Monitoring Application for the Secure Software Development based on Risk Assessment Model

Rita Rijayanti¹, Mintae Hwang²

¹ Student, Dept. Information and Communication Engineering, Changwon National University, Korea, rita.rijayanti@gs.cwnu.ac.kr
² Professor, Dept. Information and Communication Engineering, Changwon National University, Korea, mthwang@cwnu.ac.kr

Corresponding author: Mintae Hwang

Abstract: The development of software in a wide range of remote locations by large developers generates problems, in which a number of bugs or errors appear after deployment. Therefore, the solution is developing a monitoring application based on secure software development and risk assessment concept on Software Development Life Cycle (SDLC). The developed concept is based on mapping result of touchpoints for secure software, and NIST SP 800-30 for risk management in each stage of SDLC. The measures particularly support developer teams in remote distributed environments. The focus is on the risk assessment performed during the stage of initialization to implementation. In addition, according to the mapping results as well as business process analysis, there are five main functions related to this study, including creating projects, designing process, developing process, testing process, and deployment. Additionally, a web-based monitoring application is implemented to secure the software development process based on security control procedures at each stage, and developed using PHP programming languages and MySQL for a database. Moreover, the application is triggered by five parameters: software type development, tools, database structures, module names, and error-related problems. From these parameters, the risks could be discovered and subsequently categorized into four types of risk, such as low, medium, high, and critical, based on the impact of each risk. The results signify that the number of supports significantly decreased by 80%. Correspondingly, this application is expected to support secure software development as well as provide efficient treatment for possible errors and security risks.

Keywords: Secure Software, Risk Assessment, Touchpoint, NIST SP 800-30, Web-based Monitoring Application

1. Introduction

In an industrial era of 4.0, Information Technology (IT) emerges as a large demand in society upon the success on resolving problems in significant areas such as industry, health, banking, online shopping, education, as well as employment[1-3]. Thus, IT security-related issues, from development to implementation, have been parts of recent attention[2]. Security issues are consistently associated with risk, including risks related to planning, improperly executed performance, changes in requirements, developing processes, and risks associated with tools applied during the development process. Therefore, software development requires immediate security treatment to protect valuable assets[3].
In response to the need for applications to support daily activities, software development companies attempted to resolve the existing problems; where recently software development organizations use problem-tracking tools to monitor software process throughout software development life cycle. For instance, Jira is one of the most widely used monitoring applications to monitor development process. However, similar to other applications, Jira focuses merely on monitoring stages or status of the project progress instead of securing software development process from potential risk. In fact, this monitoring application is generally limited and becomes severely challenging when the case occurs across different projects, locations, and timescales [4].

The most severe issue is when large development teams are located in distant areas. This has been a challenge in securing software development in remote distributed environments, which subsequently creates problems, for instance, functionality, bugs or errors, and data loss due to low security controls that generate several supports to redeploy the problems. However, fixing or modifying the application is difficult due to these problems, which lessens the developers’ credibility from the users. At this point, a solution is required. Hence, this study intends to build a monitoring application by adding functionality to secure development process related to potentially emerging risks from the initial stage until the stage is completed. In other words, until the stages of the development process are done or deployed. Furthermore, the whole stages of software development are related to secure software development process which are based on the mapping result of software development life cycle, secure software development, as well as risk management methods conducted in previous studies [5].

The paper is organized as follows. Section II describes related research. Section III illustrates the mapping of secure control design. Section IV describes the design of the monitoring application. Section V describes a sequential process of the monitoring application. Section VI displays the implementation of the monitoring application. Section VII presents the conclusion.

### 2. Related Works

#### 2.1 Risk Analysis Method

This study limits the risk analysis method to ISO 27001, Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE), and National Institute of Standards and Technology (NIST) SP 800-30 as the samples. According to ISO 27001, the focus of risk analysis is on asset categorization, while the output is to set operational standards [6]. OCTAVE is a risk-based information system security assessment and planning method (focusing on information assets) with output as a security strategy (Context: Implementation, critical) [7]. NIST 800-30 is intended to carry out a risk assessment to establish security controls and to improve the control of information technology organizations related to the Software Development Life Cycle (SDLC) with an output in the form of Security Controls (Context: Target work procedures). From this description, NIST 800-30 is the appropriate method to create a security control in terms of working procedures to be applied to the monitoring application.

#### 2.2 Monitoring Application

Monitoring applications are commonly developed to monitor or detect development problems [5]. One of the approaches in monitoring applications for software development in terms of handling projects is by automatically extracting process definition in the form of constraint (i.e., Jira tracing tools) which generate mining reports for different types of issues across projects and at different times. However, the downside of this approach is the inability of handling huge resources and secure code [4]. As a response, this study is intended to resolve this limitation.
2.3 Secure Software Development

Integrating risk management in SDLC and applying risk management to the application of SDLC are unceasing issues[5]. Predominantly, the integration between risk management and software models is conducted in a rapid application concept instead of a full development process, for example, SDLC[8-10]. On the other hand, a study focuses on measuring software security performance before software development to continuously improve software security in developing an application by conducting security evaluations in each process[8]. Moreover, different studies assume that software security needs to focus on security for confidentiality and integrity [9]. On top of that, developers need to create a rule-based access control policy model to build secure code[11]. Further, the close resemblance to the developed monitoring application has been discussed in [5], in which the integrating risk assessment and threat modelling within the SDLC for a software defect are identified and detected during the development life cycle.

Currently existing studies focus solely on developing secured software using a single method. There has been no study emphasizing on combining SDLC, touch points for software security, and NIST SP 800-30 risk management to create more adequate security controls for secure software development based on possible risk at each stage of software development. Therefore, the study is designed to establish secure software development by implementing the combined method.

3. Procedure Mapping for the Secure Software Development

[Fig. 1] illustrates relationship mapping of software development, touchpoints for secure software, and the NIST SP 800-30 risk management, which has security control outputs in the context of a work procedure form. Combining these three methods provides better results in developing secure software controls as there is a stage that exists in touchpoints for secure software that does not exist in NIST SP 800-30 risk management and vice versa, which balances each other[5].

Both touchpoints and NIST SP-800-30 perform to secure software development based on risk, with varied risk focus and handling. Touchpoints are one of the three pillars as well as the best practice of software security that mixes destructive and constructive activities. Destructive activities include identifying possible attacks and exploits on software, while constructive activities are related to design and functionality, where the two activities are based on possible abuse cases, to analyze risks in software development accordingly. On the other hand, NIST SP 800-30 is a risk management standard focusing on metrics, testing, and validation; thus, to promote, measure, and validate the security of information system related to business process. This method is recognized as the best practice for improving security control in businesses process related to SDLC.

Implementing IT security controls throughout the software process begins by identifying common
problems that arise during the software development stage, development environment, and vulnerabilities based on business procedures, environment, and threats that could negatively impact the future.

Identifying initial requirements help determine the possibility of each risk in each section, where each risk is assessed based on the value and impact of the asset. The determination of risk outcomes is classified into low, medium, high, and critical. Furthermore, the secure control recommendations are evaluated from the risk results before being mapped into the security control guidelines for the entire SDLC process.

4. Design of the Monitoring Application

4.1 Risk Analysis and Assessment Process

Based on the secure map in [Fig. 1], risk assessment is started from an identification, where each phase of SDLC is vulnerable to different risk factors; thus, identifying and understanding the risk is conducted in the preliminary stage for effective risk management. Accordingly, the design process begins by identifying business processes, assets, problems, threats, and vulnerabilities that commonly occur based on business procedures and environmental conditions along with subsequently determine impact and risk level which is measured based on value of the asset and the impact.

4.1.1 Risk Identification

[Table 1] Identifying the Impacts from Threats and Vulnerabilities

<table>
<thead>
<tr>
<th>Threat</th>
<th>Vulnerability</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large supports are created after the deployment stage.</td>
<td>There is no requirement for creating security functions in software development, as well as the lack of historical records of each module.</td>
<td>Multiple bugs to rebuild.</td>
</tr>
<tr>
<td>Processing time deviates from the scheduled working time.</td>
<td>Lack of knowledge from human resources (skills, tools, modules, and linkages between data) or uneven expertise.</td>
<td>Requestor / User distrust towards developer.</td>
</tr>
<tr>
<td>Failure to implement software design (functional / UI) per module.</td>
<td>Lack of knowledge from human resources (skills, tools, modules, and linkages between data) or uneven expertise; there is no standard for creating document for software development design.</td>
<td>Requestor / User distrust towards developer.</td>
</tr>
<tr>
<td>Penetration access to the terminal.</td>
<td>Unstable network connection.</td>
<td>Access failure results in data loss, hence requires additional time in process of repairmen.</td>
</tr>
<tr>
<td>Hardware damage.</td>
<td>There is no regular schedule for hardware maintenance (the maintenance is performed when problem occurs).</td>
<td>The development and/or testing process stops working (processing time is deviated from the scheduled time).</td>
</tr>
<tr>
<td>Inaccuracy during testing process (requirements and test cases do not match).</td>
<td>Test cases are incomplete, and lack of human knowledge (skills, tools, modules, and linkages between data) or uneven expertise.</td>
<td>Multiple bugs/errors appear when the software is delivered to the Requestor/User (requiring more frequent corrective support)</td>
</tr>
<tr>
<td>Loss of documents related to the project or corrective support activities.</td>
<td>Misconduct on access authority.</td>
<td>Examining problems (presumably occurred beforehand) is challenging, particularly in repairmen when error occurs.</td>
</tr>
<tr>
<td>Debugging or Human error.</td>
<td>Lack of human knowledge (skills, tools, modules, and linkages between data) or uneven expertise and unstable network connections.</td>
<td>Triggering bugs to appear when the system is running.</td>
</tr>
<tr>
<td>Sabotage during the development process (hardcode) and problem in deploying processing time.</td>
<td>Misconduct on access authority and inconsistent system requirements (frequently changes).</td>
<td>Generating data loss (source code, stored procedures, client data, and documents), termination on operational activity system, leading to Requestor/User’s skepticism.</td>
</tr>
<tr>
<td>Random code modifications (Source code &amp; store procedure)</td>
<td>Misconduct on access authority.</td>
<td>Loss of source code and store procedures prompt to operational process termination.</td>
</tr>
</tbody>
</table>
In the initialization stage, the needs for overall software development are analyzed. This can be implemented in the requirement and use case stage of secure software development touchpoints to identify abuse cases that potentially arise at each stage related to the threat associated with the business processes. Five functions, however, attribute to this, such as creating projects, designing process, developing process, testing process, and deployment. Each process is assessed and identified to detect potential risks. The risk is discovered from the identification stage before comparing similar conditions between the software development environment and real conditions of the system being developed, where actors related to the actual software development process include the Project Manager (PM), Team Leader (TL), Analyst, Developer, Database Administrator (DBA), Quality Assurance (QA), and Requestor/User.

Based on NIST SP-800-30, the potential risk is identified following the impact of each threat and vulnerability derived from identifiers functions. As an example, in the design process, the possible threat is "Penetration access to the terminal"; the vulnerability is "Unstable network connection" which the impact is Access failure leads to data loss; thus, the potential risk is "functional software errors or Poor software performance (slow)". The details are illustrated in [Table 1]. Afterwards, the overall processes are adapted to the risk assessment concept derived from NIST SP-800-30 by including the existing security control and future planning to create security design. For identifier risk as illustrated in [Table 2].

4.1.2 Risk Assessment and Classification

The next step is to divide the existing risks and subsequently perform the risk assessment process. The identified risk and values for conducting a risk assessment are described in [Table 2]. The results are utilized to classify the existing risks based on the type of risk level and are adjusted to risk management NIST SP 800-30. The risk classification is illustrated in [Table 3]. The classified risks are created in the form of a risk matrix, as illustrated in [Table 4].

<table>
<thead>
<tr>
<th>No</th>
<th>Risk Identification</th>
<th>Threat values</th>
<th>Vulnerability Values</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software functional error</td>
<td>9</td>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>Incorrect modification (source code update error)</td>
<td>9</td>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>3</td>
<td>Loss of source code during the development process</td>
<td>8</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>Loss of data related to Requestor/User (failure in backup and restore)</td>
<td>9</td>
<td>9</td>
<td>81</td>
</tr>
<tr>
<td>5</td>
<td>Data input error</td>
<td>8</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>Human error</td>
<td>7</td>
<td>7</td>
<td>16.3</td>
</tr>
<tr>
<td>7</td>
<td>Poor software performance (slow)</td>
<td>8</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>Debugging Problem</td>
<td>9</td>
<td>9</td>
<td>40.5</td>
</tr>
<tr>
<td>9</td>
<td>Required documents are incomplete</td>
<td>7</td>
<td>5</td>
<td>35</td>
</tr>
</tbody>
</table>

The Risk level is different from risk value. Risk value is the result of multiplying the threat and vulnerability. On the other hand, the risk level is the classification of the risk type (low, medium, high, and critical). The risk type is derived from multiplying risk level with likelihood. Furthermore, the risk value is equated as follows:

$$RV = T \times V$$  \hspace{1cm} (1)

Note: RV: Risk Value; T: Threat; V: vulnerability

The level of risk is classified into four types, namely:

- Low when relates to the user interface, such as the size of the figure, table or button size, and position.
- Medium when the risk impacts moderately on certain data other than sensitive data, thus requiring immediate action.
- High when the risk impacts sensitive data, thus requiring control planning and immediate treatment.
- Critical, when the risk impacts sensitive data, causes damages, and affects the performance of other functions; thus requiring special treatment.

### [Table 3] Risk Classification

<table>
<thead>
<tr>
<th>No</th>
<th>Risk Identification</th>
<th>Risk Value</th>
<th>Likelihood</th>
<th>Risk Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software functional error</td>
<td>81</td>
<td>130</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>Incorrect modification (source code update error)</td>
<td>81</td>
<td>130</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>Loss of source code during the development process</td>
<td>64</td>
<td>20</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>Loss of data related to Requestor/User (failure in backup and restore)</td>
<td>81</td>
<td>20</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>Data input error</td>
<td>64</td>
<td>25</td>
<td>M</td>
</tr>
<tr>
<td>6</td>
<td>Human error</td>
<td>28</td>
<td>20</td>
<td>M</td>
</tr>
<tr>
<td>7</td>
<td>Poor software performance (slow)</td>
<td>64</td>
<td>30</td>
<td>M</td>
</tr>
<tr>
<td>8</td>
<td>Debugging Problem</td>
<td>40.5</td>
<td>120</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>Required documents are incomplete</td>
<td>35</td>
<td>30</td>
<td>M</td>
</tr>
</tbody>
</table>

### [Table 4] Risk Matrix

<table>
<thead>
<tr>
<th>Impact</th>
<th>Exceedingly Frequent</th>
<th>Frequent</th>
<th>Rarely</th>
<th>Seldom</th>
<th>Exceedingly Rare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td>3, 5, 6, 7</td>
<td>8, 9</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.2 Design of Security Controls

### [Table 5] Risk Identification and Recommendation

<table>
<thead>
<tr>
<th>No</th>
<th>Risk Identification</th>
<th>Security Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software functional error</td>
<td>Focus on protecting the application before and after deployment</td>
</tr>
<tr>
<td>2</td>
<td>Incorrect modification (source code update error)</td>
<td>Focus on designing process</td>
</tr>
<tr>
<td>3</td>
<td>Loss of source code during the development process</td>
<td>Focus on developing code and deploying</td>
</tr>
<tr>
<td>4</td>
<td>Loss of data related to Requestor/User (failure in backup and restore)</td>
<td>Focus on developing code and deploying</td>
</tr>
<tr>
<td>5</td>
<td>Data input error</td>
<td>Focus on developing code and testing process</td>
</tr>
<tr>
<td>6</td>
<td>Human error</td>
<td>Possibly occurs in each step of the development process</td>
</tr>
<tr>
<td>7</td>
<td>Poor software performance (slow)</td>
<td>Focus on designing and developing code</td>
</tr>
<tr>
<td>8</td>
<td>Debugging Problem</td>
<td>Focus on developing the code and deploying</td>
</tr>
<tr>
<td>9</td>
<td>Required documents are incomplete</td>
<td>Possibly occurs in each step of the development process</td>
</tr>
</tbody>
</table>

The next step is to determine recommendations for security controls. The result of this security recommendation is applied in designing monitoring applications in the form of a working procedure in the software development process. Further, the possible risk from identification based on the analysis of business processes and security recommendations are displayed in [Table 5]; on the other hand, the low risk level is excluded from the risk identification list, assuming that a low condition equals to secure application. Thus, no security recommendation is required. However, this method could have different
implementation results for each case depending on the existing procedures in each organization, the potential risks, functionality, and the users involved.

The solution derived from the security recommendations applied in monitoring applications based on risk identification is explained as follows.

- **Software functional error**
  
  This issue is resolved by creating rules when monitoring the process at each stage of software development. This is conducted by PM before forwarding the message regarding status and condition from the monitoring process to the Requestor/User (User Accepted test). These rules are intended to check the overall conditions as a response to each request, as well as to check the risk possibility from the preparation process to the end of the development process. In addition, in creating a design, the analyst refers to the requestor's requirements; the PM subsequently confirms before approving the design, and later to be sent by the analyst.

- **Incorrect modification (source code update error)**
  
  Two solutions were implemented in this study. Creating a rule in the software development process is the initial solution. Here, upon acquiring TL approval, developers modify codes and data relationships between modules before updating the data and deploying/checkout in beta or local version before implementing the source code on the live server. These are parts of review stage and control changes. The second solution is providing access to manage updates on each Source Code (verification) by developers, DBA, TL, and PM.

- **Loss of source code during development process**
  
  The solution is creating a rule in which each activity or change related to user’s data should be confirmed to the Requestor/User by PM or TL as the person in charge. In this case, each developer should have backup data separated from the beta or live server, and is responsible to upload the updated code to the monitoring application. In addition, to store data history, scheduling for restoring data is required; thenceforth, the development team is notified before restoring the data.

- **Loss of data related to Requestor/User (failure in backup and restore)**
  
  The solution is to provide a rule for the Developer or DBA to check the database environment in a certain state before updating, in the sense that no team is working on the server or database when backing up or restoring the data. In addition, the development team is notified before recovery process.

- **Data input error**
  
  The solution is to create a rule in which the developer team confirms the overall data input correctly by checking the requirements approved by the Requestor/User regarding the format and the relationship between the data. All related documents should be input into the monitoring application to facilitate secure development and testing process.

- **Human Error**
  
  The solution is training by the company to upgrade the skills in the software development process to attain larger experience and qualifying skills in performing the assigned duties and responsibilities (technical and functional), which is intended to minimize human errors at work.

- **Poor Software Performance (slow)**
  
  The solution is to create code standardization in software development to ease the review. In this case, monitoring the network connection and all user devices are conducted periodically (upgrade based on the needs).

- **Debugging Problem**
  
  The solution is to create a rule that allows double-checking before debugging process; in practice, verification by TL as the person in charge was required before the execution. In addition, the development team is notified before debugging process.

- **Documents Required are Incomplete**
The solution is to create a rule by creating notifications in case documents, such as Project Charter, Test Case Document, Release Note Document, and Checklist that have not been completed. The notification alert, however, appears when the team transitions to the next stage. Additionally, a monthly/weekly review for each document created by PM, TL, and QA is mandatory.

5. Sequential Process of Monitoring Application

The sequential process of monitoring application is described as a flowchart, illustrated in [Fig. 2], where an actor is assigned for each function to perform each rule, for instance, creating project by PM; analyzing and designing by TL and analyst as based on requirements; developing software process by Developer and DBA; recovery of database and server by DBA; testing by QA; and deployment by PM, TL, Analyst, Developers, and DBAs. The deployment, however, is proceeded upon approval by PM.

![Flowchart of Monitoring Application](image)

The monitoring stage begins with requests submitted by requestors; therefrom, PM creates a new project. Afterwards, the process of identifying is performed based on requirements. Thence, the decision is made whether the project is to be approved, suspended, or rejected. In case the new project is approved, PM and QA start documenting each stage, while the analyst creates the design. The design should be verified by the requestor and PM. Upon verification, the analyst submits the design results to the developer to proceed in the development process. Otherwise, redesigning is mandatory.

Developers and DBA start working accordingly upon receiving notification that the design has been...
completed. By referring to the emerging risks, both developers and DBA should adhere to the instructions and information displayed in monitoring applications regarding “what and how” the code should be implemented in the software development process. Each result/update from the developers and DBA is checked/validated by QA to correspond the requirements, and to eliminate bugs. When the process successfully passes QA verification, developers and DBAs subsequently change the status and send notifications to TL, Analyst, and PM before checking out either to the beta or live server. Furthermore, the project could be closed in case all parties have verified all stages, particularly the Requester/User, TL, and PM.

6. Implementation of Monitoring Application

6.1 Environment

The monitoring application is created as a web-based application using PHP programming language, while the database uses MySQL, to store comprehensive data. The server is divided into local, beta, and live for the security side. The application is triggered through a request, and accordingly, the PM creates a new application or improves the application that has already been developed. The request contains information regarding the application type, data, module name, related module or application, problems, and the environment. This application displays all requests from the requestor and is sorted based on the highest urgency or risk level to be treated first. This information has implemented a reference to create a new project from the requestor based on potential risks during the development. This application is utilized to monitor all software development processes, from the initial stages of request from the requestor to deployment. However, each stage of the development is to be monitored for security.

6.2 Implementation

This application displays activities, such as “to be performed”, “in progress”, and “resolved” by the developer team, thenceforth, the teams and related parties such as Requestor/User are able to monitor the application development status. Additionally, general information, such as status, components, development category, application type, requestor, and priority are illustrated in [Fig. 3], a priority of which depends on the environment, the risk (low-critical), personnel to be assigned, as well as the estimated processing time as a subtask.

The “Priority” in [Fig. 3] is the level of risk automatically generated by the monitoring application using the information derived from data input by PM, such as requirement identification, environment, and the team to be assigned to work on the project. Furthermore, the security controls are set automatically based on this level of risk.

In this monitoring application, the project could be started upon approval by PM, where each section is equipped with an approval function as a control to prevent errors that negatively impact the application results. This is one of the procedures to control and monitor the development process.

[Fig. 4] describes the development requirements based on a request from the requestor/user, as well as information on whether the request contains sensitive code or not (other related functions and applications). In case the information contains sensitive code, the valid incentive section automatically displays 'YES' or 'TBD (to be announced)'; this section, afterwards, displays a notification alert to validate whether the project is approved or rejected, considering that the notification is associated to varying sensitive functions and data.

In addition, in case a sensitive code appears and is approved, a critical status appears automatically. Accordingly, TL and Analyst ensure the functionality and code correspond to the requirements properly. Thus, the design submitted to the Development and DBA contains security instructions. Furthermore,
Monitoring Application for the Secure Software Development based on Risk Assessment Model

based on the risk presented here, Developers and DBAs follow the instructions and information displayed in the monitoring application regarding “what and how” the code is implemented in the software development process. On top of that, sensitive functions are required to be unlocked by TL or Analyst before checking out to local, beta, or live servers.

[Fig. 3] Sample Monitoring Web Page with General Project Information

[Fig. 4] Sample Monitoring Web Page for Project Logging

[Fig. 5] illustrates comments from the overall team activities as “important notes or logs” from each personnel to determine the stage where the activities have been performed as one of the monitoring
activities. This section confirms the status, particularly for QA and requestors/users required to ensure that the development corresponds to the requirements. In addition, developers include the inclusive relationships and code updated in this section as historical data, which is to utilized in case redevelopment is necessary for the future.

![Comments](image)

*Comment by Sri Mulyani [22/Feb/12 11:11 AM]*

Sir,

You have submitted a project request

http://ara.icdcapmo.co.id/browses/COMMIT-12362

We are pleased to inform that the issue has been completed, and tested on QA side today.

The QA report is attached within this email.

Please review the report and let us know if you have feedbacks, concern, suggestion or additional requirement.

Regards,

Sri

*Comment by Sri Mulyani [22/Feb/12 11:11 AM]*

Passed from QA

*Comment by Sri Mulyani [21/Feb/12 12:01 PM]*

Test starting

*Comment by Ayana Budi Kusumianti [21/Feb/12 11:27 AM]*

Test ready for test

*Comment by Ayana Budi Kusumianti [21/Feb/12 11:20 AM]*

done

*Comment by Rina Ristyanti [20/Feb/12 03:49 PM]*

Re-assign Sri Mulyani as QA.

Thanks,

Rusi

*Comment by Sri Mulyani [06/Dec/11 09:19 AM]*

Begin testing

*Comment by Rani Noviandi [06/Dec/11 09:17 PM]*

AA Sir,

Please check these reports.


Thanks,

Rusi

*Comment by Rina Ristyanti [06/Dec/11 10:07 AM]*

Re-assign Sri Mulyani as QA

*Comment by Riesman R Marpaung [02/Dec/11 09:28 PM]*

acknowledge

![Fig. 5] Sample Monitoring Web Page for Requestor Note and Transition Process

6.3 Result

The monitoring application in this study is different from similar monitoring applications in a way that the design mainly focuses on monitoring the secure software development process that is implemented in each step, as well as the risks in accordance to the concept of secure software touchpoints and NIST SP 800-30. By focusing on risk and monitoring, as well as incremental validation, the level of security in the software development process increased as proven by a decrease number of bugs or errors after deployment; correspondingly create a suitable, secure, and upstanding results. Moreover, the number of supports that require redevelopment significantly decreased by 80% (10-15 to 3-5 per day/case, even 0 case).

7. Conclusions

The objective of this study is to develop a web-based application to monitor software development process from the initial stage until the status is closed or completed. The stages in the SDLC process, however, are mapped with touchpoints to secure software development and NIST SP 300-80 for risk
management. The mapping results are presented as a procedure, where the risks are grouped based on the asset's value as well as the impact, which is categorized as low, medium, high, and critical.

In this study, the risk level is automatically set based on the information derived from the requirements. Afterwards, the results are employed to determine the implementation of security controls during the software development process. This research uses Touchpoint and NIST SP-800 30, the focus is on business process. In addition, there are five functions in monitoring the application: creating a project, adding a sub-task, development, documentation, QA processes, and deployment, where each process has features for confirmation, verification, and logs. However, this method potentially creates different implementation results of security control for each case, depending on the procedures and environment in each organization, possible risks, functionality, and users.

By implementing this monitoring application during the development process, the number of bugs and errors that requires redevelopment significantly decreased by 80% (10-15 to 3-5 cases per day, even 0 case). However, the drawback of this application is, unlike other monitoring applications which are accessible through multidevice, the monitoring application developed in this study which is accessible merely through Personal Computer due to limitation in user interface and functionality. Moreover, the monitoring application is currently not possible to be implemented in mobile application as the mobile application does not support some features to implement checkout code and store procedure to the server.

The challenge is to fit this method on a wide range of systems and ensures that the risk is defined properly from each stage to prevent potential problems. Furthermore, the researchers consider adding performance evaluation and feature to validate the application versions before checkout, thus, improving the performance during the deployment.

References

DOI: http://dx.doi.org/10.1109/ICIMTech53080.2021.9535057

DOI: http://dx.doi.org/10.1109/ACCESS.2021.3052311

DOI: http://dx.doi.org/10.1109/ICISCE50968.2020.00159

DOI: http://dx.doi.org/10.1109/ICSSP.2019.00021

DOI: http://dx.doi.org/10.1007/978-981-15-1465-4_54

DOI: http://dx.doi.org/10.1109/CITSM.2018.8674294

DOI: http://dx.doi.org/10.1109/ITC-CSCC.2019.8793421

DOI: http://dx.doi.org/10.1109/ACCESS.2020.3007355

DOI: http://dx.doi.org/10.1109/NCG.2018.8593135

DOI: http://dx.doi.org/10.1109/INVENTIVE.2016.7823275

DOI: http://dx.doi.org/10.1109/MS.2018.290101728