CHAPTER 7
Security Metrics for Software Development Phases

7.1 Introduction

Most of the security vulnerabilities are result of flaws that are introduced inadvertently during design and development of a software system. To decrease software vulnerabilities, the overall defect content must be reduced. It is important to measure the vulnerabilities in order to mitigate the security defects and produce more secured product. The primary goal of metrics is to quantify data so as to facilitate insight and provide degree of trustworthiness [JAQ07, W+09]. Metrics help the project management team to effectively manage the software development process as well as judge the product. It expedites decision support, especially assessment and prediction of security aspects of the software [SAV08].

Security metrics are quantitative indicators of the security aspect of the software that assesses the quality related imperfections of the software [CCZ08]. The security of the software can be assessed through product and process level metrics. Most of the security metrics analyzes security at the system level [AMR07, MW05, CIS10a, NIL10]. A number of security metrics evaluate security consideration during the various stages of software development [SEH08, NP07, ALL09]. Several other security metrics are aimed towards specific stage of development process of software such as design, coding and testing [AFC10, W+09, LINK66]. It has been observed that the metrics describe either the security aspect of the software or assesses security risks during Software Development Process (SDP) but the realization is not at a glance.
Moreover, the metrics discussed for SDP do not provide enough details regarding implementation. Also, the metrics are not implemented on the basis of security concerns during stages of software development process.

To understand the significance of security metrics, we reviewed the available product and process metrics for various stages of SDP. Further, we have proposed security metrics and effectiveness factor for software development stages that may help in controlling the security flaws. The process metrics based on different software development stages and the product metrics are illustrated in Section 7.2. In Section 7.3, we propose proactive security metrics required during the software development process that focuses on the security issues of its various stages. The effectiveness factor of security metrics on various development stages are presented in Section 7.4. Three case studies illustrating the security metrics and effectiveness factors are presented in Section 7.5. The results and conclusion are presented in Section 7.6 while we summarize the coverage of the chapter in Section 7.7.

7.2 Review of Existing Security Metrics

Metrics serve as a basis for project planning as well as states the degree of safety to avoid danger. To understand the significance of security metrics, we discuss existing security metrics for SDP and the product metrics.

7.2.1 Security Metrics in SDP

Metrics are beneficial during planning, organizing, controlling, and improving software development process. Security metrics also help to communicate and improve performance, measure the effectiveness of security considerations and help in diagnosing problems. A variety of security metrics have been discussed that portray the security issues of software development stages.
Requirements Gathering and Analysis

The number of security requirements gathered can be assessed by metrics such as Total number of security requirement (Nsr), Ratio of security requirements (Rsr), Number of omitted security requirements (Nosr) and Ratio of the number of omitted security requirements (Rosr) [SEH08]. Measures such as Percent of relevant software security principles reflected in requirements specifications, Percent of security requirements that have been subject to analysis, and Percentage of security requirements covered by attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis also determine security considerations in requirements gathering phase [ALL09].

Software Design

The design phase metrics assess the security issues pertaining to design artifacts. The security metrics to be considered during design stage are Number of design decisions related to security (Ndd), Ratio of design decisions (Rdd), Number of security algorithms (Nsa), Number of design flaws related to security (Nsfd), and Ratio of design flaws related to security (Rfd) [SEH08]. Architecture and design metrics include Percentage of architectural/design components subject to attack surface analysis and measurement, Percentage of architectural/design components subject to architectural risk analysis, and Percentage of high-value security controls covered by security design patterns [ALL09].

A set of metrics have been derived from viewpoint of information flow. It is based on object-oriented design artifacts viz. composition, coupling, extensibility, inheritance and design size of the given object-oriented, multi-class program. The metrics are divided for individual classes into three kinds of accessibility viz. instance attributes, class attributes, and methods. These are defined as Classified Instance Data Accessibility (CIDA), Composed-Part Critical Classes (CPCC), Critical Class Coupling (CCC) and Classified Methods Extensibility (CME). Metrics defined for Inheritance are Critical Superclasses Proportion (CSP), Critical Superclasses Inheritance (CSI), Classified Methods
Inheritance (CMI), and Classified Attributes Inheritance (CAI). Metric defined for design is Critical Design Proportion (CDP) [AFC10].

**Coding and Implementation**

The code level metrics judge the data structures and the implementation level details. A number of coding/implementation metrics are defined in the literature. Some of the metrics include Number of implementation errors found in the system (Nerr), Number of implementation errors related to security (Nserr), Ratio of implementation errors that have impact on security (Rserr), Number of exceptions that have been implemented to handle failures related to security (Nex), Number of omitted exceptions for handling execution failures related to security (Noex), and Ratio of the number of omitted exceptions (Roex) [SEH08]. Security measures for coding also involve Percentage of software components subject to static and dynamic code analysis against known vulnerabilities and weaknesses, Percentage of defects discovered during coding that was injected in architecture and design in requirements specification, and Percentage of software components subject to code integrity and handling procedures, such as chain of custody verification, anti-tampering, and code signing [ALL09].

In an attempt to quantify security at source code level, the metrics developed are Stall Ratio, Coupling Corruption Propagation (CCP), and Critical Element Ratio (CER). Stall ratio measures program’s progress as hindered by vivacious activities. Such activities do not contribute to the overall progress of the program such as a=a+0. It is calculated as the ratio of Lines of non-progressive statements in a loop to Total lines in the loop. Good stall statements include statements to write error messages, writing logs etc. Code with high stall ratio is more prone to attack. Coupling indicates the dependency of methods on each other in some way. CCP measures the total number of methods that could be affected by erroneous originating method. It has been provided as Number of child methods invoked with the parameter(s) based on the parameter(s) of the original invocation. The code may contain certain objects that when instantiated during run time may destabilize the process. It implies that more the critical
elements, higher are the risk. CER is calculated as ratio of Critical Data Elements in an Object to Total Number of Elements in the Object. The critical elements can be corrupted by the malicious user input. If the critical data objects change, the whole process may be subject to security risk. Thus, the code with higher CER should be tested more carefully [CCZ08].

Testing

Security testing metrics focus on identifying the security performance of the software being developed. Testing phase relates to metrics such as Ratio of security test cases (Rtc) and, Ratio of security test cases that fail (Rtcp) [SEH08]. CERT annual report mentions measures for testing security as Percentage of defects discovered during testing that was injected in coding, architecture and design, and requirements specification; Percentage of software components with demonstrated satisfaction of security requirements as represented by a range of testing approaches; and Percentage of software components that demonstrated required levels of attack resistance and resilience when subject to attack patterns, misuse/abuse cases, and other specified means of threat modeling and analysis [ALL09]. SANS reading room provides testing metrics as Security Testing Coverage [NEL10]. The other metrics include Number of Misuse Cases Considered while testing, and the Number and Types of Vulnerabilities Tested.

Maintenance

When the project is completed and is deployed, the software enters maintenance phase. The maintenance phase metric indicates the quality of the efforts during previous phases. These metrics cannot alter the product but reflects the quality of maintenance phase and helps to increase the customer satisfaction [KAN02]. The observed maintenance phase metrics include Ratio of software changes due to security consideration (Rsc), and Ratio of patches issued to address security vulnerabilities (Rp) [SEH08]. Other metrics are Mean time between security incidents, Mean-time to patch, Mean-time to complete changes, and Percent of changes with security exceptions [CIS10].
The quality of technical documentation can be enhanced by the support of documentation metrics. The metrics can help in evaluating through GQM approach that analyzes clone detection and test coverage analysis [W+10].

### 7.2.2 Product Metrics

Product metrics describe the product quality characteristics such as security. Security metrics have been defined on the basis of vulnerabilities and have been proposed on the basis of Common Vulnerabilities and Exposures (CVE), and Common Vulnerability Scoring System (CVSS). CVE is an industry standard for vulnerability and exposure names, and CVSS is a vulnerability scoring system designed to provide an open and standardized method for rating software vulnerabilities. Metrics defined is representative weakness of software i.e. those weaknesses that lead most of the vulnerabilities to be exploited by the attackers and are given below:

Security metrics, SM(s), for software s is given as

\[
SM(s) = \sum_{i=1}^{m} (P_n \times W_n) \quad \ldots \quad (7.1)
\]

where \( W_i \) (\( i = 1, 2, \ldots, m \)) is the severity of weakness in the software s and \( P_i \) (\( i = 1, 2, \ldots, m \)) is risk of the corresponding weakness. \( W_n \) is the severity of the weakness and is given by

\[
W_n = \frac{\sum_{i=1}^{K} V_i}{K} \quad \ldots \quad (7.2)
\]

where \( K \) is number of vulnerabilities of weakness W with corresponding base scores as \( V_i \). In Equation 7.1, \( P_n \) can be expressed as percentage and is given by occurrence of each weakness in the overall weakness.

\[
P_n = \frac{R_n}{\sum_{i=1}^{m} R_i} \quad \ldots \quad (7.3)
\]
In Equation 7.3, $R_n$ is the frequency of occurrences of each representative weakness. If $K$ is the number of weaknesses during $M$ months, $R_n$ can be expressed as

$$R_n = \frac{K}{M} \quad \ldots \quad (7.4)$$

SM(s) can range from 0 to 10 if following formula hold true for $P_n$.

$$\sum_{n=1}^{m} P_n = 1 \quad \ldots \quad (7.5)$$

Thus, software security metrics have been defined on the basis of representative weakness and can be calculated using Equations (7.1-7.5) [W+09]. Common Criteria (CC) defines seven assurance levels for a product varying from EAL1 to EAL7 that specifies how thoroughly the product is tested and ranks assurance on evaluated products. Higher assurance level only means that the product has undergone more number of security tests [CCMB12]. NIST presents system level controls addressing the information security program which includes access control, awareness and training, audit and accountability, etc. It provides some of the measures for audit processing failures that include software/ hardware errors, failures in the audit capturing mechanisms, and audit storage capacity being reached or exceeded [NIST800-53]. CIS security outcome and practice metrics define 20 metrics under six business functions. For example, Risk Assessment Coverage and Security Testing Coverage metrics assess the application security. The change management metrics include Mean time to complete changes, Percent of changes with security reviews, Percent of changes with security exceptions. Vulnerability management metrics include Mean-time to mitigate vulnerabilities, Number of known vulnerability incidences etc. [NEL10].

### 7.3 Proposed Security Metrics

It can be revealed from Section 7.2 that most of the researchers have developed metrics specific to some software development stage, while others
addressing SDP have not provided enough details regarding implementation of the metrics. Also, the metrics developed have not considered the reasons for insecurity. In this section, we propose security metrics that analyzes the security efforts during various software development stages. We denote development stages in terms of phases (Phase I … Phase VI) that stand for requirements gathering and analysis, software design, coding, system integration and testing, operations and maintenance, and documentation respectively.

7.3.1 Phase I - Requirements Gathering and Analysis

Gathering security requirements acts as foundation for secured software. Requirements gathering stage focuses on gathering security requirements from various stakeholders along with the other functional requirements. The security requirements can be gathered using Software Security Requirements Gathering Instrument (SSRGI) from the various stakeholders by the help of misuse cases, attack trees etc. as mentioned in Chapter 6. Most of the metrics defined for this stage are direct measures and can be considered as internal performance indicators for the requirements gathering team. The various metrics proposed are discussed below:

**Number of Security Requirements Gathered (NSRG)**

The number of security requirements gathered during the requirements gathering phase can be measured by metric NSRG. It consists of security requirements gathered from stakeholders as well as through the use of various tools. Requirements can be gathered using misuse cases, attack trees, best practices, security drivers etc. Some of the security drivers include Sarbanes Oxley, HIPPA etc. The requirements may relate to authentication and authorization, password, inactive sessions etc. and can be gathered from managers and other stakeholders (refer Chapter 6). NSRG may indicate the importance of security requirements during Phase 1. Let TR and SR denote security requirements be gathered with the help of tools and from stakeholders respectively, then

\[ NSRG = SR + TR \]
For example, $SR$ and $TR$ be 6 and 2 respectively for a web-based application, it implies that most of the requirements have been gathered with the help of stakeholders. The value of $NSRG$ focuses on the number of security requirements gathered with the help of tools and stakeholders.

**Security Requirements Recorded Deviations ($SRRD$)**

It describes the deviations from security requirements. The deviations can be measured after designing the software system based on the requirements specification. If $SRRD$ is 0, then all the security requirements have been incorporated in the software product.

**Security Requirements stage Security Errors ($SRSE$)**

It specifies the security errors encountered during requirements gathering. It measures errors as a result of incorrect or incomplete security requirements gathered during this stage. If for a web-based project the value of $SRSE$ is 1, it indicates the number of security related errors due to security requirements.

**Security Requirements Gathering Indicators ($SRI$)**

Indicators on Requirements gathering and analysis stage or $SRI$ explain the impact of security requirements on number of security breaches. For example, in a client/server system, the value of $SRI$ as 3 provides an indicator regarding number of security breaches in first year of deployment.

**7.3.2 Phase II - Software Design**

The design identifies the work that software can perform. It mainly illustrates the various components of the software and its interrelationships with each other and the surroundings. Even when the designers are proactive, the software may not be fully secured. The design should be able to reduce the attack surface of the software making it attack resistant and tolerant. In the software design stage, the requirements gathered must be considered for design thereby helping to produce software as per specifications. Requirements and design are indispensable phases of software development. Requirements specifications are considered for design proposal whereas analysis of design generates the need for
further requirements [NUS01]. The design phase must consider non functional security requirements (discussed in Chapter 6) that includes security related architectural requirements such as robustness, privacy, integrity etc. The security aspects, exception handling and error messages can be considered during design stage. Identification and authorization of users should be included during the design so as to make end-product more secure (refer Chapter 6). The various proposed metrics at the design stage are mentioned below:

**Security Requirements Statistics (SR<sub>3</sub>)**

Security Requirements Statistics or SR<sub>3</sub> indicates the percent of security requirements gathered reflected in the design stage. If NSRD is Number of Security Requirements considered for Design, SR<sub>3</sub> can be defined as the ratio of NSRD and NSRG expressed as percentage. The value of SR<sub>3</sub> as 100% indicates that all of the gathered security requirements have been considered for the design. If value of NSRD and NSRG is 5 and 8 respectively for a web-based system, then SR<sub>3</sub> implies 62.5% security consideration during design.

**Design Tools and Test Effectiveness (DTTE)**

Let Secure Analysis of Design by Tools (SADT) indicates the use of the security analysis tools that takes into account security incorporated during the design of the software. A number of tools can be used to analyze design for security such as misuse cases, threat analysis, attack patterns etc. that help to consider the security requirements. Also, the Number of Test cases for Secured System Design (NTSSD) indicates the use of test cases as tools to analyze the secured design aspects. The Number of Aspects for Secured Design (NASD) represents the security driven aspects. NASD designates the design aspects considered from architectural design requirements, implicit requirements such as exception handling, input validation, authenticity checks etc. Thus, NASD can be expressed as sum of Number of Aspects using architectural Design Standards (NADStd), Number of Implicit Security Design Aspects (NISDA) and NSRD. NISDA comprises of implicit security requirements such as exception handling, input validations, check for authentication etc. These design aspects can be
considered for building secured software. Let $DTTE$ is represented as ratio of the sum of tools to analyze security and $NASD$ and is indicated below:

$$DTTE = (SADT + NTSSD)/NASD$$

where $NASD \neq 0$

or, $DTTE = (SADT + NTSSD)/(NADStd + NISDA + NSRD)$

If the security tools and testing have not been considered during design stage, then the effectiveness becomes zero. Based on the statistics of $SADT$, let $NTSSD$ and $NASD$ as 2, 5 and 17 respectively, then $DTTE$ is revealed as 0.41.

**Number of Design stage Security Errors (NDSE)**

It indicates the design flaws by measuring the security errors due to design stage as judged by the software development team. $NDSE$ specifies security consideration by the development team as one of the major design aspects to achieve secured software. For example, the value of $NDSE$ for a client/server based system indicates that two security errors mainly owe to negligence of security due to design stage.

**7.3.3 Phase III - Coding**

During the coding phase, security can be implemented by validating input, output, reused code, following good programming practices and coding standards. The metrics of this stage judge the implementation level details.

**Percent of Secure Coding Aspects (PSCA)**

It identifies the percentage of security aspects considered during coding as per the design. The security coding aspects can be derived from Secure Development Requirements ($SDR$) and coding guidelines (refer Chapter 6). Some of the security aspects include exception handling, input and output validation, displaying error messages, session management, user authorization etc. $PSCA$ can be expressed as ratio of Number of Security Coding Aspects ($NSCA$) to $NASD$ expressed in percentage.

$$PSCA = NSCA/ NASD * 100$$
For instance, let NCSA and NASD are 20 and 44 respectively, then PSCA indicates 45.5% of secure coding aspects consideration.

**Percent use of Coding Standards (PCS)**

The metric indicates the use of coding standards. It acts as an estimate for the metric Number of Security Errors (NSE) coding standards are considered during code implementation.

**Number of Security Errors (NSE)**

The coding errors creep in due to unsafe functions, improper use of pointers, allocation and deallocation of memory, improper error handling, illogical access control, breaking the design into different modules without considering the after effects, typographical errors etc. Errors are also due to code used from other libraries [JAI10]. NSE represents the flaws that can be expressed as the sum of coding errors and errors due to code from other libraries thereby identifying the trustworthiness of the code. As an instance of NSE and PCS in a medium sized web-based application, 7 security errors owe to 20% use of coding standards.

**7.3.4 Phase IV - System Integration and Testing**

Testing the software for security is aimed at evaluating the security of the software, check if it meets the specified security requirements, and identify the left security vulnerabilities. Security testing focuses on security functionality of the software product based on the various types of requirements gathered [JI11b]. Even if the design is secure, choices provided during implementation may affect security of the software such as the development platform chosen and the use of functions available on platform. The test procedures should be able to detect flaws in the software that leads to security breaches. The test cases must display all the error messages, check input buffers, connection to all ports etc. [TW02].

We now discuss the metrics of system integration and testing stage as follows:
Security Requirements for Testing (SRT)

The metric $SRT$ indicates the ratio of the security requirements tested and $NSRG$, where $NSRG \neq 0$. This may ensure the development of system as per the security requirements being gathered during requirements gathering stage. If all the security requirements have been tested during this phase, $SRT$ can be indicated by 1 as in an example a of web-based application.

Process Effectiveness (PE)

Process Effectiveness reflects the security considerations during the development of the software. It can be evaluated as the ratio between the Numbers of security vulnerabilities discovered ($N_{VD}$) and Count of Modules undergone security testing ($M_{ST}$).

$$PE = \frac{N_{VD}}{M_{ST}}$$

where $M_{ST} \neq 0$

For example, let $N_{VD} = 3$ and $M_{ST} = 10$ then for a web-based project, 0.3 security flaws per module have been identified during testing.

Security Testing Ratio (STR)

The metric $STR$ can be expressed as the ratio of modules undergone security testing to the total number of modules. This ratio helps in judging the modules subject to security testing. For example, $STR$ having value of 0.75 implies that 75% of all the modules have gone through security testing.

7.3.5 Phase V - Operations and Maintenance

A number of security flaws can also creep in during operations and maintenance phase as a result of the changes in the system or environment. To reduce security related flaws, conduct periodic risk review and vulnerability assessments, provide security awareness programs, and perform auditing, logging, monitoring, and archiving. The proposed metrics may help ensure that the changes shall be completed within time, the threats found shall be taken care of, and increase the awareness of new vulnerabilities.
Mean Time to Complete Security Changes (MTCSC)

The metric provides an indication regarding failure of system due to security flaw. It can be estimated by the number of security failures and mean time taken to repair the flaws. The metric is recorded over certain time period. MTTSF and MTTR are obtained as average of all security related failure of a system and time taken to repair the failure [PRE01]. The metric may also support in periodic risk assessment. MTCSC can be calculated as follows:

$$MTCSC = MTTSF + MTTR$$

where, MTTSF is Mean Time to Security Failure

MTTR is Mean time to Repair

Percent of Changes with Security Exceptions (PCSE)

It indicates the percentage of configuration or system changes that received an exception to existing security policy [LINK67]. This metric is a ratio of counts of completed changes with security exceptions and completed changes multiplied by 100. It helps to notice the security concern during the maintenance phase.

Rate of Vulnerability Assessments (RVA)

It indicates the number of security assessments performed by the development team in a specific time period. This metric may be defined as the number of vulnerability assessments during one quarter. For example, RVA can be expressed as 4 per quarter may imply that the assessments for vulnerabilities have been conducted 4 times in one quarter.

Ratio of changes due to security consideration (Rsc)

The metric Rsc measures the number of changes in the system requirements due to new set of security requirements [SEH08]. It also includes the need for system patches. It is calculated as the ratio of the number of changes triggered by new security requirements and the number of changes of the entire
system. For instance, for a client/ server based system, the value of $Rsc$ implies 0.33 changes in new system requirements owe to security vulnerabilities.

### 7.3.6 Phase VI - Documentation

The document must mention the security controls implemented for the proper functioning of the software along with the responsibilities of the various users and other functional and non-functional requirements.

**Number of Security Controls Mentioned (NSCM)**

Number of Security Controls Mentioned or NSCM indicates the security controls implemented in the software such as access control, disable inactive accounts automatically, account recovery method, etc. The value of NSCM such as 5 helps to identify the security controls being documented.

In Table 7.1, the dependence of metrics on phases of SDP has been depicted.

### 7.4 Effectiveness Factor of Security

In this section, we attempt to develop effectiveness factor using the metrics as proposed at each phase of software development in Section 7.3. The effectiveness factor can help to judge the security efforts during various development phases. Based on security metrics, we collected data for fifteen live projects. These projects have been developed using platforms such as Java, PHP, .NET, MySQL, C#, AJAX, Perl, Visual Basic, ASP .NET 4.0, Java Script, SQL Server, Apache, Red Hat Linux, MSDE, Jquery, Microsoft SSRS and Unix shell script and are based on various network design such as client/ server system, web-based, static web sites, mobile and desktop based systems. Also, the size of projects here range from small to very large.
Table 7.1: Dependency of Metrics at Different Phases of SDP

<table>
<thead>
<tr>
<th>S.No</th>
<th>Phases</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
<th>V</th>
<th>VI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NSRG</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SRRD</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SRSE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SRI</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SRs</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>DTTE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NDSE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PSCA</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PCS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>NSE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SRT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>PE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>STR</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>MTCSC</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>PCSE</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>RVA</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Rsc</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>NSCM</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.4.1 Phase I: Factors

The security efforts at requirements gathering phase can be judged in two stages. At first stage, we identify the existence of relationship between each pair of metrics of the all development phases. At second stage, the effectiveness factor for the software development stages is being developed.

Stage I – Relationship among metrics

We propose null hypothesis stating that each pair of metrics, do not belong to the same sample unit. The null hypothesis can be stated as follows:

H01: SRRD is unrelated to NSRG.
H02: SRSE is unrelated to NSRG.
H03: SRI is unrelated to NSRG.
H04: SRRD is unrelated to SRSE.
H05: SRRD is unrelated to SRI.
H06: SRSE is unrelated to SRI.

The null hypothesis is evaluated using paired t-test as follows:

\[
t = \frac{\overline{X}_1 - \overline{X}_2}{S_{X_1X_2} \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \]

… (7.6)

Where, \( S_{X_1X_2} = \sqrt{\frac{(n_1 - 1)S^2_{X_1} + (n_2 - 1)S^2_{X_2}}{n_1 + n_2 - 2}} \)

Here, \( \overline{X}_1 \) and \( \overline{X}_2 \) represents the means of the metrics of sample units. The number of sample units are \( n_1 = n_2 = 15 \). The paired t-test identifies if the sample means differ significantly or not. The level of significance is evaluated at 5%.

We applied paired t-test among the metrics of requirements gathering stage to test the hypotheses. The t-test results reject the null hypotheses H01, H02 and H03 with t-values 3.2248, 3.1873 and 3.0854 respectively indicating that the
metrics belong to the same sample unit and may be related to each other. Thus, SRRD, SRSI, and SRI are related to NSRG.

**Stage II - Effectiveness Factor**

Let us consider that there exists independent metric X and n number of dependent metrics as y₁, y₂,…, yₙ. Let Y be the sum of all n metrics representing total effect by independent metric X. We use Least Square Method to identify the relationship.

\[ Y = C - \alpha \cdot X \] \hspace{1cm} (7.7)

where, C is constant and is calculated as the sum of all intercepts generated for each pair of metrics indicating relationship. \( \alpha \) represents the effectiveness factor of X on Y and can be computed using Equation 7.7 as

\[ \alpha = (C - Y)/X \] \hspace{1cm} (7.8)

The domain of metrics may consist of independent and dependent metrics. Based on the practicability, dependent metrics may be SRRD, SRSE, and SRI whereas NSRG may be considered as independent metric X.

Applying Equation 7.7, the value for constant C for metrics pair NSRG and SRRD, NSRG and SRSE, and NSRG and SRI are 1.3671, 1.8423, and 2.0327 respectively. Thus, using Equation 7.8, the combined effect of independent variable X (i.e. NSRG) on dependent variable Y (i.e. SRRD, SRSE and SRI) can be given as

\[ \alpha = (5.2421 - Y)/NSRG \] \hspace{1cm} (7.9)

where \( NSRG \neq 0 \)

In Equation 7.9, the constant C is evaluated as combined effect of X on Y.

**Special Cases of Effectiveness Factor**

\( \alpha = -1 \): Effectiveness -1 implies that number of dependent metrics are large enough such that \( Y - C = X \). This also shows that the entire software development process may have not considered security.
\( \alpha = 0: \) Effectiveness 0 implies that sum of dependent metrics equals C while \( X \) can assume any value. It shows that even though security requirements have not been gathered, the security deviations from requirements and errors are recorded, leading to trivial situation.

\( \alpha = 1: \) Effectiveness can be 1 when \( Y=C-X \). This may lead to trivial case when the numbers of security requirements gathered are high as \( Y \) may acquire negative value.

\( \alpha > 1: \) The values greater than 1 implies that it may be difficult to judge the effectiveness of the security.

### 7.4.2 Phase II: Factors

The null hypothesis \( H_0 \) states that the metrics \( DTTE \) is unrelated to \( NDSE \) i.e., there is no significant difference between the means of \( DTTE \) and \( NDSE \). The null hypothesis is evaluated using t-test (Equation 7.6) and is rejected at 5% level of significance \( (t = 3.268) \) thereby indicating relation between the two. The metrics pair is further evaluated resulting in negative correlation of -0.7.

As evident from the discussion in Section 7.3.2, the metrics \( DTTE \) and \( NDSE \) are inversely proportional to each other.

\[
NDSE = \beta / DTTE
\]

\[
\beta = NDSE \cdot DTTE
\]  \( \cdots \)  \( (7.10) \)

Here, \( \beta \) denotes the effectiveness of design stage where \( \beta \geq 0 \). \( \beta = 0 \) implies that design stage security errors are nil. From Equation 7.10, it is evident that \( \beta \) shall lie between 0 and 1. The value for \( \beta \) can be interpreted as highly effective (0-0.25), moderately effective (0.26-0.50), effective (0.51-0.75), ineffective (0.75-1.00) and very ineffective (>1.00) security considerations during design phase.

### 7.4.3 Phase III: Factors

The secure coding may reduce the number of errors if the coding standards and secure design are looked upon. The null hypothesis \( H_0 \) states that sum of \( PSCA \) and \( PCS \) is unrelated to \( NSE \). Using t-test (Equation 7.6), the null
hypothesis is rejected at 5% level of significance (t=3.37) indicating that NSE is result of consideration of PSCA and PCS during coding.

**Effectiveness Factor**

Let there exists one independent variable X and one dependent variable Y as NSE. To establish the relation between X on Y, we apply Least Square Method to identify the effect of X on Y. Thus,

\[ Y = C - \gamma \cdot X \]

where C is constant and \( \gamma \) is effectiveness of X on Y. \( \gamma \) can be calculated as

\[ \gamma = \frac{(C-Y)}{X} \quad \ldots \quad (7.11) \]

where X>0

Using Equation 7.11, the combined effect of PSCA and PCS (independent variable) on NSE (dependent variable) is given by

\[ \gamma = \frac{(8.4022 - \text{NSE})}{(\text{PSCA} + \text{PCS})} \quad \ldots \quad (7.12) \]

Special cases –

\( \gamma = -1 \): Using Equation 7.12, the sum of the value of independent metrics and the constant C equals NSE. It may also result in significantly large NSE.

\( \gamma = 0 \): If \( \gamma = 0 \) then NSE=8.4022. It indicates NSE to be moderately high (approx. 8.4) and is not dependent on sum of PACS and PCS metrics. The value may not possible as NSE cannot be a real number.

\( \gamma = 1 \): Effectiveness 1 indicates that Y = C-X, i.e. number of independent metrics should be less than 8.4. It also tells that if X increases, then Y decreases. Lower value of Y may reveals that security is considered from design stage as well as security coding standards.
7.4.4 Phase IV: Factors

Security testing uncovers the security flaws of the software product. Phase IV metrics helps in determining the effectiveness of security testing. The null hypotheses for implementation and testing stage can be stated as follows:

- $H_{09}$: $PE$ is unrelated to $STR$
- $H_{010}$: $PE$ is unrelated to $SRT$

Using Equation 7.6, the hypotheses $H_{09}$ and $H_{010}$ are evaluated using t-test. The null hypotheses $H_{09}$ and $H_{010}$ are accepted at 5% level of significance having t-values as 0.6346 and 0.2878 respectively, indicating that $PE$ is not determined by $STR$ and $SRT$.

7.4.5 Phase V: Factors

The null hypothesis $H_{11}$ states that the metric $PCSE$ is unrelated to $Rsc$. $H_{11}$ is analyzed at 5% level of significance using paired t-test using Equation 7.6. $H_{11}$ is accepted revealing that the two are unrelated ($t = 1.045$) and hence effectiveness of the security efforts of this stage cannot be judged.

7.5 Case Studies

Based on the metrics developed in Section 7.3 and effectiveness factors in Section 7.4, we present three case studies demonstrating the effectiveness of security efforts during the software development phases. The case studies are based on Web, client/server and single user desktop based systems.

**Case I: Web-based System**

Journal Publishing System (JPS) is a Web-based publishing system designed for a regional society. The system assists editors by automating the article submission by contributors, article review and publishing process thereby maximizing efficiency. The software facilitates communication between editors, authors and reviewers via E-mail. Preformatted reply forms are used for communication between the editor, the author and the reviewer. JPS is a Red Hat
Linux based system developed using PHP and MySQL on Apache server. The measures and metrics of SDP stages for JPS are illustrated in Table 7.2.

**Case 2: Client/ Server based System**

Patient Management System or PMS is a LAN based client/ server system that allows the hospitals to keep track of the patient’s data. It aids in the management of personalized patient lists, physicians etc. The system registers doctors, nurses, social workers, and dieticians as well as assists them in maintaining details of the patients. The system supports internal messaging among various users. It allows generating the list of patients with their ailments, medications and tests (if suggested), and the doctor and nurses in-charge of the same. The system also permits to view the case history of the patients and fix appointment with the doctor. The system is developed using .NET Framework on SQL Serverr 2008. The metrics for PMS are presented in Table 7.2.

**Case 3: Desktop based System**

Learning System (LS) is a desktop based system, which has set of innovative educational applications for students, teachers and educators. These applications are classified into various categories such as Create, Collaborate, Teach, Research etc. The application can be used in different languages. LS is developed using .NET platform. The metrics for Learning System are illustrated in Table 7.2.

**7.6 Results and Conclusion**

In this chapter, we first tried to study the available product and process security metrics for the software development stages. Next, process metrics for secured software development process have been established for various stages. Lastly, the effectiveness factors for each of the development stages have been realized to judge the effectiveness of security efforts of the development team.

In Section 7.2, we reviewed the available metrics for different life cycle activities of SSDP. It has been observed that most of the process metrics do not
Table 7.2: Trend of Metrics for Different Projects

<table>
<thead>
<tr>
<th>S.No</th>
<th>Software Development Stages</th>
<th>Projects</th>
<th>Web-based (JPS)</th>
<th>Client/Server (PMS)</th>
<th>Desktop (LS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements Gathering and Analysis</td>
<td>NSRG</td>
<td>8</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>SRRD</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>SRSE</td>
<td>1</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>SRI</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Software Design</td>
<td>SRs</td>
<td>62.5%</td>
<td>72.7%</td>
<td>80%</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>DTTE</td>
<td>0.41</td>
<td>0.39</td>
<td>0.38</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>NDSE</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Coding</td>
<td>PSCA</td>
<td>70.59%</td>
<td>45.5%</td>
<td>30.8%</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>PCS</td>
<td>20%</td>
<td>25%</td>
<td>20%</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>NSE</td>
<td>7</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>System Integration and Testing</td>
<td>SRT</td>
<td>1</td>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>PE</td>
<td>0.5</td>
<td>1.11</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>STR</td>
<td>0.40</td>
<td>0.75</td>
<td>0.6</td>
</tr>
<tr>
<td>14</td>
<td>Operations and Maintenance</td>
<td>MTCSC</td>
<td>17 days</td>
<td>16 days</td>
<td>NA</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>PCSE</td>
<td>30%</td>
<td>55.5%</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>RVA</td>
<td>4/qtr.</td>
<td>3/qtr</td>
<td>NA</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Rsc</td>
<td>0.2</td>
<td>0.33</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Documentation</td>
<td>NSCM</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
provide sufficient implementation details considering SDP stages. Moreover, the process metrics do not focus on the security issues of SDP phases. Most of the metrics access security late in the software development process making it difficult to fix the problems early.

The various security metrics have been established on the basis of the security needs of the different development lifecycle activities in Section 7.3. The metrics instituted are \textit{NSRG, SRRD, SRSE}, and \textit{SRI} for requirements gathering stage while design stage metrics are \textit{SRs, DTTE}, and \textit{NDSE}. The coding stage metrics comprise of \textit{PSCA, PCS} and \textit{NSE}, and testing stage metrics are \textit{SRT, PE}, and \textit{STR}. The metrics established for operations and maintenance stage are \textit{MTSCS, PCSE, RVA} and \textit{Rsc}, whereas document stage metrics constitute of \textit{NSCM}. The trend of metrics for the various projects is depicted in Table 7.2. Based on the trend, following observations can be made:

- \textit{SRRD} is zero for web-based and desktop based systems indicating that there are no deviations from the security requirements during development process. For a client/ server based system, \textit{SRSE} implies more deviations from security requirements in a client/ server based system as compare to the other two systems..

- The value of \textit{SRs} illustrates moderately high security requirements aforethought during design of client/ server and desktop based systems. During the design phase of a web-based system, \textit{DTTE} indicates moderate to low use of design tools and testing effectiveness. It also illustrates moderate to low consideration of \textit{SADT, NTSSD} and \textit{NASD}.

- Low \textit{PSCA} and \textit{PCS} have resulted in high \textit{NSE} in client/ server based software. \textit{NSE} may also indicate security errors due to low experienced software professionals in secured coding leading to fifteen security errors although security requirements gathered are high.
For a web-based system and desktop based system, \( SRT \) indicates that all the security requirements being gathered have been tested for security. \( PE \) reveals high to low errors per module for web-based and client/server based system. \( STR \) shows that 40% modules have undergone security testing for a web-based system.

Operations and maintenance phase metric \( PCSE \) implies that the out of all changes expected in a web-based system, only 30% have been related to security policies. It shows that moderately high security consideration during earlier stages of software development resulting in less security exceptions. \( Rsc \) reveals 20% and 33.3% changes in JPS and PMS are due to security related issues.

\( NSCM \) indicates low security controls that do not have much importance during documentation for all types of systems.

Although NSRG is high for client/server based system, security is given low consideration during all the stages of software development.

On the basis of measures and metrics of varied industrial projects (refer Annexure 7.2), we developed the Effectiveness Factors (EFs) for requirements gathering, design and coding stages in Section 7.4. The effectiveness factors for the case studies discussed in Section 7.5 are illustrated in Table 7.3. Based on Table 7.3, following observations can be made:

- The \( \alpha \) values (using Equation 7.9) for web-based, client/server based and desktop based systems illustrates that security considerations are moderately high and low for web-based and client/server based systems respectively during requirements gathering stage. The \( \alpha \) value for desktop based systems implies that the effectiveness of security considerations cannot be judged. This is inline with the fact that security is a major issue in web-based systems and low for desktop based systems.
Table 7.3: Effectiveness Factors

<table>
<thead>
<tr>
<th>S.No</th>
<th>Projects</th>
<th>Web-based (JPS)</th>
<th>Client/Server (PMS)</th>
<th>Desktop</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>α</td>
<td>0.405</td>
<td>-0.26</td>
<td>1.048</td>
</tr>
<tr>
<td>2</td>
<td>β</td>
<td>0.41</td>
<td>0.78</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>γ</td>
<td>0.0155</td>
<td>-0.101</td>
<td>0.087</td>
</tr>
</tbody>
</table>
The $\beta$ values (using Equation 7.10 and Table 7.3) for web-based and client/server-based system indicates moderately effective and ineffective security considerations during design stage. The $\beta$ for desktop based system signifies that no security errors have been detected in the design.

The $\gamma$ values (using Equation 7.12) for web-based, client/server, and desktop-based systems indicate that the coding errors are low in web and desktop-based system as compared to client/server-based system. The values may be the result of the coding errors.

The metrics can assess security considerations more efficiently when measured during software development process. We analyze various security metrics available in the literature for product as well as process. Further, we developed a set of metrics in view of the security issues addressing the stages of software development process. The security metrics developed aims to evaluate the efforts of the various software development stages regarding security consideration. The metrics can help the development team to judge its performance for security and may focus on gathering more security requirements, consider more of implicit security aspects, make use of standards, increase input validation etc. by providing security training to the development team. Metrics may also act as a checklist for increasing security aspects of the software product. It may support the developers to enhance the software development process. It shall help predict the phases that require more security forethought during development. It may be concluded that the metrics and effectiveness factors provide a way to assess and visualize security during the software development process.

### 7.7 Summary

Security metrics serve as a basis for assessment of security efforts throughout the software development process as well as quantitative measurement of the degree of trustworthiness of the system. There exists many metrics focusing
on some specific stage of development process or on the product as a whole. The available metrics have been analyzed for the various stages of software development process as well as the product as a whole to realize the metrics at a glance. Most of metrics do not consider the security issues of development stages and can be assessed late in SDP. Moreover, sufficient details have not been provided so as to analyze the metrics. Hence, we instituted a set of metrics covering all the stages of SDP as well as effectiveness factors to analyze and judge the security efforts. The metrics have been analyzed with the help of case studies based on web, client/ server and desktop software systems. It revealed that security is moderately considered during web-based software systems, while security has been given moderately low consideration during development of client/ server based systems. Thus, our metrics may support the development team to evaluate and monitor the security efforts throughout the software development stages.