1. INTRODUCTION

As IT services play an important role in our life and society, security incidents affect not only these services but also our society. So, development of secure software services is essential and some development methods have been proposed such as Secure development lifecycle methods (CLASP) (Viega, 2005) and Microsoft SDL (Howard & Lipner, 2006). However, these methods are rarely used effectively (Geer, 2010) in software development. The reason for this is that although these lifecycle methods are for building software from scratch, most actual software development entails software enhancements in some form or another. Software enhancements are to add new features to existing software and/or to modify existing functions, and are done frequently in many services. However, it is difficult to ensure security with software enhancements using traditional methods, because these might have a major impact on existing software and security and it is hard to modify...
the existing software effectively to enable the security without comprehensive knowledge about security. We cannot assume that all engineers have the knowledge in practice. Therefore, current secure development lifecycle methods are problematic for accomplishing software enhancements.

It is important to estimate modification costs at the requirements stage of software enhancements for two main reasons. First, we need to consider changes in security requirements at this stage. We should avoid unnecessary countermeasures because security degrades other non-functional requirements such as development costs, performance, and usability. Additionally, we have to develop all important countermeasures. We should therefore identify major threats at the requirements stage to develop appropriate countermeasures.

Second, we need to analyze the impact of identifying two or more countermeasures against a threat on the existing software. Security development involves costs that must be limited. This is why we need to estimate costs to choose a suitable security solution at the requirements stage.

It is difficult to estimate what impact there will be on security without comprehensive knowledge about security, because it is hard to identify vulnerability of existing software to be modified and to grasp the effect on it without the knowledge. In addition, security concerns traverse the functionalities of existing software. There are two types of impact: horizontal impact on artifacts at the same stage and vertical impact on artifacts at a later stage. For example, suppose that we add credit card information to the user profiles of a Web shopping service to allow users to pay bills with their credit cards. As credit card information is an important asset, we need to consider a new threat, e.g., the risk of theft. It is hard to find where is vulnerability, such as vulnerability of a web protocol, to realize threats without knowledge. This threat impacts one or more functions in using user profiles, such as shopping carts, item recommendations, and edit profiles. In other words, if we have identified a new asset in existing software, we might consider adding new security countermeasures to some functions. This is an example of horizontal impact at the requirements stage. However, we need to modify the affected functions to implement security countermeasures, which have vertical impact on the code. Security codes are spread out over existing software and the impact depends on security architecture. Therefore, we need comprehensive knowledge about security to estimate the vertical impact.

This paper proposes a method of analyzing the impact of security on purposes of software enhancement. The method consists of two techniques: analysis of horizontal impact using an extended misuse case, which was described in our previous work (Okubo, Taguchi, & Yoshioka, 2009), and a combination of new security patterns and a traditional technique of traceability as a means of analyzing vertical impact on security. Security knowledge is encapsulated in security patterns. As the patterns bridge the gap between security requirements and design and a traceability tool can find the impact on the code (semi-)automatically, we can determine the impact on code when security requirements change without comprehensive knowledge about security.

Our research makes two major contributions. We first propose a new process of analyzing the impact of security based on our previous work. We then propose new security patterns including requirements level patterns and design level patterns to bridge the gap between requirements and design. Although these basic ideas have been proposed in Okubo, Kaiya, and Yoshioka (2011), the previous work does not include the detail of patterns and the evaluation was preliminary. This paper illustrates complete pattern description with examples and deep evaluation and the discussion about our method.

This paper is organized as follows. The next section describes related work and security issues with software enhancements. We then describe our new method of integrating security patterns and impact analysis. The following sec-
A Formal Language for XML Authorisations Based on Answer Set Programming and Temporal Interval Logic Constraints
[www.igi-global.com/article/formal-language-xml-authorisations-based/52594?camid=4v1a](www.igi-global.com/article/formal-language-xml-authorisations-based/52594?camid=4v1a)

Agile Software Methods: State-of-the-Art
[www.igi-global.com/chapter/agile-software-methods/5066?camid=4v1a](www.igi-global.com/chapter/agile-software-methods/5066?camid=4v1a)