# Table of Contents

# Software Quality Assurance

## The Context

### Some comments on Quality Assurance in a Physics Context

Full text of this section in PDF.

Standard Quality Assurance methods are designed for INDUSTRIAL use, and as such may appear to be too rigid for many physicists. Our aim is to use those ideas and methods which have been successfully used in industry, and which can be applied to a large physics collaboration, avoiding methods which will squander essential resources of independence, versatility and commitment.

Our vision is that, although some of the QA suggestions will be onerous, and seem to be of limited value in the short term (which is the main concern of people developing private or prototype code), that clear cut procedures, based on the a number of well defined steps, and monitored by a reviews will have important long term benefits Furthermore, the use of various software tools should be made available to support such procedures.

| | Industry | Physics |
|---|---|---|
| **Size of a team** : Larger teams cannot rely on informal communication. | At least 5 | Typically between 1 and 3 |
| **Who are the USERS?** : In industry it is essential to ensure that the developers have understood the requirements of the users. In contrast, physicists often know their subject matter well, and do not see the need to formally describe requirements | The users are paying clients | Developers are also users |
| **Motivation of the developers** : Developers in industry NEED a formal structure to understand what is expected of them, and when it must be ready. | Developers are part of a large formal structure, and are less personally engaged in the success of the software. | Developers are highly motivated by the results of the software. |
| **Testing Procedures** : Formal and complete tests can only be designed if formal requirements have been defined. | Formal testing procedures are defined to ensure that the clients requirements have been met. The client NEEDS to see the results of these tests. | Physicists use intuition and checks of program output, which are known as "Physics Validation" |
| **Management** : Where there is no hierarchical management, no one can formally check that goals have been achieved. | Managers need to determine that members of the team are performing adequately | Physicists take individual responsibility for their work, and may not see the overall success of the collaboration as their primary goal. |

Differences between Industrial and Physics software development.

### Procedures for use by large physics collaboration for s/w quality assurance

It is clear that it is neither possible nor desirable to convert our current social structure from the one where we work either as individuals or in very small teams where each the individual has sole responsibility for all aspects of software development to one where we work in a large, rigidly organized teams slavishly following some How-To book of rules.

| | |
|---|---|
| Peer Reviews | Real benefit can be obtained from peer review, if the reviews are held in a constructive spirit, and are implemented in a lightweight manner.<br>The very fact that material is to be reviewed often means that more care is taken immediately, rather than put off to some indefinite time in the future |
| Documentation improves maintainability | Our software will be used by many people and over a long period of time. We cannot rely on the memory of individual physicists. Working methods which improve maintainability are essential.<br>The nature of OO code is such that comprehensibility improves enormously with a moderate amount of documentation (UML sequence and class diagrams in particular). |
| Testing procedures | Our software is complex, and developed by a large geographically scattered team. Each part of it needs some testing mechanism which can check that no ill effects have been introduced by a change in another part of the software.<br>These tests need to be performed automatically. |

## ATLAS Quality Assurance - What's in a name?

The verb "to control" means to "check, verify or regulate" whereas "to assure" means "to make certain or to ensure the happening" *(Oxford English Dictionary).*

The term "QUALITY CONTROL" is often used in industrial processes where an object can be accepted or rejected according to some metric.

But to reduce the rate of rejection it is necessary to :

- Install a procedure of production which will increase the probability of acceptance of the final result.
- Follow the procedure
- Verify (and certify) that the procedure has been followed.

It is this procedure which is known as "QUALITY ASSURANCE"

Quality Assurance is the appropriate term for software because:

- It happens throughout all the development of the software - from the requirements to the final code.
- Software is not mass produced.
- Software metrics are mostly intangible - and thus it is difficult to apply an acceptance test.
- By its very nature, software is rarely rejected - because it is so easily modified. (After all, that is why we call it "soft&".)

Quality Assurance implies that our software will not be judged by some kind of police action, but that we should all be a bit more careful in our software development!

*Write down what you are going to do, do what you have written, check that what you wrote/did is what you did/wrote*

## How can we Measure the Quality of Software?

Measuring the quality of software is more like judging a gymnastics competition than determining who has won a running race. It can only be done by direct comparative inspection of gymnasts, and only by an "expert". Software quality measurement is fundamentally a SUBJECTIVE process, in spite of various attempts by software engineers to devise metrics which produce numbers. (You will have seen some of these described if you have ever looked closely at a tool such as "Together")

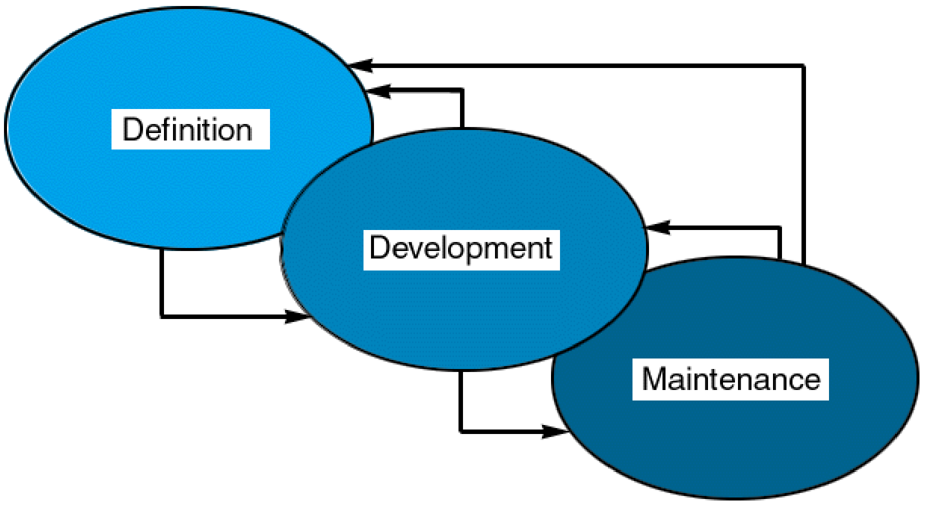
A few generalities :

| | | |
|---|---|---|
| CORRECT | Software QUALITY is based on Software REQUIREMENTS, which define what the user wants. A software which does not conform to the requirements lacks quality. | Verify your requirements, and write a suite of TESTS. The tests should be executed after every modification of the code. |
| RELIABLE | The software must get the right results ALL THE TIME, and it must be ROBUST, which means that if it gets silly input it must handle it in a sensible way. (Example - if a program is expecting a number and the user types a non numeric character, the software should be able to detect the error, and react "gracefully") | Test suites, and also cross reading of code can help with this. Always check the validity of user input. Follow coding standards. |
| EFFICIENT | The results should be obtained making the best use possible of the hardware. | It can be difficult to know what "best" means. But OPTIMIZATION should come after CORRECTNESS! Design reviews can help, and software can be checked by a tool, for such things as memory leaks. |
| USABLE | If no one can understand how to use it then it is no good | Write some user documentation, keep it up to date. Have it checked by someone who was not the author of the software. |
| FLEXIBLE | If it is impossible to make modifications then the software will soon be useless. | Write modular code. Comment lavishly. Follow coding standards. Have your code reviewed. |

## The Software Development Process

- Many models have been proposed by software engineers to try to formalize the process of producing good software.
- They are all iterative. At every step one should be able to return and make modifications.
- One of the fundamental ideas - as with all quality assurance procedures - is the detection of possible problems as early in the process as possible.

### Very Simplified Model of the Engineering Process



| | | |
|---|---|---|
| **DEFINITION** is the phase where one defines what functions the | **DEVELOPMENT** translates the definition into an architecture, then | The **MAINTENANCE** phase includes correction and |

| | | |
|---|---|---|
| software should have, what performance is required, what are the constraints, and how the software will be validated | goes into coding, and testing that the definition requirements are satisfied | enhancement to the software, and if necessary its integration into a new environment |

If we looked into each of these phases in detail there would be further iterative processes.

*A general rule of development is: the more detailed and correct your definition is, the less time take development and maintenance.*

---

Originally taken from the ATLAS static web pages http://cern.ch/atlas-computing/projects/qa/aims.php, http://cern.ch/atlas-computing/projects/qa/assurance.php, http://cern.ch/atlas-computing/projects/qa/metrics.php and http://cern.ch/atlas-computing/projects/qa/development.php.

**Major updates**:
-- SolveigAlbrand - 11 Nov 2005 -- SteveLloyd - 16 Jun 2008

%RESPONSIBLE% SolveigAlbrand
%REVIEW% TraudlKozanecki - 11 Nov 2005

---

This topic: AtlasArchive > SoftwareQualityAssurance
Topic revision: r4 - 2010-07-13 - SolveigAlbrand