How To Avoid Software Quality

by Keith Woodward

This article first appeared in

SQM

SQM issue 3

We all know why software quality should be avoided. It pleases user management if initial development costs are as low as possible. It pleases software developers if there is a large maintenance work load to keep programmers fully employed and the department well staffed.

If you wish to avoid maintenance programming, implement the system as quickly as possible and leave before the faults surface. Everyone knows that worthwhile developments will only gain acceptance from management if they are cheap and quick.

Above all one must avoid 'Paralysis by Analysis'. There is more satis faction in going directly to pro gramming which is much more fun. Users are pleased if they are NOT asked to confirm what they want before the work starts. They like an excuse to change their mind later. That is when they will insist on all the unnecessary complications which bedevil the existing system. Bothering people with questions at the start of a project can be irritating and may alert opposition that might otherwise remain in the dark. Start work before anyone asks awkward ques tions. The sequence is

1. Write the programs
2. Write the specifications
3. Test the System
4. Decide what the requirements should have been in the first place.
5. Decide whether it was worth all the expense and trouble.

Therefore, the first rule is to forget the nonsense about feasibility, analysis etc. The second rule is "Always underestimate!".

THE DEVELOPMENT CYCLE

A real software project starts with a low estimate reduced further by management and reduced even further by lack of sufficient staff with the right skills. The first Phase is known as

Good God we're never going to finish.

Realisation of this truth results in a heavy transfer of staff to the project. Many will spend much of their time in familiarisation. The staffing will not only exceed the original estimate but even the most comprehensive and realistic of estimates. The second phase is known as

Why are we spending so much money?

Vain attempts to cut back are frus trated by users producing fresh requests. Then an attempt is made to terminate the project. This re sults in the cutting of the 20% of requirements which produce 80% of benefits and eliminating any system testing, user training or documentation. This phase is known as

What can we cut out?

The unfulfilled needs of users and continuous bugs produce a period of quick fixes fading into a high level of support and maintenance which declines only with declin ing use of the system. This phase is called

When will we ever be rid of this?

Disillusion prevails and the system falls into disrepute. Other meth ods are substituted The final phase, which lasts for ever since nobody will admit to it is

Oh! We don't use that any more.

This is a successful development. The users are happy as nobody is made redundant and they can carry on much as before except that they will now claim to be computer lit erate. The developers are happy as they have gained a lot of work and experience.

PROGRAMMING

Programmers obtain great satisfaction from trying out new ideas and finding clever solutions to simple problems. This provides lots of fun for future generations in sorting the unnecessary complexity created. An eccentric choice of language can be very rewarding (e.g. A customised database in BASIC or least squares calculations in COBOL).

Avoid wasting time by designing a program in advance, or introducing structure or commented code unless this can be done In such an obscure way as to create admiration for the cleverness of the programmer. Work should never be checked by another person. Not only does it waste time but it can be very disturbing to have someone looking over your shoulder. Two people conversant with the same thing is unnecessary duplication and damages everyone's job security. There is, however, no harm in giving people tasks for which they are totally untrained and inexperienced. This is called learning on the job.

PROCEDURES AND STANDARDS

Procedures should be introduced but ignored unless it is convenient. Not only must they be longwinded and difficult to understand but kept concealed. Their existence can then be revealed to clinch an argument. New unrecorded procedures can supersede those which are recorded. Forms can be designed to prevent any kind of communication.

In this way procedures can be used to prevent unnecessary external interference and ensure that bor ing work is never done. Procedures and standards can help to remove individual responsibility for qual ity if they are introduced without individual acceptance. They should, therefore, be Imposed with all feedback discouraged.

TESTING

Management believe that one should get things right first time. They should not be disabused of this notion. If you must test, do it superficially.

Working out comprehensive tests for each major path In the system will not be appreciated. Those tests that are carried out should not be saved in a way that allows their repetition (as this wastes space) or documented (as this wastes time). If possible ensure that the system can only be run In a live environment on grounds of security. Being tied to a fixed release date minimises testing.

MAINTENANCE AND SUPPORT

This is important in occupying staff while there is insufficient develop ment work. When there is devel opment, emergencies may be settled by a quick fix which should ensure that even more work is generated later. Do not review the system as someone may discover that it is not cost effective. For similar reasons do not keep a rec ord of maintenance as this provides evidence. If the hardware changes, rewrite the system by all means but do not re-design be cause this may eliminate all those interesting routines in tended to overcome the restrictions of a pre vious machine generation.

STAFFING

The prime aim of recruitment is to enhance the salary and position of the recruiter. This may be under mined if staff stay too long and gain too much knowledge. It is better therefore to recruit on the basis of some obscure but tempo rarily valuable piece of expertise rather than long term experience and proven ability. This enables salary levels to be in creased for new recruits but not for existing staff. Do not train. Make sure people with related knowledge are kept separate. The whole concept will be undermined if staff attempt to apply their own professional stan dards. A high turnover is the key.

HOW TO AVOID BEING FOUND OUT

As stated above, all evidence of failures should be destroyed, in cluding records of software changes and the introduction of new ver sions. All maintenance is budgeted as a single overhead so that costs for individual systems are untrace able and there is a means of laying off over-budget development. The main purpose of Quality Assurance is to provide superficial cosmetics while steering clients, auditors and assessors away from what is really happening. Professional assessees maybe employed to memorise pro cedures and lie convincingly. If in doubt blame the operators, the users, the hardware and other people’s software. The chances are, you will be right.

URL for this page: http://www.avnet.co.uk/tesseract/QiC/articles/Woodward/3.html