Andrew K. Johnston
Questa Computing Ltd
"Coppertrees"
Forest Road
Effingham
LEATHERHEAD
KT24 5HE

01483 283408
07940 538713
andrewj@andrewj.com
Fax 07946 689586
www.andrewj.com

Muzzling the Alligators - A Pragmatic Approach to Quality

A paper presented to the "Managing Software Quality in the 1990s" conference, when I was Information Services Quality Manager at Eurotunnel Plc.

1. A Conflict of Advice

The newcomer to software quality management is faced with a wealth of advice from all corners and in many formats. Analysis, however, usually reduces the brew quite rapidly to two main components: a lengthy stream of ideas for ways in which to change the software production process to rapidly and easily improve quality; and, on the other hand, the concerns of experienced practical software builders that such changes will be unpopular, impractical and ineffective.

While some of these latter concerns can easily (and often correctly) be ascribed to laziness or good old-fashioned Ludd-ism on the developers' part, it is nonetheless true that many of the purveyors of quality have vested interests, or at least axes to grind, and there is a minefield of inappropriate solutions.

Eurotunnel, with its very public commercial and time pressures, provides an interesting case study. The main finding is that to improve quality, one must recognise the real problems which constrain and motivate the developers, and offer practical aid in order to gain acceptance of the quality initiative. Once over this hurdle a slow, practical progression is the only way to keep things improving - too great a change or too academic a basis will alienate the developers, and an evident willingness for the QA staff to "get their hands dirty" will not only improve relationships, but will uncover problems hidden to more conventional methods.

2. Matching Problem and Solution

2.1. Following the Paper Trail

The hapless Quality Manager turns to books, courses and consultants for help. In his enthusiasm to move forward, he may well try to adopt an existing quality system, complete with development method. However, such offerings may have a variety of drawbacks:

- ✓ **The paper trail**. Too many quality systems and development methods centre around an endless stream of forms and documents. Since it is easier to prescribe format rather than content, that is what tends to happen. Worse still, many formal quality systems verify only the existence of the paper and the project team's assurance that all is correct, and there is no attempt to independently verify the content. Many projects drown under shelf-feet of expensive but unnecessary and, ultimately, inadequate documentation.
- ✓ **The infantry approach**. These failures can occur because most formal development methods are reminiscent of an army advancing, one step at a time, into the Enemy's machine gun fire. The attempts to "de-skill" software development entirely miss the point that it is essentially a skilled activity, which will fail if the talent is constrained to an inappropriate lowest common denominator. Obviously, even commandos and special agents require control, but it must be relevant and more sophisticated.
- ✓ **Lack of real support for design, build and test**. Commercial methods, training and standards have failed to keep pace with developing software architectures. Development organisations working with relatively new tools (for example fourth-generation languages and relational databases!) may find that the commercial publications lag ten years behind the actual tools and techniques they are using, and have two alternatives: do without, or grow their own.
- ✓ **Optimism about developer motivation, comprehension and skill**. The average developer is not terribly interested in quality matters. Pride and a keenness to get the job done mean that anything which demonstrably improves the quality or productivity for the same effort is acceptable, anything which involves much extra work probably isn't. Good analytical, programming or project management skills are rarely an indicator of facility in design or with written language. Documents on quality often remain unread or wrongly interpreted.
- ✓ **Expense.** Some of the these elements are very expensive. While it is true that doing nothing may be even more expensive, making the wrong choice can have a large direct as well as indirect cost, and it is also impossible to put all the elements in place together. In the longer term, it may well be true that "Quality costs nothing", but that does not take cashflow into account.

This is not to say that all quality initiatives are worthless. Some are of great value, but the steps must be matched to the real problems and the readiness of the environment in order to provide a visible benefit.

2.2. A Multitude of Problems

The developer will typically respond to a quality initiative "I haven't got time!". His own immediate problems frequently manifest themselves as a shortage of time, money or both, and even modest improvements in standards and procedures can fall foul of them. Help must be practical, an answer to the question "What do I actually **do**?".

Developers within Eurotunnel have faced a number of problems which the books totally fail to mention, and which even carefully crafted standards and procedures may not address. Some are rare, but others are all too common:

- ✓ **Missing Users**! Eurotunnel is that relative rarity, a new business. Inevitably, it has grown top-down, with the real users of the systems only now being recruited. Thus the systems have been defined in advance of the business, often by hard-pressed senior managers lacking time and the user perspective. This leads naturally to...

- ✓ **Unstable Requirements**. The requirements have changed and kept on changing. Typically, rather than actual change there has been an unwillingness to commit to a decision, followed by a growth of unexpected detail when the decision is at last taken and understood. The problems are exacerbated by the complex relationship between Eurotunnel and its main contractor, TML, and by the number of other (governmental and financial) bodies who have an interest in the decisions. With the best of intentions, change control procedures are entirely ineffective until a certain critical stability is attained.
- ✓ **Geographical, Linguistic, and Cultural Diversity**. Eurotunnel is an Anglo-French company, with software development on at least four separate sites, and users on at least as many more. A policy (occasionally inappropriate) that all major applications should be capable of operation in at least the two main languages causes major problems. The linguistic diversity extends to the development team, and documents may change language mid-document.
- ✓ **Pre-Standard Systems**. Standards and toolset evolve, but leave untouched a growing raft of old systems which pre-date the improvements. It is unrealistic to try to update these systems, but they typically consume more than their fair share of maintenance resources, and the different practices embodied in them may be at odds with later moves to a more advanced and coherent architecture.
- ✓ **Non-Standard Architectures**. Other systems may not fit at all. Functional fit, price, architecture and quality compete when a solution is chosen. Unfortunately, architecture and quality don't always win, particularly when one is searching for bilingual packages to meet specialist needs. Eurotunnel's standards and procedures may just not cover the chosen solution.
- ✓ **Emerging Technology**. The opposite of the old system, and a special case of the non-standard architecture is the case where new tools are adopted very early in order to address particular technical problems, but without the supporting standards and knowledge of the pitfalls required to control such usage.
- ✓ **Complex Supplier Chains and Diverse Quality Approaches**. These problems are bad enough when they affect an internal development. They are exacerbated dramatically in an external development. Naturally, there are problems of formalising and expediting communications, and problems of unstable requirements sour the relationship. The supplier may have a different quality system (or none at all), and may not be willing to improve matters. Worse still is when the supplier depends heavily on a third party (the vendor of the toolset or the "package"), and is unwilling or unable to exert control on the quality of some of the basic elements of the system.
- ✓ **The Mythical Package**. The belief in a "package" solution is convenient for supplier and client alike, with the client believing that the use of a package will guarantee quality and shorten timescales, while the supplier can argue against changes to the functionality or quality, and can retain rights even when most of the source code is re-written. In reality, these approaches are only valid for a small, horizontal application such as a word-processor. The fiction of the package serves only to confuse and obstruct when a larger vertical application is under development. In this case, it may be more appropriate to consider the supplier as producing a custom vehicle from a kit of parts.

2.3. The Way Forward

Pessimistic as this may sound, there are techniques which can be employed in such an environment to good effect. Solutions must be targeted at the developers' problems if they are to gain acceptance, but within an overall framework which avoids wasting effort on piecemeal improvements. The framework must act as a guide and a reference structure, rather than a constraint on the introduction of new elements. Thus the adoption of a structured method and the creation of a Quality Manual, probably the key elements of the framework, must be seen as the construction of a scaffold to support the construction of the "real" tools, not as ends in themselves, and certainly not as a structure from which to hang the Quality Manager.

3. Some Successes

The development of Eurotunnel's quality system followed very much this pattern. An early decision to adopt Oracle's relational database and 4GL on large minicomputers as the strategic development environment suggested the adoption of Oracle's CASE*Method as the core structured method. Early attempts to build this up into a special "Eurotunnel Development Method" failed because the aims of the various parties were not sufficiently close - internal and external initiatives were meeting the same problems described above.

The CASE*Method is far from perfect: it lacks any real support for the procurement process (and the customisation of package solutions), the production of documentation and software testing. While Oracle provide good training in Analysis and some aspects of Design, the later stages are very much at the developer's discretion, and there is no real definition of the deliverables at each stage. However, Eurotunnel decided that whatever the imperfections, rather than risk further false starts the CASE*Method would provide as good a base as any other, and the missing ingredients would be added to it. Earlier, internal initiatives concentrated on the core development documents and the design and programming standards, while external consultants were invited to build on the same foundation adding support for procurement, documentation and testing.

Obviously, the quality of any system has to match its environment. It is appropriate to expect a system for a few specialist users, or one in which the quality of data may be suspect, to achieve the same quality as a major core application handling key operational data whose quality must be assured. This concept of "fitness for purpose" expressed using a "quality profile" was introduced at an early stage to reflect this need for appropriate quality targets.

The quality system has thus evolved, itself growing and developing. As in any evolution, there have been some failures, and some growths have been limited, but there have also been some significant successes.

3.1. Technical Standards and Templates

The adoption of "standards" is frequently seen as some sort of constraint on the creativity of the developer. However, if the standards can both reduce the scope for wasted development and ease the solution of difficult problems they will, gradually, be adopted. Adequate technical sophistication is important - if there are just a few apparently random or arbitrary rules they will tend to be ignored, but a comprehensive and coherent set of directives makes more sense. Against this must be set the importance of clarity - all developers are not equal, and too complex a rule set will be difficult for the less able (and may be seen as a burden on the more productive). Comprehensive and regular explanation and training is essential.

The production, to the standards, of both development documents and programs can be considerably improved (in both speed and quality) by the use of templates which provide a standards-compliant starting-point. Developers are grateful not to have to continually re-invent the difficult first steps, and see compliance as a convenient by-product. With a little ingenuity, the more repetitive parts of the production process can be automated, but by making the templates specific to the requirements and standards of the company, there is none of the rework cost often associated with traditional "program generators".

Eurotunnel's standards, for example, embody a series of fairly complex techniques which allow the production of genuinely bilingual Oracle forms. Without templates and specially-developed tools, these would be ignored or abused, whereas in most cases developers now find the extension of the monolingual system a fairly straightforward process of translation.

There is a regular life-cycle for Standards and Templates: **resistance**, **ignorance** (in which the new standards are ignored), **grudging acceptance**, and finally **enthusiasm**. Moving through the life-cycle requires considerable effort from the QA group, but it is rewarded, if successful, by much earlier enthusiasm for the next endeavour. Eurotunnel has now reached the point where developers hound the QA group chasing standards for new areas of work.

3.2. Procurement Services

Eurotunnel decided that given the enormity of the task of establishing all the necessary information systems, coupled with the uncertainty of requirements, it would be a good idea where possible to purchase an existing "package" solution and use that as a basis for further development. In order to ease the procurement process, an external consultancy was contracted to help develop a series of procurement guidelines and procedures, supplemented by template Invitations to Tender and contracts. Since the administration of these required good procedural understanding and an overview of the requirements outside the particular project areas, these tasks were given to the QA group. This job of "procurement control" has grown, with the result that the QA group now takes a key mediating role in contract negotiations, attempting to marry the requirements of the project, the information services management and the Eurotunnel procurement officers.

Overall, this has been a considerable success, with huge improvements in procurement efficiency. The overall view afforded from such a vantage point has on more than one occasion protected Eurotunnel from exposure to disadvantageous terms, and project managers have been very grateful for the reduced effort on their part required in the contracting phase. Formalised communications embodied in the contract and the procurement procedures reduce the risk of the contract going astray, but it is now acknowledged that these need to be policed more strongly to ensure their adoption even when time pressures act against them.

3.3. Email Communications

With four development sites, and users in several more, electronic communications have been essential. By avoiding procedures based on paper forms, but instead making gradually more sophisticated use of electronic mail, with shared filing systems and document templates, we have managed to keep communication efficient and effective as the business and the Information Services group have grown dramatically.

3.4. Reviews, Inspections and Audits

There appear to be two schools of thought on the role of the QA group - interventionist and non-interventionist. Eurotunnel's group is strongly in the interventionist school. On the basis that the mere existence of a program or a document in no way ensures its content, we have established a structure in which every document undergoes peer or QA-group review. Similarly, all source code is given a "safety check" before use against the live databases, and a sample is checked more rigorously as part of any QA audit. These measures have proven very successful, and have regularly alerted Information Services management to areas in which a particular project is weak. Existing suppliers are now being subjected to the same quality audit process, and although sometimes surprised at the results, most of them seem to regard any criticism as constructive.

The quality audit program has also been extended to prospective suppliers, as a now-regular part of the procurement cycle. Perhaps unsurprisingly, some formally-qualified suppliers (those with, or near to ISO 9000 certification) have revealed under questioning surprising gaps in their quality procedures, whereas, happily, others have satisfied us that without formal certification they have a deep enough understanding of the issues to minimise the risks to successful project completion. In one case the results of the audit were sufficient to change the procurement decision, and in many of the others the project team has been alerted early to risks which in the normal scheme of development might not have been revealed until the project was much further under way.

3.5. Centres of Technical Expertise

Eurotunnel has tried, quite successfully, to take advantage of several new tools and techniques, for example the newer versions of the Oracle database and SQL*Forms 4GL. The QA group has been keen to get involved early in the introduction of these tools, so that the standards can develop in parallel with the early work, and so that there is a central repository of knowledge about these tools. Developers who see their work being turned directly into standards and templates are keen to share their ideas, and, in turn, the QA group can more effectively support their customers (the developers) by understanding the technical issues.

3.6. Testing

Software testing is an area where the theoretical treatment is considerably adrift of the real problems. Books consistently consider only very simple cases, where it is possible to draw, for example, a decision table for all states of the system. The real world is quite different - imagine trying to draw a decision table for the accounting system! Public courses on testing are little better: they typically try to be all things to all men, and again concentrate on over-simplified cases. The purveyors of such courses in the UK seem to have only passing familiarity with databases, and their recommended tools and techniques don't apply to 4GLs.

Having discovered this sorry state of affairs, and aware that our testing was rather rudimentary, Eurotunnel commissioned a group of external consultants to work with us to develop more appropriate test standards. These are slanted directly towards our two main cases: internal development in SQL*Forms, and external developments based on existing software. In the cases of internal developments, a high reliance is placed on structural unit test techniques, and inspections play a key part as a proven, efficient way of finding errors.

Since one of the main problems is the lack of a common basis of understanding about testing, we have also developed a series of training courses for developers, testers, IS managers and even users. These have been a resounding success, and are sufficiently unusual, with their pragmatic 4GL bias, that we have been able to run them commercially.

However, having the right techniques and skills in place has not, of its own accord, solved some of the underlying problems with testing. The structured testing techniques are much more efficient at finding errors - their aim - but the volume of errors found in substantially complete systems may be so large as to be demoralising, and selling the message that finding and removing errors adds value to the system may prove difficult. Also, despite evidence that in recognised high-quality software environments testing is the dominant part of the software cost (Microsoft, for example, employs roughly five testers for each programmer) there is still resistance in commercial information services environments to accept testing budgets as high even as 30% of total, although that may be necessary to meet the target quality of the system.

Finally, while we have attempted to introduce automated test tools they have met with considerable resistance, and it is now becoming apparent that developers prefer manual testing techniques, finding them easier to apply until the software has attained reasonable stability.

The jury is still out on whether Eurotunnel's testing will ultimately be adequate, but the standards and training initiatives have certainly moved the work from a very raw state into a much more mature one, where the problems are at least understood.

3.7. Change Control

Eurotunnel, by its nature as a growing, new, business suffers a very high rate of change in requirements. The adoption of prototyping and iterative approaches to capture these requirements means that software changes just as much as the documentation.

It has proven relatively easy to enforce *version* control: documents are registered centrally, and changes are well marked; while a system of controlled baselines and separate programmer workspaces, in conjunction with a version control tool for the source code mean that it is fairly easy to monitor changes to individual software items.

What has not proven so easy is *configuration* control: it is still not always possible to trace changes in software back to documentary changes, and with internal communications, documentation and source code all in different environments, it is not always easy to build up a picture of all the elements of a system.

It is possible to suggest a life-cycle for the ability to handle changes, and progress from one state to the next demands the achievement of a certain critical stability, as well as the attainment of certain techniques and discipline. Initially many projects are **chaotic**, with few rules and no real ability to even detect changes. Sooner or later, they should progress to **ordered** in which rules are applied and changes can be monitored. However, only when the overall configuration is clear and changes can be planned ahead of an urgent requirement can the project truly be considered **controlled**.

Within Eurotunnel, very few projects are still chaotic, but, equally, few have yet attained the stability for controlled status and it is clear that such control cannot successfully be applied artificially before that stability exists.

4. The Role of the QA Group

It can be clearly seen that in order to contribute actively, rather than passively, to the improvement of software quality the QA group has to adopt a more pro-active stance than is traditional. Within Eurotunnel, the function is much more that of a "development support group", and requires a blend of approaches:

- ✓ **Guru** The QA role requires substantial technical knowledge. To effectively support developers, to create effective standards and to administer any central resource of technical information the environment and tools must be well understood.
- ✓ **Policeman** It is a mistake to believe that even the best rules will be followed or the best tools used of their own accord. Time pressures, developers' inertia and their desire for creativity will all at one time or another work against the standard approach. It is necessary to detect such departures, evaluate them and their reasons, and decide whether they can be allowed, whether they can be rectified amicably, or whether an issue needs to be escalated for a management decision.
- ✓ **Priest** If the QA group is seen as knowledgeable and fair, it will inevitably become an early port of call when differences arise between or within the project teams. It is often possible for the QA function to defuse the situation and negotiate a reasonable agreement, independent of the differing objectives.
- ✓ **Smith** The QA group may be responsible for some of the development, test or documentation tools. In this case, the ability to tailor tools to the specific requirements of the developers is a considerable advantage.
- ✓ **Auditor** It is not possible to assume that just because a document or a piece of software exists, the target quality has been attained. Since quality is the sum of product and process, the QA role has to be able to investigate both, knowing the right questions to ask and being able to evaluate evidence to verify the answers.
- ✓ **Negotiator** Inevitable, with the multiple role of assessor of quality, recommender of improvements and participant in the procurement process, negotiating skills are most important. Where possible, the QA role must be able to relate to differing viewpoints, and to find a course of action which will satisfy the requirements of the organisation, while still allowing the individual participants to attain their reasonable objectives.

There is no obvious single route to achieving these abilities. A background in various aspects of software development is essential, and it is obviously a good idea for the group to include a number of individuals with different backgrounds and talents. The "poacher turned gamekeeper" is a key member, since only by knowing and understanding the pressures on the developer will the QA role be able to help deflect and handle them.

5. The First Steps Forward

There are two common strategies for developments of any kind. The first is to try to provide the maximum product (or improvement) for a fixed cost. This tends to lead to an overall plan in which a number of higher-cost or higher-risk elements have been included. Frequently, these elements will then get a disproportionate share of the effort.

The alternative strategy is to look at all the possible components, and to try to evaluate their cost, risk and benefit. Each can then be assigned a notional "value":

$$value = (benefit) / (cost * risk)$$

- it is then straightforward to plan to implement the highest value elements first, and the lower value elements later. This has the clear advantage that the users can see an earlier return on their investment, and that the development can be modified or even stopped at a later date but some clear benefit will have been achieved.

This is therefore a very good strategy for the implementation of a quality system:

✓ Firstly, recognise that the customers, the end-users of QA are the developers themselves. They will be unwilling or unable to implement any change which does not provide a clear return on their investment (of effort).

✓ Assess the problems currently faced by the developers. Are there any which admit a simple solution? If so, early provision of such a solution will win friends and encourage the adoption of later ideas.

✓ Find a framework into which the various initiatives can fit. If one exists, it will be easy to understand how the different elements relate to each other. Without such a framework, management may not understand and support individual changes.

✓ Evaluate the cost, risk and benefit of each change. While a new method or expensive tool may sound very good, if it has little direct benefit, if the amount of change required to current practices is significant (which will increase the cost), or if the level of current understanding in the organisation is not high enough (so that the risk is high) then the value of the introduction will be too low. Be cynical, and ask the developers and their managers to add their assessment of benefit, cost & risk.

✓ Finally, dare to be different! What works in one organisation won't necessarily work in another. Theoretical approaches to subjects such as testing lag behind developments in the toolset, and the old methods may have to change. The received wisdom is that the QA organisation should sit apart and monitor paperwork - in practice there is much more to be done.

Improving quality is a slow, and sometimes painful process. There is no magic wand, no silver bullet. All parties must be pragmatic, and must not underestimate the time and effort required. Gradual changes carefully designed to bring defined benefits are the best approach, and must match the culture and status of the organisation. The guiding principle must be "highest benefit, simplest first".