**Chapter 16: Software Quality Assurance**

Table of Contents

[Six Sigma 3](#_Toc63534598)

Software quality assurance controls the processes used in producing software. This ensures that the software we end up producing is of high quality, with little to no errors. SQA assures that standards, rules and procedures are established and followed throughout the software development life cycle. Quality assurance in general is used in every aspect of production, not just software.

There are three approaches to ensuring quality assurance:

* Designing systems and software with a top-down and modular approach. A top-down approach can be compared to designing the context diagram before getting into the details instead of the other way around. A modular approach means that the parts of the system should be developed independently from each other.
* Documenting software with appropriate tools
* Testing, maintaining and auditing software

We will be looking into the last two points later on.

## Six Sigma

The goal of the Six Sigma approach is to eliminate all defects in the product. It is a methodology, a philosophy and a culture that must be followed by those working on the project. It uses a top-down approach.

In Six Sigma, project leaders are called Black Belts and project members are called Green Belts. Master Black Belts are those who have worked on many projects. They are available as a resource to project teams.

The main goal of Six Sigma is to reduce the time taken for production and the number of errors. In general, there was roughly errors per million products. Six Sigma reduced this to errors per million. (What is this in reference to? Which company? You can’t just quote numbers. This sounds like a scam.) Of course, it is not possible to reduce this number to .

The steps involved in Six Sigma are:

1. **Define** – In this step, we must define exactly what the problem is, along with what opportunities we have to fix it and what the customer expects to see.
2. **Measure** – In this step, we measure values related to the problem. For example, if the system is crashing, how long does it take to recover from the crash? How long does it take before the next crash? If the user wants to do a simple task, how many different clicks does he need to do to get the task done? All of these are measures of time wasted for the user.
3. **Analyse** – We need to analyse the causes behind the problem.
4. **Improve** – We eliminate the defects and also improve upon our work to increase the productivity of the system.
5. **Control** – This is like a review step. Here, we take a look at what improvements we gained in this process and we also make sure to include what we learnt in future projects.