FALL 2014 GROUP 1

DESIGN PHASE METRICS MEASUREMENTS

TWITTER PROJECT

Patrick Burton

Arvind Nair

Lakshmi Swathi Chavvakul

CSCI 50600

FALL 2014 GROUP 1

Design Phase Metric Measurements:

For the design phase, the metrics have been measured as per the design document.

We have measured the following 3 metrics:

1. Flexibility
2. Understandability
3. Sufficiency
4. Security

Flexibility Metric Measurement:

Definition:

It is defined as the relative ease to extend a class or to add a new functionality.

Meaning:

Flexibility tells us as to how far we can make additions to our software without much change. This gives us a good measure of the software being open to changes. High flexibility means that additions can be made on the software very easily and hence can be developed further.

Formula:

0: anticipated additional requirements require extensive change to the design.

10: most anticipated requirements do no change to design.

$$\%\text{ of Flexibility} = \frac{\text{Total points for each iteration X 100}}{\text{10 X Number of items}}$$

Iteration Number	Item Type	Points	Total Points (% of Flexibility)
Iteration 1	1. Changing of security features.	6	60%
Iteration 2	1. Changing from file system to MySQL.	6	75%
	2. Adding more features in tweet functionality.	9	

Iteration 3	1. Deploying on AWS.	9	90%
	2. Adding more features like timestamps.	9	
Iteration 4	1. Future additions.	9	90%

Table 1 : Flexibility Metric Measurement

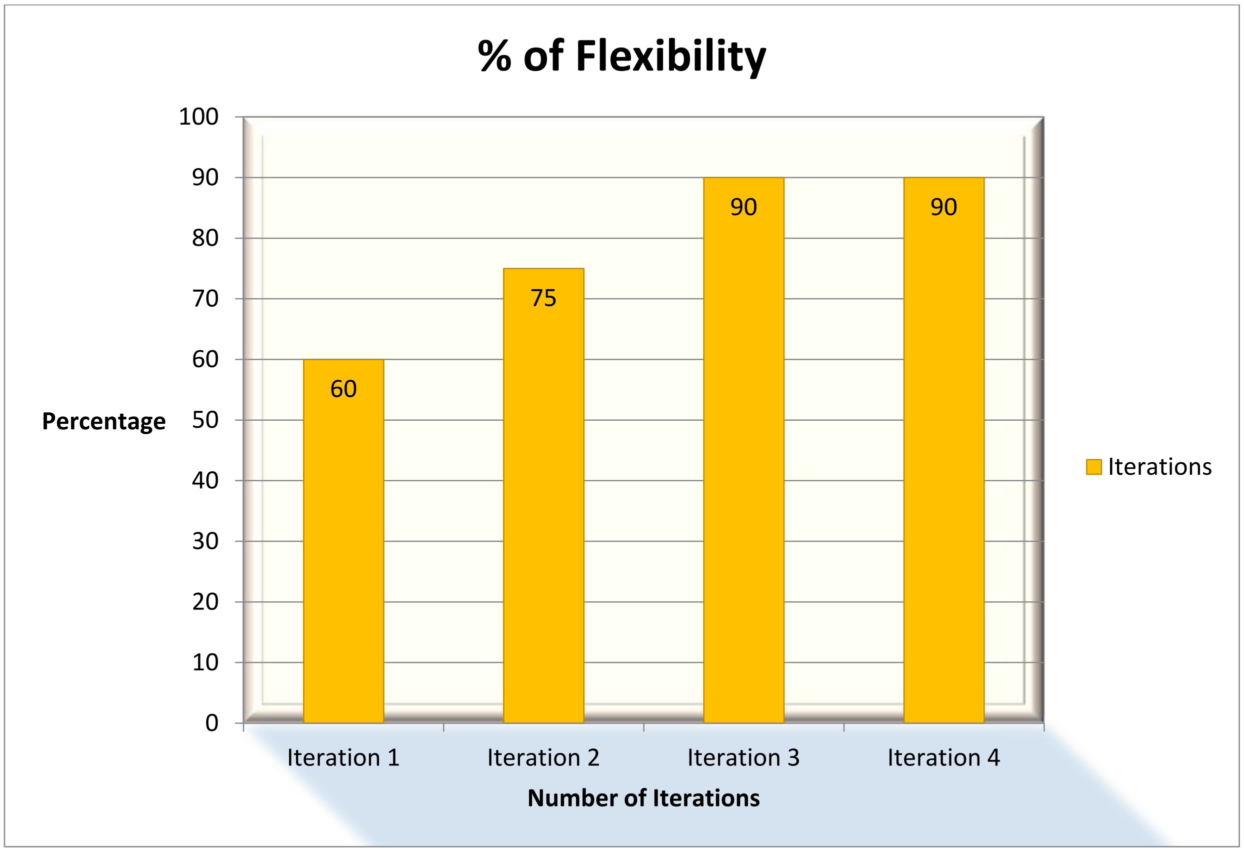

Figure 1 : Flexibility Metric Bar Graph

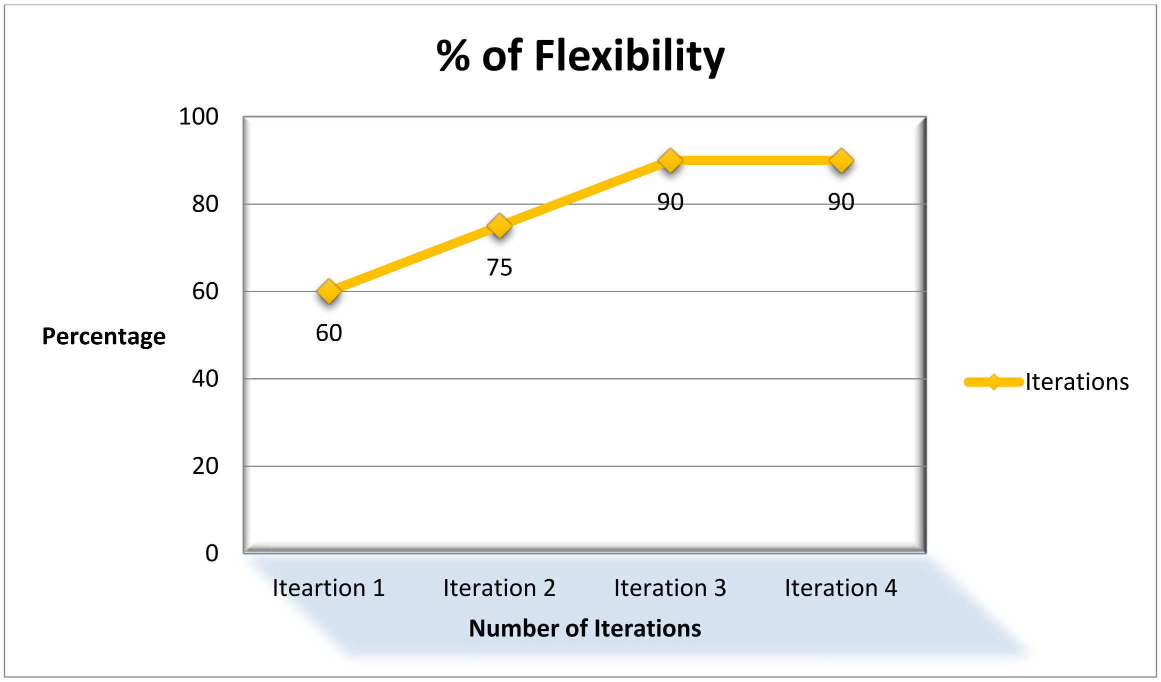

Figure 2 : Flexibility Metric Line Graph

Analysis of Results:

As the iterations increase the flexibility of the software increases and then becomes flexible. This metric shows us that in the last iteration we have improved our software design to be able to easily add new extensions.

Understandability:

Definition:

How cohesive and clear are the parts and how low is the number of connections between parts.

Meaning:

This metric gives a good idea of how the software design parts can be connected and the software can be visualized. It helps us as to how well the design is understood to be implemented.

Formula:

$$\text{\% of Understandability} = \frac{\text{Total points for each iteration}}{10} \times 100$$

0: Unclear parts and many connections among them.

10: very understandable parts with few interconnections.

Iteration Number	Description	Points	Total Points(% of Understandability)
Iteration 1	The connection between the file system is confusing.	7	70%
Iteration 2	Using CAS service is slightly difficult to understand the connections.	8	80%
Iteration 3	Understanding is easy to implement tweet additional features.	9	90%
Iteration 4	AWS and CAS are slightly confusing to implement.	8	80%

Table 2 : Understandability Metric Measurement

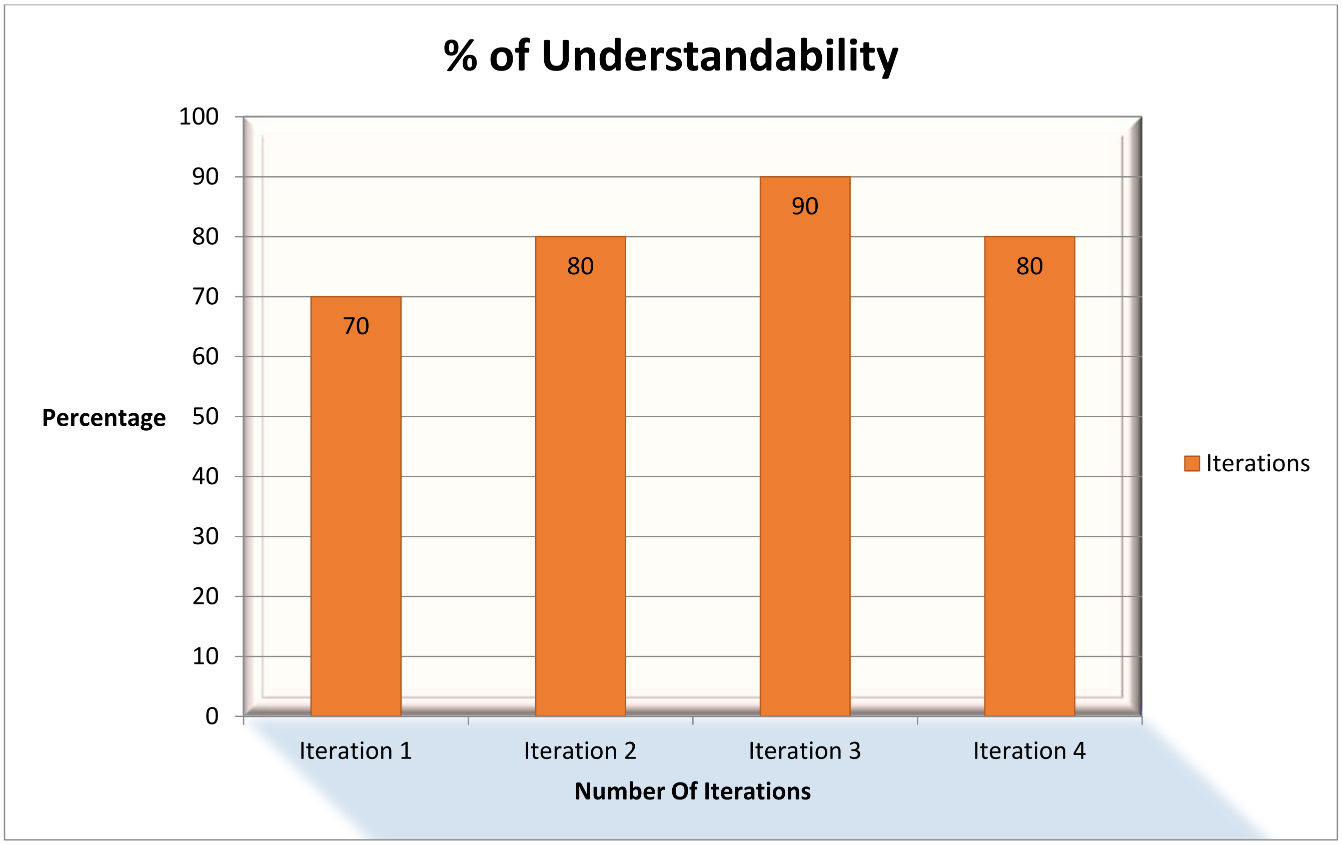

Figure 3 : Understandability Metric Bar Graph

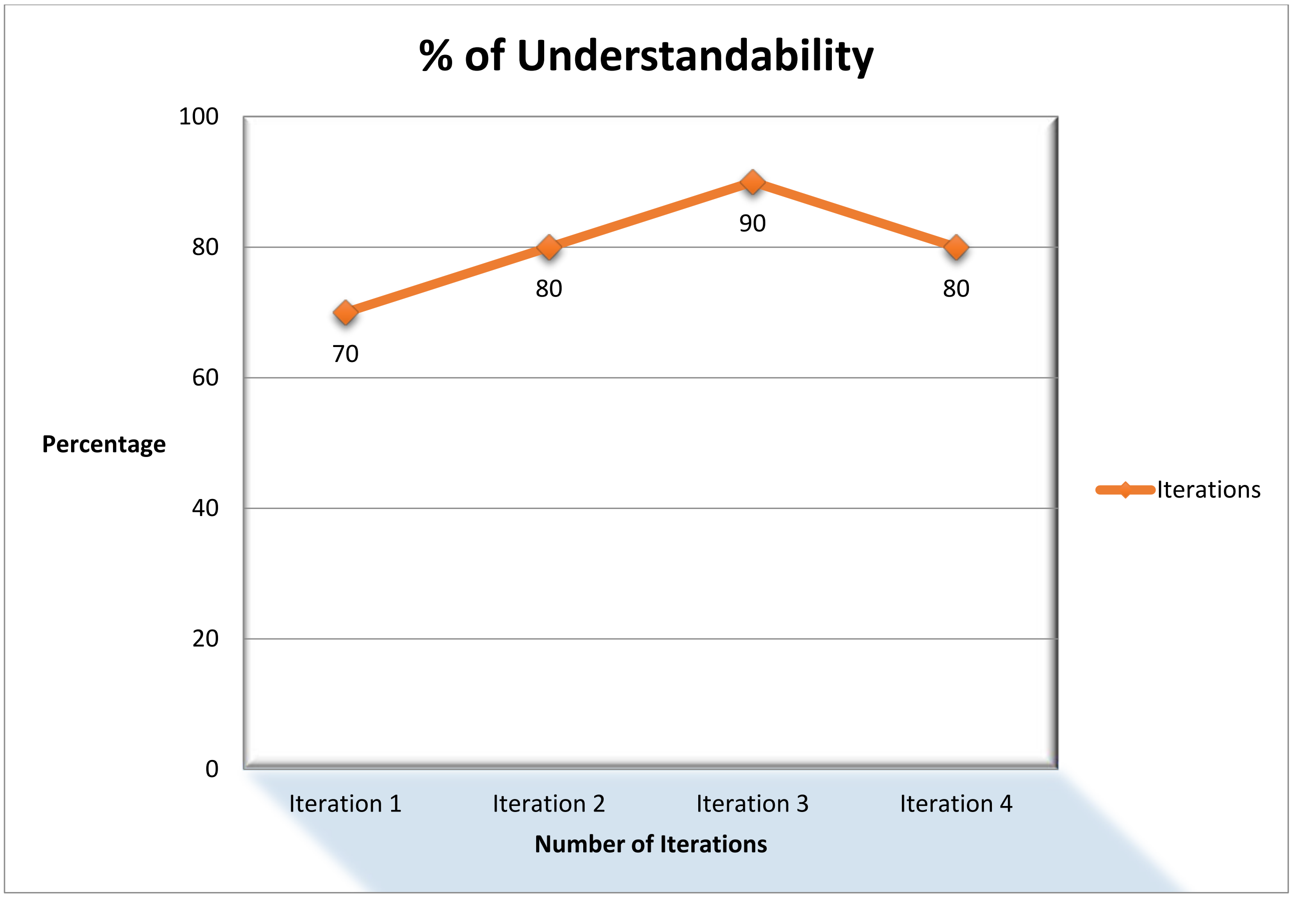

Figure 4 : Understandability Metric Line Graph

Analysis of Results:

The average of understandability is around 80% and only some parts are slightly confusing. But in general the design can be understood and implemented properly.

Sufficiency:

Definition:

Sufficiency in design metrics shows how evidently the design accommodates the requirements of the system.

Meaning:

This metric considers as to how each of the iteration in design phase satisfies all the basic requirements of being able to login from Desktop or mobile devices and tweeting from them and the tweets being shown in both devices.

Formula:

0: Unrelated to the requirements.

10: obviously accommodates every requirement.

$$\% \text{ of Sufficiency} = \frac{\text{Total points for each iteration}}{10} \times 100$$

Iteration Number	Description	Points	Total Points(% of Sufficiency)
Iteration 1	Satisfies all basic requirements.	10	100%
Iteration 2	Satisfies all basic requirements.	10	100%
Iteration 3	Satisfies all basic requirements.	10	100%
Iteration 4	Satisfies all basic requirements.	10	100%

Table 3 : Sufficiency Metric Measurement

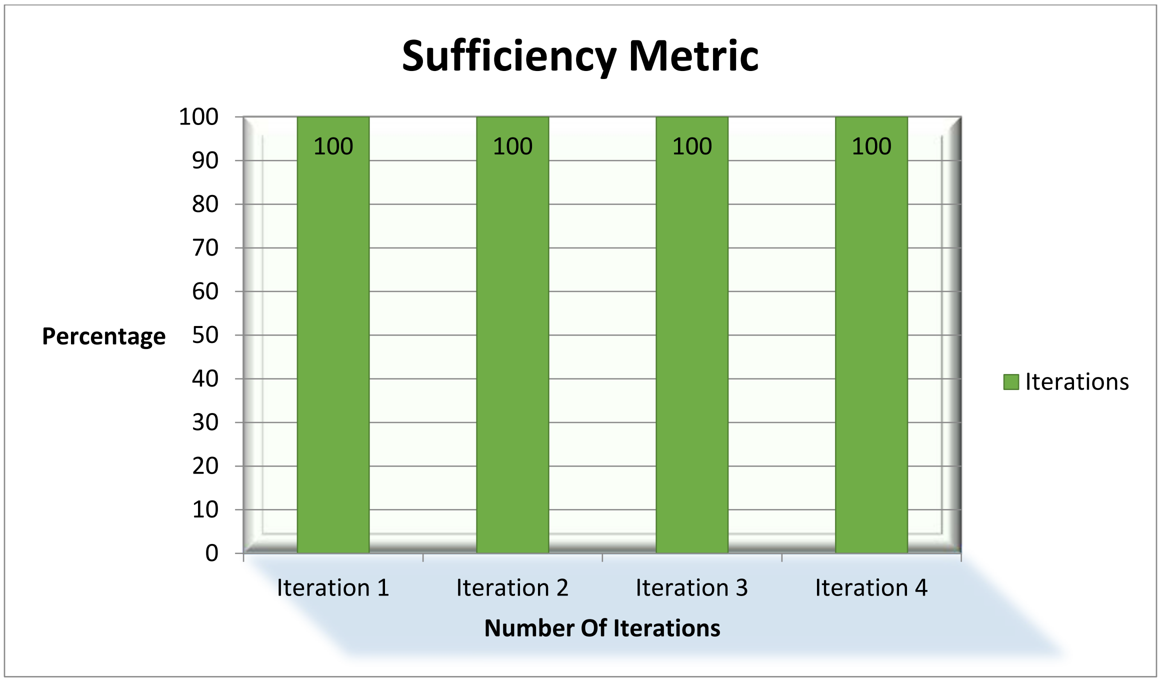

Figure 5 : Sufficiency Metric Bar Graph

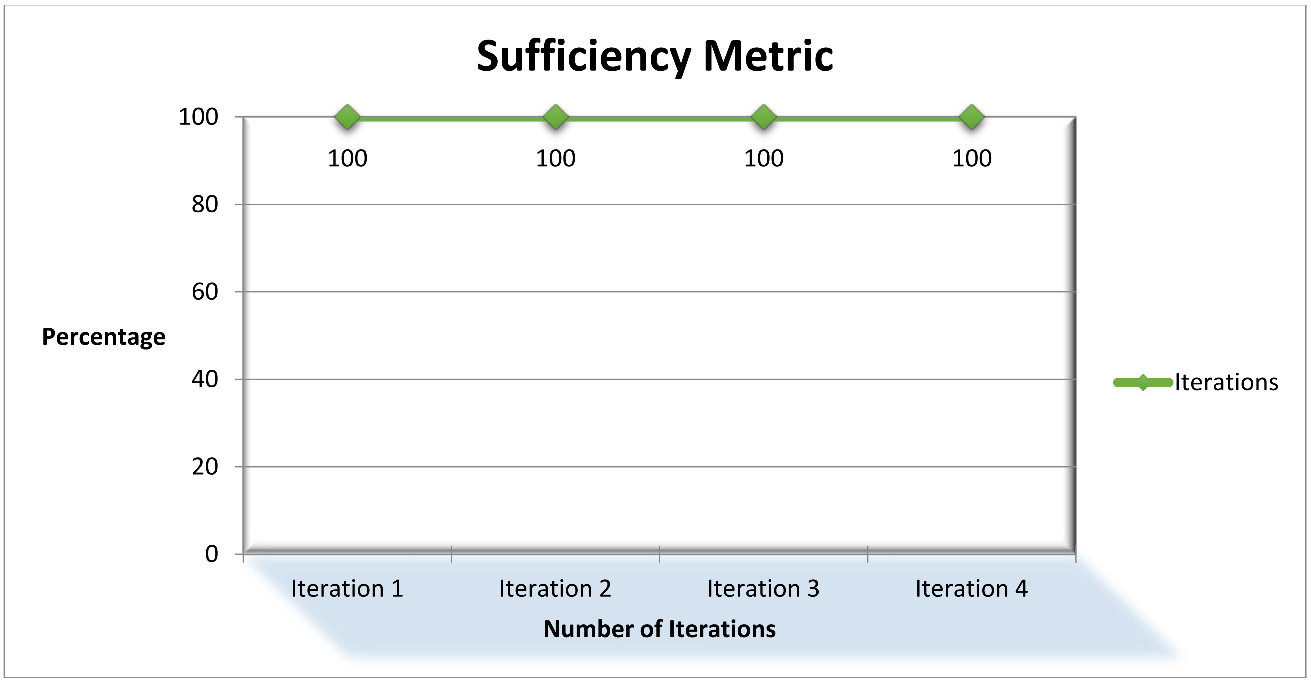

Figure 6 : Sufficiency Metric Line Graph

Analysis of Results:

This metric tells us that at each iteration how all the basic requirements of the software are satisfied. As per our design, all of our iterations satisfy the basic requirements. We are just adding more features to the software and improving them.

Security:

Definition:

Security in design metrics shows how many vulnerabilities can be expected in the design.

Meaning:

This metric tells us how in our design vulnerabilities can be expected. This would help us later on change our design as required to increase the security. If this metric measurement is high it tells us that the system built is very secure.

Formula:

0: will probably exceed the maximum allowable vulnerability limit.

10: will probably have as few vulnerabilities as can be expected.

$$\text{\% of Security} = \frac{\text{Total Points for each iteration X 100}}{\text{10 X Number of items}}$$

Iteration Number	Item Type	Points	Total Points(% of Security)
Iteration 1	1. Using file system not that secure (can be copied and read easily).	6	55%
	2. Not using https just simple authentication	5	
Iteration 2	1. Using file system not that secure (can be copied and read easily).	6	75%
	2. Using CAS Authentication	9	
Iteration 3	1. Using MySQL Database with No SQL injection	8	80%
Iteration 4	1. Deploying on AWS with CAS and Using MySQL Database with No SQL injection.	9	90%

Table 4 : Security Metric Measurement

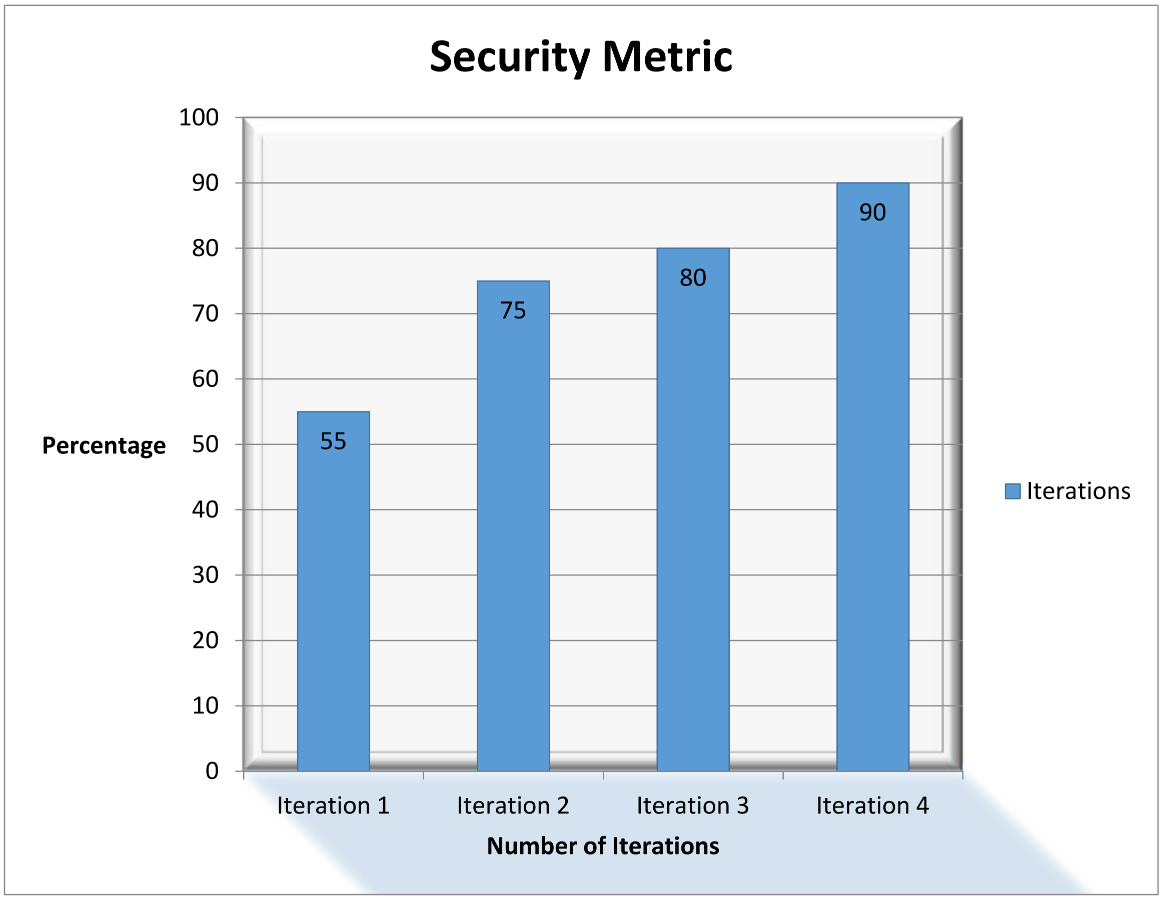

Figure 7 : Security Metric Bar Graph

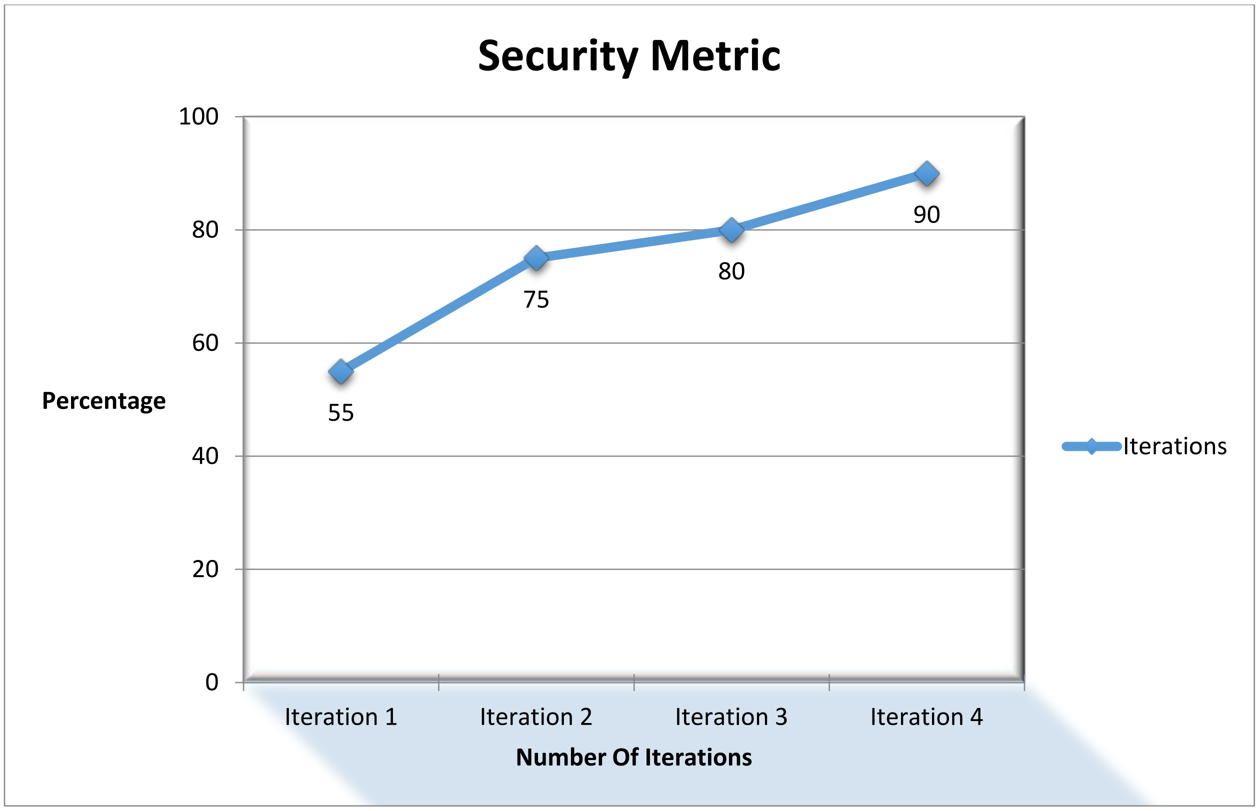

Figure 8 : Security Metric Line Graph

Analysis of Results:

As seen in the graphs and table the security increases at each iteration and we have considered the vulnerabilities as per the design at each iteration and made changes accordingly to improve it. In the last iteration, the system is very secure.

Conclusion:

Unlike the requirements phase metrics, the design phase metrics must show an increasing trend though for Understandability metric, it may not be the case as when we were following agile methodology, we were adding more features which may be slightly complicated to the basic working model. Our system shows an increasing trend and hence the software must be of better quality towards the final 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).