







Case Study 1

**The Paper Trail**

The following example is of people facing a situation where the right or wrong decision is not immediately obvious. Think through the issues, the underlying ones as well as the superficial. Analyse if and where these people are getting it wrong, what they might have done better and how they might be motivated to get it right.

Programmer A has been maintaining system X since it was developed. A major upgrade is to be undertaken, and new programmers are being brought in to work with programmer A.Programmer A is discussing with his manager how best to spend the time remaining before the new team starts. There are one or two minor modifications that users have requested. These will not take long, and could be cleared up before the new team arrives, thus giving it a clean start. On the other hand, a couple of minor straightforward modifications might be just the thing to get the new programmers familiar with thesystem. Programmer A feels he should use the time to comment the existing code and bring the system documentation up to date. Being the sole maintainer, he has been a bit lax at times. His manager thinks that getting started on documenting the requirements for the major upgrade might be a better use of the time, to give the new team a flying start. The old code will eventually be ported to another language. Is there any point in updating its documentation? If there is any catching up of paperwork to be done, programmer A could update his QA backup forms – the forms that all personnel are expected to keep to show how, when and where electronic data is backed up. Programmer A defends his position of keeping his own personal records as he is the only one with access to,or ability to use, the data he has responsibility for backing up. Furthermore, he points out, the whole system will be changed when thenew team starts. The manager agrees, but points out that a QA reviewwould still put a black mark against the non-existence of this paperwork.

They debate the pros and cons. What is to be gained or lost by the timing

of each of the suggested tasks?

Case study 2

**Hidden Benefit**

As part of a job interview, Candidate A and Candidate B are given a programming task. They must make a simple modification to a piece ofsoftware. The system comprises tens of thousands of lines of code. There is insufficient time to tell them the details of what the system does. All they know is that at certain identifiable points, the code makes a 5% adjustment to a total. For the new version, the adjustment must be 7.5% under one set of conditions, 3% under another, and must stay at 5% for the rest of the time.

Candidate A's objectives are speed and the least possible modification of the source code. The points that might require modification are easily identifiable. Constants within the existing code already exist for the values 3, 5 and 7.5. Candidate A uses these, addsconditional statements only at the places where the new conditions could occur and turns in a working version that requires little new compilation.Together with this, Candidate A makes the recommendation that themodification be re-done when time allows, such that the calculations are no longer reliant on the constant values, and the testing of conditions expanded to include all possible points.

Candidate B's objective is to make the modification in a reasonable time, but with due regard to code structure. This involves removal of any use of constant declarations. The required values will beread in from an external table. Conditional statements are placed at all identified points, on the grounds that even if this modification does not require it, a future one might. Together with the working version,requiring full recompilation, Candidate B turns in a recommendation that the objects subject to these percentage shifts be better encapsulated, such that calculations are invisible in the main source code, and that the conditional statements themselves are no longer necessary.

When required to argue for their own solution against the other candidate's, A claims to have demonstrated the ability to perform a quick and workable amendment. Preventive maintenance, by way of code amendment, can easily follow when time allows. This solution gets a working version back on the clients' desk in the quickest time. Although not the best structured code, speed of delivery might give a commercial advantage.

B says that A has taken this argument too far. There might be need for another quick fix before the necessary preventive maintenance can be done. A quick-fix on top of A's solution might, for example,change the value of one of the constant declarations with catastrophic effect. B claims a solution that, although not perfect, has provided a working system in a reasonable time and one that can bear the weight of another quick-fix without disastrous results.

Try to argue the case from both sides, and see if you can make a convincing case for employing either candidate.