchiya, Hiratsuka-shi, Kanagawa-ken, Japan

Abstract. Earlier software architecture design is essential particularly when it comes to security concerns, since security risks, requirements and architectures are all closely interrelated and interacting. We have proposed the security driven twin peaks method with a mutual refinement of the requirements, and architectures. However, there are multiple alternatives to an architecture design for initial requirements, and their choices depend on non-functional requirements (NFRs), such as security, performance, and costs which have a big impact on the quality of the software. We propose a new method called TPM-SA2 to avoid any back-track in refinement. Each architectural alternative in TPM-SA2 is refined so that it aligns with the requirements. For each refinement, the requirements can be updated vice versa. TPM-SA2 enables us to predict the impacts on the NFRs by each candidate for the architecture, and choose the most appropriate one with respect to the impact. As a result, we can define the requirements and architectures, and estimated the development costs earlier than ever.

1 Introduction

Security requirements analyses are often caught in a dilemma. Although precise and comprehensive threat analysis needs architectural design in detail, it is hard to obtain decomposed architectural design specification in the early requirement analysis stage. The pure water fall model of software development is not enough, so that we need to elaborate on the security requirements and the architecture simultaneously. The twin peaks model [12] is a promising alternative for such development styles involved in security requirements analysis.

In addition, it is also difficult to simultaneously consider several kinds of non-functional requirements for selecting an appropriate software architecture. There is a trade-off among different non-functional requirements, e.g., security may affect the performance and usability, and the performance is generally related to the scalability. We need to find suitable architectures to meet such non-functional requirements with priority.

We have proposed a concept of security oriented twin peaks model, called the Twin Peaks Model application for Security Analysis (TPM-SA) [14].

L. Iliadis, M. Papazoglou, and K. Pohl (Eds.): CAiSE 2014 Workshops, LNBIP 178, pp. 286–298, 2014.

Although we evaluated our method using some examples in our previous work, the evaluation revealed the following practical issues:

1. The boundary between the requirements and the design is unclear;
2. Although there are generally two or more architectural alternatives, it is hard to select the best architecture without conducting a detailed design analysis;
3. There is no criterion for evaluating the architectural alternatives; and
4. We need a method for considering other non-functional requirements such as the performance requirements.

Thus we proposed a new requirements analysis method called TPM-SA2 (Twin-Peaks Model application for Security Analysis, Square) [15]. We can analyze two or more non-functional requirements and the architecture with a twin peaks model using this method. We can, then, select the best architecture based on an estimation of the impact on the non-functional requirements. In other words, our major contributions are an elicitation process for two or more requirements, and the evaluation framework for the architectural alternatives from the multi-requirements points of views.

In this paper we do detailed analysis and evaluation of our TPM-SA2 with an example of software on a geolocation service.

This paper is organized as follows. Section 2 discusses techniques including our previous work. Section 3 describes our new method based on the Twin Peaks model, and Section 4 illustrates an example to which we applied our method. Section 5 discusses the results from using our method, and Section 6 reviews the related works. Finally, we summarize the paper and discuss our future work.

2 Related Work

This section describes related works on analysis of security requirements and architecture. At first, we introduce our recemt work called TPM-SA, and then compare with other works.

2.1 TPM-SA: Twin Peaks Model for Security Analysis

We proposed mutual refinement process between security requirements and architecture called TPM-SA [14]. TPM-SA is based on the Twin Peaks Model (TPM) [12].

The TPM-SA offers a developing framework for two-level structures for adopting TPM for the security. The first level defines the entire process in the form of a spiral that achieves mutual and stepwise refinement of the security requirements and architecture. The second level defines each cycle of the spiral that requires the detailed steps for the security analysis. The term “architecture” in this paper includes the system structure, software structure, design specifications, infrastructure, middleware, and the programming language.

The security requirements (countermeasure) and the architecture are refined in the spiral flow stages with the iteration of the security analysis and design. This process is assumed to be conducted after the functional requirements have been elicited. Therefore, the inputs for this process are the functional requirements. In the first cycle of the spiral, the functional requirements are the inputs. The first refinement level of the architecture can be assumed from the functional requirements. A security analysis is conducted with the help of the assumed architectural information. It helps in imagining the architecture-specific assets and attacks. As a result, the security requirements are outputs at the first refinement level. In the subsequent cycles, the security requirements that were outputs in the previous cycle are the inputs, and then more refined security requirements and architectures are the outputs. The main motivation of the "refinement" in the TPM-SA is to add some new requirements by taking the architecture into considerations, and adding a new architecture from these requirements. This is the essential difference between the original TPM and Security Twin Peaks [5]. Their refinement mainly focuses on decomposing the already identified goals and requirements, and does not focus on adding new elements.

For the steps in each cycle of the spiral, we classified the artifacts into two groups, "Requirements" and "Architecture". We also classified assets into two groups: Architecture-independent Assets (AIA) and Architecture-specific Assets (ASA).

2.2 Other Works

We use the following criteria for comparing the methods that contribute to simultaneously refining security requirements and architecture.

(C1). A method that enables us to update the requirements and architecture simultaneously.
(C2). A method that can be performed step by step based on the architectural decisions.
(C3). A method that enables us to comprehensively find the security requirements based on the analysis of the architecture independent assets (AIA) in addition to the architecture specific assets (ASA).
(C4). A method that enables us to maintain the traceability among the assets, threats, and countermeasures.
(C5). A method that enables us to analyze multiple quality requirements such as the usability, reliability, and security.
(C6). A method that helps us to decide when we may stop refining the requirements and architecture in the early stages of system development, i.e., the requirements definition stage.
(C7). A method that enables us to find and compare several alternatives for refining the requirements and architecture.

Table 1. Comparison of methods based on criteria

method \ criteria	(C1)	(C2)	(C3)	(C4)	(C5)	(C6)	(C7)
TPM-SA2	√	√	√	√	√	√	√
TMP-SA	√	√	√	√			
GORA				√	√		
KB				√	√		
EXP			√				
PATTERN				√			
SQUARE		√		√			
STP	√	√		√			
LISA	√						
ATAM		√			√		

The acronyms in Table 1 denote the following methods.

- TPM-SA2: Method proposed in this paper.
- TMP-SA: Previous version of our method for refining simultaneously the security requirements and architecture [14].
- GORA: Goal-oriented requirements analysis [8] [9] [11] [18].
- KB: Knowledge based approaches [17] [6].
- EXP: Exploring exceptions [9] [4].
- PATTERN: Pattern-based approaches [2] [3].
- SQUARE: System Quality Requirements Engineering [10].
- STP: Security twin peaks model [5].
- LISA: Integration of requirements and design decisions using architectural representation [19].
- ATAM: Architecture Tradeoff Analysis Method [1] [7].

As listed in Table 1, TPM-SA2 is developed to meet the fifth, sixth and seventh criteria above.

3 Revised TPM-SA: TPM-SA2

We have proposed a security driven refinement method for the requirements and architecture that enables for the early and appropriate architecture selection from multiple alternatives [15]. The reason why we mainly focused on security is that it has greater, more unintended and unavoidable impacts on the software. Most of the other NFRs like the usability and performance are predictable, and if some problems are found in the later stage they may be acceptable. However, most of the security problems found in the later stages are critical and force the stakeholders to remedy them.

3.1 Steps in TPM-SA2

TPM-SA2 lets the developers predict the refined requirements and architectural designs in the first refinement level. It also lets them estimate the impact on the

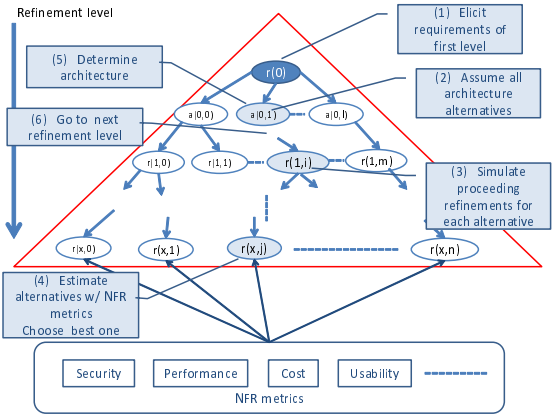

Fig. 1. Prediction Process in TPM-SA2

NFRs for each alternative. This step provides a solution to the criteria (expressly (C5)-(C7)) mentioned in Section 2.2; a reduction of redundant backtracking in refinements and the consideration of other NFRs.

for (C5). With TPM-SA2 analysts can estimate multiple NFRs for each designs.
for (C6). TPM-SA2 provides analysts rough estimate of NFR impacts for each architecture at the requirements analysis stage.
for(C7). With TPM-SA2 analysts can compare the architecture choices with estimating the NFRs impacts.

The inputs and outputs of TPM-SA2 are as follows:

- Inputs: functional requirements and non-functional requirements (NFRs). The NFRs consist of the quality requirements, design and architectural constraints.
- Outputs: refined requirements and intended architecture.

Figure 1 indicates the process for predicting in the architecture assumption steps of TPM-SA2. The process can provide analysts a rough estimate of the effect of architectural choices on security, NFRs and costs in advance.

Step 1). Obtain the requirements from the first refinement level 0 ($r(0)$). The inputs above are these requirements.
Step 2). Assume the architectural alternatives $a(0,0)$, $a(0,1)$,.. $a(0,n)$ which meet $r(0)$. Steps 1) and 2) are the preparation steps of the prediction process.
Step 3). Apply the mutual refinement in TPM-SA to each alternative. TMP-SA is explained in 2.1. Since each alternative can cause refinement and/or modification of $r(0,0)$, each alternative is related to different requirements.

Step 4). Evaluate each architectural alternative in the final level x with respect to the NFRs for the requirements corresponding to the alternative. The evaluation criteria such as the metrics should be prepared for each NFR. Choose the best one after the evaluation. The developer will follow the branch path that contains the leaf with the best NFR value in the following refinement steps.

Step 5). Determine the architecture that includes the best leaf chosen in Step 4) within its branch path.

Each path from $r(0)$ to $r(x, i)$ in Figure 1 corresponds to the mutual refinement of the requirements and architectural alternative. Our previous method TPM-SA [14] can be used for this refinement.

3.2 Prioritizing Alternatives with Respect to NFRs

As to the steps in Section 3.1, several different architectural alternatives are obtained. For each architectural alternative, initial requirements are respectively refined and/or updated. We then prioritize these alternatives with respect to the NFRs specified in the initial version of the requirements. If we cannot prioritize the alternatives with respect to one NFR, we have to continue the refinement of the architecture and requirements according to the steps in Section 3.1. We may stop the refinement when we can perform the prioritization.

Since each NFR has different criteria for the prioritization, we first prioritize the alternatives with respect to each NFR. The experts for each NFR have to provide its criteria. For example, a criterion about the learnability can be defined based on the expected scenarios for system usage, and the characteristics of an architecture that partially provides the effects to such scenarios. The usability experts have to give such criterion based on their expertise. When this method is applied to the improvement of existing systems and the costs for the development is crucial, we can use the impact analysis [13] as a criterion for the costs. A prioritization with respect to a NFR normally differs from a prioritization with respect to another. For example, architecture X is preferred to architecture Y with respect to the usability. However, Y is preferred to X with respect to the performance.

If a NFR is more important than the others, we can simply use its criteria for prioritizing the architectural alternatives. If several NFRs are important, we have to combine the results of their prioritizations so that we can simultaneously take them into account. We can use AHP technique [16] for the combination of several different prioritizations.

4 Illustrative Example: Geolocation Service

In this section we illustrate the application of TPM-SA2 to a system to clarify how it works and how effective it is.

4.1 First-Level Requirements

We focus on a context-aware system that provides services to users according to their context, e.g., geolocation. Some examples of the services are recommending good restaurants that are close to users or finding vacant car parks. The first-level requirements for the system are as follows:

- Functional requirements (FRs)
  - F1: Users can search the services provided according to the location and time, and several services are recommended.
  - F2: Users can rank the recommended services according to their preferences.
  - F3: Users can rank the services according to the current trend.
  - F4: Users can select some of the services.
- Non-functional requirements (NFRs)
  - Q1: The entire FRs must be securely performed. (security)
  - Q2: The entire FRs must be quickly performed. (performance)
  - Q3: The development costs of entire FRs should be low. (cost)
  - Q4: The system will be in operation as soon as possible. (time of delivery)
  - Q5: Users can easily use the FRs. (usability)

Even for the first-level requirements above, there are several assets to be threatened as follows:

- A1: Recommended services.
- A2: Ranking based on personal preferences, time and location.
- A3: Ranking based on the trend.
- A4: Chosen services.

These assets can be regarded as architecture-independent assets (AIAs) in TPM-SA. However, privacy information such as the person's name, age, and gender are not always an asset based on the first-level requirements because the traceability between the person and his/her private information is not always required in these requirements. They are actually candidates for the architecture-specific assets (ASAs) because some architectural types involve them in the refined requirements.

4.2 Exploring Alternatives

By the threat analysis for the first-level requirements, the following threats can be identified.

- Tampering of the service choices
- Tampering of the service ranking information
- Tampering of the services that a user selected

On the other hand, there are two major choices at architecture level 0 for the first-level requirements.

Table 2. Architectural alternatives

level 0	1	2	3
mobile	anonymous (a1)		
	user identification	by user ID, password	rich client (a2)
			cloud (a3)
		by terminal id	rich client (a4)
			cloud (a5)
ubiquitous	setting new terminals and sensors	anonymous (a6)	
		user identification	by id cards, passwords (a7)
			by biometrics (a8)
	using existing terminals (such as vending machines)	anonymous (a9)	

1. Mobile devices and their network infrastructure Many people today have their own smartphones, which provide a powerful functionality for storing data, communicating with servers, identifying the current location and time, and so on. Most of the requirements can be mainly implemented on the smartphones using various sensors.
2. Ubiquitous infrastructure like in Oulu city[1] In contrast to the mobile devices infrastructures, public devices are pre-located all over the city within this infrastructure. The users thus do not have to bring their own devices, but can use the system based on the requirements above. In this case, most of the requirements are implemented on the servers.

The requirements may need to be modified, added, or even deleted according to the architectural choice. Architectural designs may also be affected by the change in requirements. For example, the first requirements do not require the identification of the users. However, some architecture such as smartphones might bring about new requirements for user identification because they are closely linked to their owners and their personal information. If the architectural alternatives containing user identification and authorization ((a2)-(a5)) are chosen, the software has to be able to protect the new assets: the user credentials and privacy data. The following threats arise according to the assets.

- The leakage of the users' private sensor data such as the time and geolocation
- The leakage of the selected services that might cause a violation of the users' privacy

Therefore, most cost is needed for implementing the protection of these assets in the alternatives because of the potential threats. More cost is needed for the authentication with user IDs and passwords ((a2), (a3)), while authentication based on a terminal id needs less cost ((a4), (a5)), but contains higher security risks.

[1] `http://www.ubioulu.fi/en/home`

There are other architectural choices: rich client type ((a2), (a4)) or server type (a3), (a5)). The former mainly needs implementation on clients systems, and the latter mainly on servers. Although the rich client type usually provides better usability than the server type, it brings about other assets from the client environments and other protection costs as well.

4.3 TMP-SA in an Alternative

Since each alternative is derived based on the TMP-SA, we will explain how to concretely derive one alternative. We focus on alternative (a3) in Table 2 because it is threatened more often than the others.

Based on the infrastructure for mobile devices, the location information becomes ASA because it could be tampered with while being sent. We don't have to worry about the time because the system will have its own clock. In addition, the communication paths also become ASA because all the information is sent and received via wireless communications. As a result, we find the following ASAs at level 0 in the mobile infrastructure listed in Table 2.

- A5: Location information
- A6: Communication paths

To protect these new assets, we need the following functional and non-functional security requirements as countermeasures.

- F5: Location will be identified.
- F6: Devices will communicate with the system.
- Q6: F5 will be accurately achieved.
- Q7: F6 will be achieved in integrity.

User identification is useful for F2 because the users' preferences can be registered in advance. It is also useful for F3 because the system can accurately identify the trend. Without the user identification, some malicious users may intentionally and repeatedly select some specific services to make them more popular than the facts. The following asset is thus added at level 1 (a3) in Table 2.

- A7: User identification

Of course, others must not reuse A7. Based on this asset, the following requirements are elicited.

- F7: The system will accept the user identification, and authorize the user.
- Q8: F7 is confidentially achieved.

In the same way, the following assets and requirements are identified and elicited at level 2 (a3).

- A7': Issuable user identification.
- F8: The system will issue the user identification.
- Q9: F8 will be identifiably achieved.

Table 3. Criteria weight

security	0.140
performance	0.047
usability	0.196
cost / time of delivery	0.616

Table 4. Estimation results using AHP

id	security	performance	usability	cost / time of delivery	total
(a1)	0.0617	0.0245	0.0075	0.2553	0.3491
(a2)	0.0122	0.0101	0.0400	0.0523	0.1146
(a3)	0.0457	0.0026	0.0115	0.0761	0.1359
(a4)	0.0048	0.0081	0.1076	0.0630	0.1835
(a5)	0.0159	0.0022	0.0294	0.1694	0.2169

Finally, we have the following at level 3 (a3).

- A8: Personal preferences registered in advance.
- F8: Personal preferences will be registered and updated.
- Q10: F8 will be achieved in integrity.

We have three assets, four functional requirements and five non-functional requirements in addition to the first-level requirements. In the other alternatives listed in Table 2, some of them do not have to be taken into account. For example, A7', F8, and Q9 are out of the scope in (a4) and (a5) because the terminal ID cannot be issued by the system.

4.4 Prioritization

With our prediction method, nine architectural alternatives listed in Table 2 can be assumed. We prioritized the NFRs based on an assumption of a certain development project in which the priority order is the cost, security, usability and performance. Then, we estimated the alternatives (a1)-(a9) using analytic hierarchy process (AHP) [16]. AHP is a method for decision making based on mathematics and psychology. With AHP, analysts determine weights for every decisions and its factors. The criteria weights computed are listed in Table 3.

We eliminated (a6)-(a9) because they require the social ubiquitous infrastructure, which the system cannot solve by itself. The evaluation results for the other five alternatives is described in Table 4. Unless the system needs the user identification, (a1) is the most appropriate decision. If it requires user identification, (a4) is the best choice.

5 Discussion

As shown in Section 4, one of the advantages of TPM-SA2 is that the developers can estimate and compare the impact of multiple architectural alternatives and decide on the best one in the very early levels of refinement. Another advantage over TPM-SA is that with TPM-SA2 developers consider not only the security but also other NFRs such as the performance, cost, and usability. As the "security" column of Table 4 indicates, if the developers evaluate only the security using TPM-SA, the second order of priority is (a3). But it may not meet other NFRs such as the cost. With TPM-SA2, the second order is (a5) using evaluation of multiple NFRs according to the stakeholders' preference.

Our TPM-SA with prediction has some drawbacks. One of them is that the prediction step in the first level is heavy and time-consuming since it requires the traversal of all the branches through all the refinement levels. However, the following information might reduce the deeper traversal from some nodes.

- Past estimation results
  This knowledge is particularly useful for the enhancement of existing software. If there is an architectural design that is the same as that that the design developers are assuming through the prediction process, and the NFR values have already been estimated, and the developers can use these values instead of proceeding into the deeper levels.
- NFR patterns
  Patterns that contain sets of NFR values related to certain architectural designs are also useful. If the assumed architectural alternative matches one of the NFR patterns, the developers can use the values of the pattern such as the past estimation results.

6 Conclusion

We proposed a new method called TPM-SA2 that elicits non-functional requirements and an architecture based on the Twin Peaks Model. In detail, we at first specify the concerns about not only the security requirements but also the other non-functional requirements with the architectural alternatives. We can, then, select the best architecture based on estimation of the impact on the non-functional requirements. TPM-SA2 enables us to explore the non-functional requirements step by step to make adequate architectural decisions from the multi-non-functional requirements' points of view. We applied our method to an application to evaluate it. The results of the evaluations illustrated that our method is especially effective for the iterative development.

Our future work includes developing a tool that supports our methods. Since TPM-SA2 uses domain knowledge concerning the non-functional requirements and expected architectural decisions, such issues can be automatically aligned with the currently elicited security requirements using the domain ontology. Mining architecture patterns which can help estimating security and costs and

pruning, is also the future work In addition, non-functional requirements elicitation process can be automatically managed according to the obstacles. The non-functional requirements sometimes conflict with other kinds of non-functional requirements. Therefore, we have to improve our TPM-SA2 to detect such conflicts to resolve them (semi-)automatically. We also need to apply TPM-SA2 to realistic projects to clarity its advantages and limitations.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley (2003)
2. Fernandez, E.B.: Security patterns and secure systems design. In: Bondavalli, A., Brasileiro, F., Rajsbaum, S. (eds.) LADC 2007. LNCS, vol. 4746, pp. 233–234. Springer, Heidelberg (2007)
3. Ferraz, F.S., Assad, R.E., de Lemos Meira, S.R.: Relating security requirements and design patterns: Reducing security requirements implementation impacts with design patterns. In: ICSEA, pp. 9–14 (2009)
4. Guo, Z., Zeckzer, D., Liggesmeyer, P., Mackel, O.: Identification of security-safety requirements for the outdoor robot ravon using safety analysis techniques. In: International Conference on Software Engineering Advances, pp. 508–513 (2010)
5. Heyman, T., Yskout, K., Scandariato, R., Schmidt, H., Yu, Y.: The security twin peaks. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 167–180. Springer, Heidelberg (2011)
6. Houmb, S.H., Islam, S., Knauss, E., Jürjens, J., Schneider, K.: Eliciting security requirements and tracing them to design: An integration of common criteria, heuristics, and umlsec. Requir. Eng. 15(1), 63–93 (2010)
7. Kazman, R., Klein, M.H., Barbacci, M., Longstaff, T.A., Lipson, H.F., Carrière, S.J.: The architecture tradeoff analysis method. In: ICECCS, pp. 68–78 (1998)
8. van Lamsweerde, A.: Elaborating security requirements by construction of intentional anti-models. In: ICSE, pp. 148–157 (2004)
9. 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)
10. Mead, N.R., Hough, E., Stehney, T.: Security quality requirements engineering (square) methodology. Tech. Rep. CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University (2005)
11. 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)
12. Nuseibeh, B.: Weaving together requirements and architectures. IEEE Computer 34(3), 115–117 (2001)
13. Okubo, T., Kaiya, H., Yoshioka, N.: Analyzing impacts on software enhancement caused by security design alternatives with patterns. IJSSE 3(1), 37–61 (2012)
14. Okubo, T., Kaiya, H., Yoshioka, N.: Mutual refinement of security requirements and architecture using twin peaks model. In: COMPSAC Workshops, pp. 367–372 (2012)
15. Okubo, T., Yoshioka, N., Kaiya, H.: Security driven requirements refinement and exploration of architecture with multiple nfr points of view. In: IEEE International Symposium on High Assurance on Software Engineering (HASE) (to be appeared, 2014)

16. Saaty, T.L.: The analytic hierarchy process: planning, priority setting, resource allocation, 2nd edn. RWS, Pittsburgh (1990)
17. Saeki, M., Kaiya, H.: Security requirements elicitation using method weaving and common criteria. In: MoDELS Workshops, pp. 185–196 (2008)
18. Tanabe, D., Uno, K., Akemine, K., Yoshikawa, T., Kaiya, H., Saeki, M.: Supporting requirements change management in goal oriented analysis. In: RE, pp. 3–12 (2008)
19. Weinreich, R., Buchgeher, G.: Integrating requirements and design decisions in architecture representation. In: Babar, M.A., Gorton, I. (eds.) ECSA 2010. LNCS, vol. 6285, pp. 86–101. Springer, Heidelberg (2010)