Experimentus
excellence in software quality management

Experimentus February 2009 e-newsletter

www.experimentus.com

NEWS ALERT!

Experimentus - First UK company to achieve TMMi Assessor Accreditation

Experimentus (www.experimentus.com), a leading UK software quality management consultancy, announces that it is the first IT solutions and services provider in the UK to be awarded accreditation by the TMMi® Foundation for its in-house developed TMMi assessment method. This prestigious endorsement confirms that Experimentus' TMMi assessment method delivers the highest professional standards in test and quality related process assessments.

The TMMi® Foundation (www.tmmifoundation.org) is a not-for-profit organisation, which has the aim to promote advanced test and quality practitioners and is dedicated to improving software test and quality related processes and practices. The Foundation owns and supports the TMMi model, provides an independent assessment scheme for assessors and lead assessors, along with the structure to accredit assessment methods.

In accordance with ISO 15504, the Foundation created TAMAR – the global standard requirements for TMMi assessment methods. Today's accreditation confirms that Experimentus' TMMi assessment method complies with TAMAR.

Geoff Thompson,
Experimentus Consultancy Director commented:

"This accreditation further illustrates our commitment to delivering a consistent and "best in class" service to our clients. It provides them with the assurance that, as a service provider, we are adopting to the assessment standards required by the Foundation. Moreover, whatever your assessment is testing and quality, be it structured or agile, the systematic and disciplined structure of the method we have in place means that we can conduct the TMMi assessment the service will be identical."

Marc Miller,
TMMi® Foundation Director responsible for Accreditation commented:

"This is a real milestone not only for Experimentus but the testing community as a whole. TMMi has real value because it is the first industry testing model to be owned and actively managed by the testing community. Accreditation is a key component of its credibility. The TMMi Foundation creates an environment where all companies, irrespective of size, can have the opportunity to validate their test process against an industry standard model and provides a roadmap for future improvement. Over time it will enable companies to compare how well they are performing against others in their industry."

Experimentus' TMMi assessment method will be independently reviewed by the Foundation every three years, as well as undergoing annual internal reviews to ensure all new industry changes and innovations are reflected in it.

Martin Adcock,
Managing Director of Experimentus commented:

"We are delighted, our goal has always been to deliver effective and valuable technology services to our customers. As a leader in software quality management, Experimentus must meet the global standard for assessment methodologies, utilising thorough, repeatable industry recognised methods."

NEWS ALERT!

Software Testing – the future?

Comment by Geoff Thompson,
Experimentus Consultancy Director

When I was set the challenge of writing an article on the future of software testing I thought, where do I start? Then I realised that I was wondering where to start. Then I found the following headlines in two well known online journals:

'BA: software failures key in T5 fiasco' Computing 8 May 2008.

'Update: lack of testing blamed for Terminal 5 fiasco.' BA executive tells MPs' Computer Weekly 9 May 2008.

In summary both articles made the same point. During questioning by MPs, Willie Walsh (BA Chief Executive) reported that construction work had not completed in time squeezing the window for pre go live testing. He acknowledged that errors had and agreed the risks and issues, and decided that the risk was acceptable to meet the opening date.

In hindsight Walsh said "If were to poke one issue I would have done differently, it is that perhaps we should have delayed the opening" and having designed six months of testing, we subsequently compromised on that." Computer Weekly 9 May 2008.

Why did it interest me? Quite simply the management of the project did exactly the right things: project management and risk management were used to assess whether slippage in earlier parts of the testing could be offset by reducing test activity and timescales would have an impact on the quality of the finished article. Obviously the risk was assessed as low and a decision was taken to reduce testing and go live on time (a decision now reported by Walsh).

What I also found interesting was that one of the papers saw software testing as the issue. Initially though there was an agger of truth in that the first few days of T5 did not run smoothly. It is also more than likely that a new version of the flight information system (FIS) software would have enabled the right decision to be made to divert the 56 aircraft to Heathrow. The test organisation had a grip on the real issues that mattered to the management team and were able to communicate those real risks maybe a different decision would have been taken, or at least the decision to go live would have been a decision based upon the right information with complete awareness of the impacts.

To me, testing's role is the provision of data to enable go live decisions to be made (I have heard this termed as Trusted Advisor and Project Guardian). Yes, we might be criticised for the decision but ultimately the decision belongs to the Sponsor and the management team, but the data used in the decision and predominately be derived from test activity and must be sufficient to enable the right decision to be made.

Given that this is a major element of our role, I suggest that we need to get this right in the future if software testing will be very dim. We are in the best position, or at least we should be, in a position to give the most accurate reporting of project progress information. Our test environments are where errors and risks are discovered and where we can give the most accurate assessment of the risks.

Project delivery is not difficult, it is simply the bringing together of competent people to do the right things at the right time. If we all work together and talk about how the system should actually work (or where it should be) then, for example, a developer can learn from a business analyst.

A project manager who just cares about meeting delivery dates and budgets, but does not investigate issues, is a fool in my book, and one that shouldn't be working in our industry. Likewise Eden V. Murphy, a test analyst, who withholds test information and may be different for each different level in the test hierarchy.

1. Success for a Sponsor may be a system that saves or makes them £5bn a year. In this case the project team involved needs to include data to reflect achievement when it goes live.
2. Success for a Test Manager may be that there is an agreed test approach, there is testing independent of the rest of the project and all high priority test cases are agreed up front and to be completed before the system goes live. Success may also be for successful test manager needs to have a protected test budget, which they will use to investigate issues, even if they are not in the test plan. Success for a Test Analyst may be today but maybe one that doesn't help point B above (building quality in), not testing it out. Perhaps a better success criteria would be:
a. Test activity defined that reducing the amount of test execution to one week a day for the test analyst, to think together and identifies up front where things may go wrong and stop them occurring. Using a pre-test review to identify issues, the test team will was about to fall down, would you write a test and then go and run it? Then check it had, raise a bug report and you a lot of money to have it repaired? Or would you do something else as well as possible to help? Talking and keep the cost, time and quality at the right levels? Translating that into a project, we know the success criteria for project manager would be: Cost and quality delivered, lack of leaps of preventing rather than detecting issues as this is the most expensive approach to delivery and to meet all these objectives.

Test managers and project managers must also reflect progress towards achieving these criteria. *But we shouldn't forget that the primary function of a project is to deliver a solution to a business issue and risk is resolved, and so by necessity, must the provision of information.*

Testing isn't provide the right information it actually adds little value and that I suggest maybe the situation we find ourselves in a lot of organisations today, we all work very hard at our profession but in one outside of it seems to understand what we do and why.

Whilst working on one client site I remember interviewing a Test Manager who was most upset that the Programme Board had asked him to stop issuing his daily report for what he described as stupid reasons. When I met and interviewed members of his programme team they said that the report was stopped because it was the Test Manager's favourite way of not doing real work. And, what was requested was that the report included relevant data (such as where the issues lay, what elements of the software was not working etc) or was configured. The Test Manager saw no need to include this data to allow them to be interpreted. Even the Test Manager cannot understand what he did wrong and therefore will be unable to change his ways to deliver because they did not listen to them.

Software testing is ever to be seen as a profession and be respected in the future then we should focus on the information we need to enable the Programme to make the right decisions.

Now I am not suggesting that I know the answer. Each person will have different ideas, but I suggest that importantly, their requirements may be very different to what they request individually when asked and what they actually need. Therefore I suggest that a group influences individual perspectives and results. If you get someone appointed to an action from each individual in a group of people, when you put them together the actions may and often changed but it is our job to identify what they are and focus our approach to testing in order to deliver these. We need to be conscious of the daily syndrome.

The data requirement is never static,, so we also need to be conscious of the changing needs for information as a project progresses.

To enable the future of software testing do we have to influence people's approaches to Test Management and this provision of information? Is it a qualification thing, or maybe standards, there is not treasures of good information on the subject but is it read? The following sections look at each of these individually and their relevancy to the issue.

Software Testing Qualifications

My chapter in my book (still to be published) is titled Software Testing Qualifications for a few years now! Firstly working with ISEB (Information Systems Examination Board) and lately internationally with the ISTQB® (International Software Testing Examinations Board). So I guess I am a little biased, but I believe that the qualification scheme in the UK is industry leading due to the value of these types of qualification, some calling for them to be banned, others suggesting that having got them you are able to walk on water and everyone in between. The reality is that a good tester or test team requires a lot of skills and knowledge combined to make them successful. In my view the qualifications account for about 10% to 15% of the knowledge required for the practical application of testing.

At ISTQB Foundation level there is a short training course (3 days) followed by an exam. However the quick learners amongst us could very quickly understand what is required and pass these exams, maybe even without ever testing anything, there is no minimum experience required to take these exams.

Intriguingly ISEB have recently revised their single Practitioner Exam, into three, The Intermediate (a mandatory common module which contains their view of the generic skills required for a tester) and two specialist modules, Test Analyst and Technical Test Analyst, each requiring a 3 day training course followed by an exam. There is a minimum entry requirement but they do not include practical test experience, more a passing of the Foundation Exam (or equivalent).

ISTQB, on the other hand, have just launched their Advanced level role-based qualifications and like ISEB, there are three of them: Test Manager, Test Analyst and Technical Test Analyst. Each of these requires candidates to have passed the Foundation exam and to have practical experience, example the role of a Test Manager in a review is very different to that of a Test Analyst with a minimum training period of 5 days each followed by an exam and the practical experience of 3 years.

So I believe qualifications have a very important part to play in the education and verification of software testers and test managers. But the reality is that the exam candidates leave with an understanding of the basic vocabulary and approach to testing but in most cases lack the practical capability and certainly no understanding of how to communicate outside of the test team.

Standards

There are a few out there, for example BS7925 – 1&2 (the British standard for Component Testing and its associated glossary) and IEEE829 for the documentation standard for software testing.

In their favour Standards are developed by a very wide working group (these can be as many as 1000 contributors) who build and review the content to ensure that it represents a consensus from right around the globe. However, by necessity they are general and some concessions have to be made to gain global agreement (in fact the revision of IEEE829 was withdrawn and reissued as a standard as the detail was not agreed and provide one solution for everyone so they are useful but they can't provide the whole answer.

Standards don't, in any example I have seen, indicate how to provide and what information is needed to communicate effectively with the programme, although they can help. For example in IEEE829, it discusses entry and exit criteria as well as closure reports looking at

Lots of organisations seem to think that by putting people in which ever role they choose, and providing templates for documentation they will get great results. This ignores the need to work as a team and to communicate, and as in most instances is a strategy doomed to failure.

I think it worthy of a stop here to review another element we need to consider. We can have the best people in the best roles with the resources, I refer to anyone involved in a software development project, but if they don't work as a team, then success is unlikely. Some business process re-engineering from the early 90's identified communications issues and, ironically, by the 2 dozens directors and managers, that we have to spend more time on software development we understand what it is we all need, and how we can deliver it.

If the test manager for Terminal 5 had provided the following information to Willie Walsh, would he or the point the decision to reduce testing occurred I wonder if the result may have been different.

Test Manager –"Mr Walsh we have reviewed the progress of testing and have identified that we can reduce testing by 2 days, reduce testing in the way suggested the following will happen:

In the first 5 days of operation:

1. Over 23,000 bags will get lost, leaving you with a bill well in excess of £4 Million to deal with passengers
2. 6,500 flights will definitely be cancelled
3. Staff will not be able to park due to the security system and so we will have to close Heathrow
4. 28 lifts will not be operating
5. You will definitely be pulled up in front of the UK Parliament to explain yourself

I recommend we delay launch for a minimum of a week so we can resolve these issues."

Would he have agreed to the same launch date? That I cannot say, but at least if he had, he would have understood the issues he was to face with the press and the public.

As you may say hindsight is a great thing, if we could all see into the future we could perhaps all determine these types of issues easily but how do we do that prior to delivery?

Planned activity against actual activity and lessons learned. Although it mentions these key elements of a test project it doesn't provide any help in providing the right detail.

They are, however, excellent process models that provide guidance on how to establish your test project.

Test Models

There are many models available for testers to review against. Most consultancies will have a Test Maturity Model of their own. Most will be ideally lead to creating a sales opportunity for them as they will actually have to help the organisation that has been reviewed against the model to improve what they do. The nearest commercial model that really does set out to help companies is the TPI® model provided by Sogeti. However, there is now a fifth in the TPI family of models, called TMap® Next which has been updated and those updates have not been reflected within TPI®.

Probably the best model available today to help answer good processes in use across test projects is the new TMMi Model. This model has been developed by the TMMi Foundation, an international organisation with an aim to establish a non commercial test process assessment model for the industry. The TMMi model takes as its starting point the Capability Maturity Model Integration (CMMI) and others including the Gelperin and Hetzel's Evolution of Testing Model (Gelperin and Hetzel), which describes the evolution of the testing process over a 40-year period, Beizer's testing model, which describes the evolution of the individual tester's thinking (Beizer), research on the TMM carried out in the EU-funded MB-TMM project and international testing standards, e.g. IEEE 829 Standard for Software Test Documentation (IEEE 829) and the British Computer Society's BS 7925/2 Standard for Software Component Testing terms used in Software Testing (BTCS).

The model is structured in the CMMI with 5 levels of maturity. At level 2 of the model projects begin to be assessed on their capability to measure progress through testing. However like the CMMI model it is not prescriptive as to what is measured just that measurement occurs, although help is provided regarding the minimum that is expected.

Tools

How to some these are the panacea, I prefer to look at tools as the slaves of the process, in most of the examples I have seen referring to information provision there does seem to have been thought put in to the data that the tool provides. The Dashboard seems to be the latest obsession, accessory or the must have tool. I have to say I have seen some good examples of these, such as the Changepoint and Compuware tools are in ODIM. In principle these provide data derived from actions within the system.

Tools can be provided from providers of testing tools, they can be good as they are customisable to your processes. For example, with Test Management tools is that the tool is implemented straight out of the box with no configuration. Testers then load test cases into the tool with no reason why they are being loaded and then they are executed. The tool then reports on the data that is held in the tool. The problem is that the data recorded is of little value to the project as it is not the right information. Many of the tools can be configured to provide the right information, I might well have also decided, like NASA did, to launch. Now I know it's a weird thing, but in reality if we can't provide information that is relevant to the project, testing will struggle then all the great work we have done to improve the processes and practices in the organisation will be wasted and the tester will be seen as an industry and will always be seen as the also ran profession in all about communication.

Sometimes, alright, I concede most of the time the information we need to provide is not what the Programme wants to hear, but it's our job to ensure the right information gets through to the right person and that they have the information they need to make decisions. We need to be prepared to be the bearers of good and bad news.

Teamwork is absolutely key to achieving project delivery in the future, and working software testing continues to deliver value. Sadly this is something that gets very little daylight in IT. We are one of all or nothing. We're not responsible but we don't really consider that they need to be taught how to manage people and how to build working teams. Good developers are promoted and asked to manage people, and it's the same in testing. How many testers ever get sent on a project management course to learn the basic project management processes? Very few, in fact it's an organisation I worked in it was considered a course that was too advanced for test resources to attend. However developers and managers aspiring to become Test Managers were allowed to attend.

I think a software delivery project is much like an orchestra, it feels very further togethering it can be beautiful, but if only one element works on their own the result will be disastrous. We need to work together, and I think the Agile approach to delivery does delivers beautiful results.

So we're just back to where I started. Good information provision comes from being part of a team who communicate regularly and who understand progress through the different lifecycle stages, all the way through to the live environment.

At this point I can hear you ponder the quote – "You cannot manage what you cannot (or do not) measure" (anon). So we must know:

- What do we need to measure and does the information and importance of the information change during the lifecycle of the project?
- We must still capture the number of test cases, tests run, tests passed/failed, but we must learn to identify how to translate this information for the different audiences we have.
- We need to capture "quality requirements" and therefore identify how to measure and report on them. They must be clear and concisely defined, easily understood, not ambiguous and easily demonstrated.
- Testers must encourage the breakdown of the silos. Why not work with the developers, share ideas and help them?

No one can define today what the future of testing is but by learning to communicate more effectively and by becoming an integrated element of the project, I believe the future of testing is assured.

NEWS ALERT!

Experimentus Team Accredited TMMi Assessors First in the World

- **Four Experimentus Consultants Accredited To Use The Experimentus TMMi Method**
- **First Software Testing Consultants In The World To Become Accredited**

The software quality management consultancy, Experimentus announces that four of its senior consultants have been accredited by the TMMi® Foundation to use the Experimentus TMMi assessment method.

The accredited Experimentus TMMi assessment team consists of the most effective methods to assess and improve software quality processes.

The prestigious endorsement confirms that Experimentus and its team of consultants adhere to the highest professional standards test and quality related process assessment.

Martin Adcock,
Experimentus Managing Director, commented:

"This award once again highlights the expertise, experience and high standards we have in place at Experimentus."

To mark this Accreditation, Experimentus is conducting a survey to identify the maturity of software testing and quality across industry sectors in line with the Test Maturity Model (TMMi). On completion, Experimentus will produce and provide participants with two valuable pieces of information:

- **A high level view of their organisation's testing maturity against the TMMi**
- **A White Paper featuring industry benchmarking data**

For more information and to carry out the short survey please go to either:
www.experimentus.com
or
http://experimentus.tmmibenchmark.sqlpmo.com/

Ten Tips to Reduce Software Development Lifecycle Costs

With the credit crunch affecting IT budgets, Experimentus highlights ten ways to reduce software development lifecycle costs within IT departments :-

1. Focus on well documented Functional Requirements
Without well documented Functional Requirements there is potential for a system to be delivered which does not meet the customer's expectations for its operation, perhaps it does not work as they believe it should have been delivered or they believe that they have worked in line with the requirements as documented. This can lead to development rework, which is likely to delay the timescales for delivery and cause friction between the customer and the software team.

2. Ensure all Non Functional Requirements are clearly defined early
Many organisations focus solely on the Functional aspect of systems – WHAT it does – rather than the Non-Functional – HOW it does it. Both Functional and Non-Functional elements comprise everything that is required from a system.
Without a clear and early definition of Non-Functional requirements, it is possible that a system could be delivered which does exactly what the customer wants but it difficult to use, slow, insecure, unreliable, or is not scalable. Once again, this can lead to development rework although some non-functional areas are so integral to the design of the product it can be difficult to correct them without affecting the system.

3. Ensure Change Control procedures are in place
Without Change Control, system development can find itself in an uncertain and constantly changing state. For example, development may be making changes to software which are not required by the customer or which are not signed off. Changes might also be made to the requirements whilst development continues and each change without a proper impact assessment.
As there is no control over what requirements are available or not, but they can start to develop their knowledge of the changes is passed on in a timely fashion. Additionally, the impact of some changes might be so large that they cannot be incorporated into the software project at this stage. In this case, Change Control procedures will ensure that a formal impact assessment rework might have to take place. For these reasons every change should be strictly controlled. Change can be allowed, but should only take place after a full impact assessment has been carried out and agreed by the customer.

4. Ensure strict Configuration Management is in place for code, data and documentation
Without Configuration Management, testing of a high quality is impossible. This is because there cannot be high levels of confidence in the right version of the software or documentation is being tested or that the right version will be released to the customer.
Organisations with Configuration Management will have control over version of a product and can perhaps release a "gold" version, a version which then enters the field UNTESTED. It is viewed as the potentially error faults may have been introduced which change the behaviour of parts of the system from that which was intended or, in some cases, even disabling the entire system.

5. Focus quality approach on product risks, focus on preventing rather than detecting defects
It has been proven many times that the later in the software development lifecycle that a defect is found, the more costly it is to rectify it. For example, correcting a mistake in a requirements document before development begins is a far quicker and cheaper activity than correcting the software after it has been developed against the faulty requirements document.

6. Begin UAT/User Acceptance Testing) test design during requirement definition
If an organisation is to be sure that a system has been developed correctly, then the test team should be involved with the project from the very beginning during the requirements definition. Not only can the testers use their expertise and experience to help assess whether or not particular requirements are testable or not, but they can also start to design the tests which will be executed to ease the test schedule later.
In this scenario, the User Acceptance Tests should correspond directly to the user requirements. This is because the requirements are the items that the user wishes to see in the system and the tests represent the way that the system is proved to meet those requirements. It clearly follows that the best time to write the UAT is when the requirements are being constructed.

7. Begin system test design during the system design stage
It is important that the system is designed to be easily testable, if possible. This can only occur if testing expertise is involved in the system design stage. Outlined below are some of the ways that testers can help to ensure that the system produced fulfils this aim. During this early involvement, the testers can also start to construct the System Tests, which helps ease the testing schedule. An additional benefit to this approach is the enhanced team collaboration between the system design and testing team.

8. Focus test effort on unit test where it is cheaper
As previously stated, correcting defects earlier in the software development lifecycle is far easier, quicker and cheaper than otherwise. Given this premise, it is important that many regression tests are carried out at unit test level. This is because the main reason for regression testing is to find errors that have been introduced by the reworking of software, for example, to correct defects. However, not only does this approach find potential defects much later stages of testing than they should, it is possible that they may not be found at all as later stages of testing normally perform different testing activities, in different ways, to those used in unit testing.

9. Plan test approach using risk to define focus of test effort
Test time is often limited, due to reasons like fixed release dates, or late software delivery from the development team. This can often mean that not all of the planned tests can be carried out.
In this scenario, it is vital that those elements of the system that are considered critical to product success have been tested. Therefore, the approach to testing should be driven by the risks that the project wishes to address. For example, if the system has a high medium risk and low risk areas of the system can be tested, in turn, as time permits.
This risk-based approach to testing ensures that, whatever testing activities that are carried out, the most important elements of the system which remain untested (and potentially flawed) are the lowest risk areas possible from those that are left to be tested.

10. Use tools to test for functional and non functional errors very early on when they are cheaper and more effective at removing errors, and throughout the lifecycle
Where automated test execution is appropriate, it can be repeatedly utilised from an early stage of the software development lifecycle, as a 'smoke test'. This method has the potential to find a lot of defects quickly at an early stage, saving time in later test phases.
Additionally, sometimes specific tools may indicate flaws in the system design – which should be exposed as early as possible. For example, poor performance tools can indicate parts of the system that are available, might reveal a poorly designed module, which would hinder the system beyond the tolerance levels of the user when released.
In this way, the tools can be used to identify errors, which are difficult to find by manual testing, or a activity that would be much harder if performance testing was only conducted when the system was about to be delivered to the customer.

Martin Adcock,
Experimentus Managing Director, commented:

"At Experimentus we ensure our clients have the strategies and the most appropriate processes in place to get the most out of their software. In the current economic climate, more for less or finding areas where our clients can reduce costs across the development lifecycle are increasingly becoming the business drivers."

NEWS ALERT!

Experimentus Director Wins 2008 European Testing Excellence Award

- **Geoff Thompson wins 2008 EuroSTAR Award**
- **Recognition of his leadership and contribution in software testing**

Geoff Thompson, Director of Experimentus has won the 2008 European Testing Excellence Award.

Mr Thompson received his award at the European EuroSTAR event held in Holland. The theme of this year's conference was "The Future of Software Testing" a topic that Mr Thompson has addressed recently within various industry speeches and articles.

The European Testing Excellence Award is presented annually to an individual who has made a significant contribution to the field of software testing within Europe. It recognises individuals who have demonstrated leadership and excellence in solving testing challenges through innovative approaches and solutions.

Martin Adcock,
Experimentus Managing Director, commented:

"This is a great achievement and great recognition from the industry for the work Geoff has done to help develop the software testing and software quality management industry in Europe and in particular the work he has put into the BCS SIGIST, UK Testing Board, TMMi Foundation, ISTQB and ISEB over the years."

Focus on well documented Functional Requirements

Every software product, no matter what its type or application, has Requirements. A Requirement is literally "that which is required, a thing demanded by circumstance". Every software product or service has a purpose, that which it is constructed to achieve, and thus will contain at least one Requirement. After all, if there is no purpose to a product, then it is not constructed for to do so it expends time, money and energy for no return.

Therefore, every software product has an aim. From this, it could be said that every software product has a Requirement, or a number of Requirements, that it must fulfil.

The importance of Requirements to the product depends entirely upon the product achieving its preconceived aim. However, the real world is a lot more complex than this.

Indeed, when it comes to measuring the success of a software product, there are many factors that can influence this and that are not directly relate to satisfaction of the initial aim (or Requirement) of the software. This is because the requirements of the user are changing during the course of the product being created or even cases where the aim has not been met but the product of work which is considered to be of value has been achieved that the product can be used for a different aim than for the case in incremental, evolutionary-style software projects.

Despite these reasons, most software projects will be viewed as successful only if they achieve the original aim of the project, that is, if they achieve their Requirements.

It should follow that in order to construct a successful software product, the development team has merely to adhere to the defined Requirements. After all, if the software follows the Requirements with a high degree of accuracy, surely success must result. Once again, the real world is significantly more complex than this.

(Note: For reasons of simplicity and brevity, this paper specifically excludes "quality" development methods, where the Requirements management process is significantly different to that described here.)

One example would be where problems occur in misunderstanding the Requirements.

Depending upon the roles used within the software project, the person writing down the Requirements may or may not be the eventual user of the product. The development team might then develop the product directly from the written Requirements or, more commonly, a specification may be produced by the development team into a technical specification may be created.

The technical specification might then split split into two parts, the "functional Specification" (and sometimes the "Non-Functional Specification") or even into three (the Functional Specification / The division between the Functional Specification and the Technical Specification is where the work is viewed that there is a significant translation to be made between the "Real English" of the original Requirements, and the "Technical English" of the Functional Specification which is unambiguous and which developers can work with more easily) and so the extra translation step of the Functional Specification, between the Requirements and the Technical Specification.

This process can, of course, lead to a game of 'Chinese Whispers', especially if different members of the software project are writing each of these documents. The author of the Functional Specification may or may not fully understand the Requirements that he is using as a guide for his document. This should not be a problem if the project has developed a full understanding of the Requirements. If this situation, clarification can be asked for and hopefully received from the project team. However, it could be that the author misunderstands the Requirements but in fact have misunderstood, even in a subtle way, the intent of the Requirements in this documentation. Equally, these errors can then be reproduced and potentially magnified at the technical specification level. This process means that they understand