FALL 2014 GROUP 1

CSCI 50600

IMPLEMENTATION PHASE METRICS MEASUREMENTS

Twitter Project

Patrick Burton

Arvind Nair

Lakshmi Swathi Chavvakul

IUPUI

**Implementation Phase Metric Measurement:**

For the implementation phase, the metrics have been measured as per on the software implemented.

We have measured 3 metrics in the implementation phase:

1. Sufficiency

2. Security

3. Robustness

**Sufficiency:**

**Definition:**

Percentage of detailed requirements that are implemented.

**Meaning:**

Sufficiency in the implementation phase shows us how much of the requirements in a particular iteration has been implemented. This gives us a good idea of the leftover requirements that need to be implemented in the next iteration.

**Formula:**

Sufficiency= Percentage of detailed requirements implemented X 100

Total Requirements in that iteration

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration Number | Requirement ID | Requirement Implemented? | % of Requirements Implemented in particular Iteration |
| Iteration 1 | ID 1 | ✓ | 100% |
| ID 2 | ✓ |
| ID 3 | ✓ |
| ID 4 | ✓ |
| ID 5 | ✓ |
| ID 6 | ✓ |
| ID 7 | ✓ |
| ID 8 | ✓ |
| ID 9 | ✓ |
| ID 10 | ✓ |
| ID 11 | ✓ |
| ID 12 | ✓ |
| ID 13 | ✓ |
| ID 14 | ✓ |
| ID 15 | ✓ |
| Iteration 2 | ID 16 | ✓ | 100% |
| ID 17 | ✓ |
| ID 18 | ✓ |
| ID 19 | ✓ |
| Iteration 3 | ID 20 | ✓ | 100% |
| ID 21 | ✓ |
| ID 22 | ✓ |
| ID 23 | ✓ |
| ID 24 | ✓ |
| Iteration 4 | ID 25 | ✓ | 100% |
| ID 26 | ✓ |
| ID 27 | ✓ |
| ID 28 | ✓ |
| ID 29 | ✓ |
| ID 30 | ✓ |
| ID 31 | ✓ |

Table 1 : Sufficiency Metric Measurement

Figure 2 : Sufficiency Metric Bar Graph

Figure 2 : Sufficiency Metric Line Graph

**Analysis of Results:**

The sufficiency metric in our project shows that at each iterations we have implemented the requirements specified for that iteration and there are no leftover requirements that need to be implemented. So, we have meet all requirements at each iteration.

**Security:**

**Definition:**

This metric measures as to how strong security is unbreakable in our implementation.

**Meaning:**

This metric measures how security breaching cannot occur in our system. The score of 2 points means that the security breach is very hard to conceive or implement.

**Formula:**

0: easy

1: not easy but conceivable

2: not conceivable

% of Security for each iteration= Sum of scores for each item X 100

2 X Number of Items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration Number | Item Type | Item Description | Score | % of Security for each iteration |
| Iteration 1 | Username | No SQL Injection | 2 | 62.50% |
| Password | No SQL Injection | 2 |
| Secure Login | Not HTTPS | 0 |
| File System Storage of Data | Not Secure as can be read and copied easily. | 1 |
| Iteration 2 | Username | CAS so difficult to breach | 2 | 87.50% |
| Password | CAS so difficult to breach | 2 |
| Secure Login | CAS so secure | 2 |
| File System Storage of Data | Not Secure as can be read and copied easily. | 1 |
| Iteration 3 | Username | CAS so difficult to breach | 2 | 100% |
| Password | CAS so difficult to breach | 2 |
| Secure Login | CAS so secure | 2 |
| MySQL to store data | Has No SQL Injection and when combined with CAS very secure. | 2 |
| Iteration 4 | Username | AWS + CAS Due to CAS difficult to breach | 2 | 100% |
| Password | AWS + CAS Due to CAS difficult to breach | 2 |
| Secure Login | AWS + CAS Due to CAS secure. | 2 |
| MySQL to store data | Has No SQL Injection and when combined with CAS very secure. | 2 |

Table 2 : Security Metric Measurement

Figure 3 : Security Metric Bar Graph

Figure 4 : Sufficiency Metric Line Graph

**Analysis of Results:**

In our software as per the iterations, the security becomes difficult to breach due to CAS authentication. Thus, as it is difficult to find out ways which are implementable to breach CAS security and also AWS security.

**Robustness:**

**Definition:**

An implementation's robustness is the extent to which it handles anomalous input(i.e., input whose form or content is unknown.

**Meaning:**

This metric measures how the input fields respond to the anomalous inputs. This would give us a general idea as to how well the system would respond or behave when anomalous or junk is typed into the textboxes and what we need to change accordingly in the next iterations.

In our software, robustness is measured using BVA and in our application we have four input fields i.e., username, password or CAS Authentication Box, type box and display box.

**Formula:**

No robustness: 0

Some robustness: 0.5

complete: 1

% of Robustness= Sum of degree of robustness for all input fields in the iterations X 100

Number of Input fields

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iteration Number | Item Type | Tried the following | Tried the following | Actual Behavior | Score | % of Robustness for Each Iteration |
| Iteration 1 | Username | 1. SQL Injection  2.Erroneous Input | 1. SQL injection not possible.  2.Cannot login if incorrect. | 1. SQL injection not possible.  2.Cannot login if incorrect. | 1 | 87.50% |
| Password | 1. SQL Injection  2.Incorrect password | 1.SQL injection not possible.  2.Cannot login if incorrect. | 1.SQL injection not possible.  2.Cannot login if incorrect. | 1 |
| Type box | 1. SQL Injection  2.Type too many characters. | 1.SQL injection not possible  2.No limit to characters | 1.SQL injection not possible  2. Stops on limit reached. | 0.5 |
| Display box | 1. SQL Injection | 1. SQL injection not possible | 1. SQL injection not possible | 1 |
| Iteration 2 | Username | 1. SQL Injection | 1. SQL injection not possible | 1. SQL injection not possible | 1 | 87.50% |
| CAS Authentication box | 1. SQL Injection | 1. SQL injection not possible (CAS Security) | 1. SQL injection not possible | 1 |
| Type box | 1. SQL Injection  2.Type too many characters. | 1. SQL injection not possible.  2. No limit to characters. | 1. SQL injection not possible | 0.5 |
| Display box | 1. SQL Injection | 1. SQL injection not possible | 1. SQL injection not possible | 1 |
| Iteration 3 | Username | 1. SQL Injection | 1. SQL injection not possible | 1. SQL injection not possible | 1 | 87.50% |
| CAS Authentication box | 1. SQL Injection | 1. SQL injection not possible (CAS Security) | 1. SQL injection not possible | 1 |
| Type box | 1. SQL Injection  2.Type too many characters. | 1. SQL injection not possible  2. Stops on limit reached. | 1. SQL injection not possible  2. Stops on limit reached. | 1 |
| Display box | 1. SQL Injection  2. While at top reading check to see if it goes down on update. | 1. SQL injection not possible.  2. Scrolls down when updates while reading at top. | 1. SQL injection not possible  2. Does not scroll down when it updates while reading at top. | 0.5 |
| Iteration 4 | Username | 1. SQL Injection | SQL injection not possible. | 1. SQL injection not possible | 1 | 100% |
| CAS Authentication box | 1. SQL Injection | SQL injection not possible (CAS Security) | 1. SQL injection not possible | 1 |
| Type box | 1. SQL Injection  2.Type many characters | SQL injection not possible    2. Stops on limit reached. | 1. SQL injection not possible  2. Stops on limit reached. | 1 |
| Display box | 1. SQL Injection  2. While at top reading check to see if it goes down on update. | SQL injection not possible    2. Does not scroll down when it updates while reading at top. | 1. SQL injection not possible  2. Does not scroll down when it updates while reading at top. | 1 |

Table 3 : Robustness Metric Measurement

Figure 5 : Robustness Metric Bar Graph

Figure 6 : Robustness Metric Line Graph

**Analysis of Results:**

The robustness in the first 3 iterations remains the same at 87.5% but in the last iteration it becomes 100% because all the problems in the input fields are fixed. This metric shows that the input fields in the last stage are robust and resistant to anomalous inputs.

**Conclusion:**

The Implementation Phase metrics measurements must show an increasing trend as these metrics are directly related to the quality of the software produced. For Example, an increasing security metric means that the security has become stronger at each iteration. At each Iteration, these metrics increasing show that the quality of software with respect to that part of the software has increased. It tells us what aspect of the software must be enhanced to improve the software quality and we make changes accordingly. e.g., We saw that security metrics were less and designed accordingly in the next iterations. Hence, Implementation metrics are very important indication of software quality and also we will come to know what to change in the next iterations.

**References:**

1. *Software Engineering Modern Approaches Second Edition* Eric J. Braude and Michael E. Bernstein, Wiley Publications.

2. B. W. Boehm, J. R. Brown, M. Lipow, Quantitative Evaluation of Software Quality, TRW Systems and Energy Group, (1976).