

Software Quality Assurance

SQA is an activity that is applied throughout the software engineering process.



SQA is...

- analysis, design, coding and testing methods and tools
- formal technical reviews that are applied during each development step
- a multi-tiered testing strategy
- control of software documentation and the changes made to it
- a procedure to assure compliance with software development standards (when applicable)
- measurement and reporting mechanisms



Software Quality Techniques

- Software Quality Assurance is a systematic program of activities designed to ensure that a system has the desired characteristics.
- It is generally better to focus on the process of software development than on the product itself.



Elements of a Software Quality Program



Software Quality Objectives



- Set explicit objectives from the external and internal characteristics.
- Without explicit goals, programmers can work to maximize characteristics different from the ones you expect them to maximize.

Explicit Quality Assurance Activity



- Quality is often perceived as a secondary goal.
- Software producing organizations must show programmers that quality is a priority.
- Making quality-assurance an independent activity make the priority clear.

Testing Strategy



- This is often the primary method for both quality assessment and improvement.
- Testing is important but it is often given too much emphasis and it is often the only method of quality assurance.

Software Engineering Guidelines



- These guidelines should control the technical character of the software as it is developed.
- Guidelines apply to all phases of development.

Informal Technical Reviews



- These reviews involve the developers examining their work before releasing it for formal review.

Formal Technical Review



- These can take the form of peer review, a customer review, an inspection, a walkthrough, or an audit.
- Formal reviews can be used as quality gates, periodic tests that determine whether the quality of the product is sufficient to justify moving to the next stage of development.

External Audits



- Auditors are brought from outside the organization and reports to whoever commissioned the audit.

Development Process



- This includes activities which affect quality but are not explicitly a quality assurance activity.
- An example is the process for risk management techniques.
- Companies which handle risks in a predictable, reliable way will reflect these methodology in their software.

Change Control Procedures



- Uncontrolled change can result in poor design, code which does not agree with the design, and inconsistencies in testing.
- Change will happen – it must be planned for from Day 1.

Measurement of Results



- Unless results of quality assurance methods can be measured, there is no way to know if the plan is working.

Prototyping



- This methodology of development can deliver realistic models of a system's key functions early in the development cycle.
- Prototyping is an excellent way to involve clients in the early verification of the specifications and design of the system, as well as a system of developer education about critical design issues.

Setting Objectives

Mind over software...



Setting Objectives



- Explicitly setting quality objectives allows programmers to know which objectives exist and those that they will be expected to follow.
- Programmers will work towards the objectives given them if they are clearly specified.

Weinberg and Schulman Study



Study Objectives



- A study by Gerald Weinberg and Edward Schulman investigated the effect on programmer performance of setting quality objectives.
 - Weinberg, G.M. and Schulman, E.L., "*Goals and performance in computer programming*", Human Factors, 16(1), pp. 70-77, 1974.

Study Description



- Five teams of programmers were given five versions of a program and each was expected to maximize a different quality objective.
- Four of the five teams successfully maximized the objective they were given.
- None of the teams did consistently well in all objectives.

Study Implications

- Programmers do what they are asked to do.
- Objectives conflict and it is generally not possible to do well in all of them.



Team Ranking on Each Objective

Objective to optimize	Minimum memory	Output readability	Program readability	Minimum statements	Minimum programming time
Minimum memory	1	4	4	2	5
Output readability	5	1	1	5	3
Program readability	3	2	2	3	4
Minimum statements	2	5	3	1	3
Minimum programming time	4	3	5	4	1



Effectiveness of SQA Techniques



Relative Effectiveness of Techniques

- The various quality assurance methods are not all equally effective.
- But just how effective are individual techniques and how much do they differ from each other?
- Do techniques work best alone or in combination?



Jones Study

Jones, C., **Programmer Productivity**,
McGraw-Hill, New York, 1986.

Study Implication

- This study by Casper Jones of defect detection methods revealed that the modal rate of detection for the examined methods doesn't rise above 65 percent for any single technique.
- The implication is that developers must combine techniques to improve the detection rate.

Defect-Detection Rates

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Personal checking of design documents	15%	35%	70%
Informal group design reviews	30%	40%	60%
Formal design inspections	35%	55%	75%
Formal code inspections	30%	60%	70%
Modeling or prototyping	35%	65%	80%

Defect-Detection Rate

Removal Step	Lowest Rate	Modal Rate	Highest Rate
Personal desk checking of code	20%	40%	60%
Unit testing (single routines)	10%	25%	50%
Function testing (related routines)	20%	35%	55%
Integration testing (complete system)	25%	45%	60%
Field testing (live data)	35%	50%	65%
Cumulative effect of complete series	93%	99%	99%

Glenford Myers Study



Relative Effectiveness of Techniques



- A study by Glenford Myers in which experienced developers were given a program to test found that none of the examined method had a statistically significant advantage over each other.
- When two methods were combined then the total number of defects found increased by a factor of 2.

Cost of Finding Defects



- As Martha Stewart would say, minimizing the cost of finding defects is a good thing...
 - Current evidence suggests that walkthrough-inspection methods initially cost more than execution-testing and find a statistically similar number of defects.
 - However, as the developers become more experienced at performing inspections they become more effective.

Humphrey Study

Humphrey, W.S., **Managing the Software Process**, Addison-Wesley, Reading, MA, 1989.



Cost of Finding Defects



- A study done by Humphrey in 1989 found that on the first release of a system, inspection found 15 percent of the errors.
- On the second release, inspection found 41 percent, and on the third release 61 percent.

The Principle of Software Quality

The General Principle of Software Quality is that improving quality reduces development costs.



The Principle of SQ



- The best way to improve productivity and quality is to reduce the time spent reworking code.
 - Reworking can take the form of changes in requirements, changes in design, or debugging.

The Principle of SQ



- Debugging takes about 50 percent of a developer's time in a naïve software development cycle.
- Improving the quality of software reduces the time spent debugging and reworking software.

The Principle of SQ



- A study done by NASA's Software Engineering Laboratory examined 50 development projects involving over 400 work-years of effort and almost 3 million lines of code supported this finding.
- Increased quality assurance was associated with decreased error rates but no change in overall development cost.