It’s Just Common Sense

Barbara Gorton

Presentation

F12

Friday 23rd November, 2001

International Conference On

Software Testing, Analysis & Review

November 19 - 23 Stockholm, Sweden

It’s Just Common Sense

Barbara Gorton

EuroSTAR November 2001

barbara.gorton@consignia.com

Benefits of working on old systems

- Know the customer personnel.
- Knowledge of customer business, vocabulary, priorities.
- Familiar document set and style.
- Should have a good idea of impact of change.

Disadvantages of working on old systems

- Relationships with customer may be too cosy.
- Documentation may be written to old standards and may not be adequately maintained.
- Mind set does not look for process improvements.
- Lack of tools and no budget to buy one.
- Tendency to be forgotten by senior management.

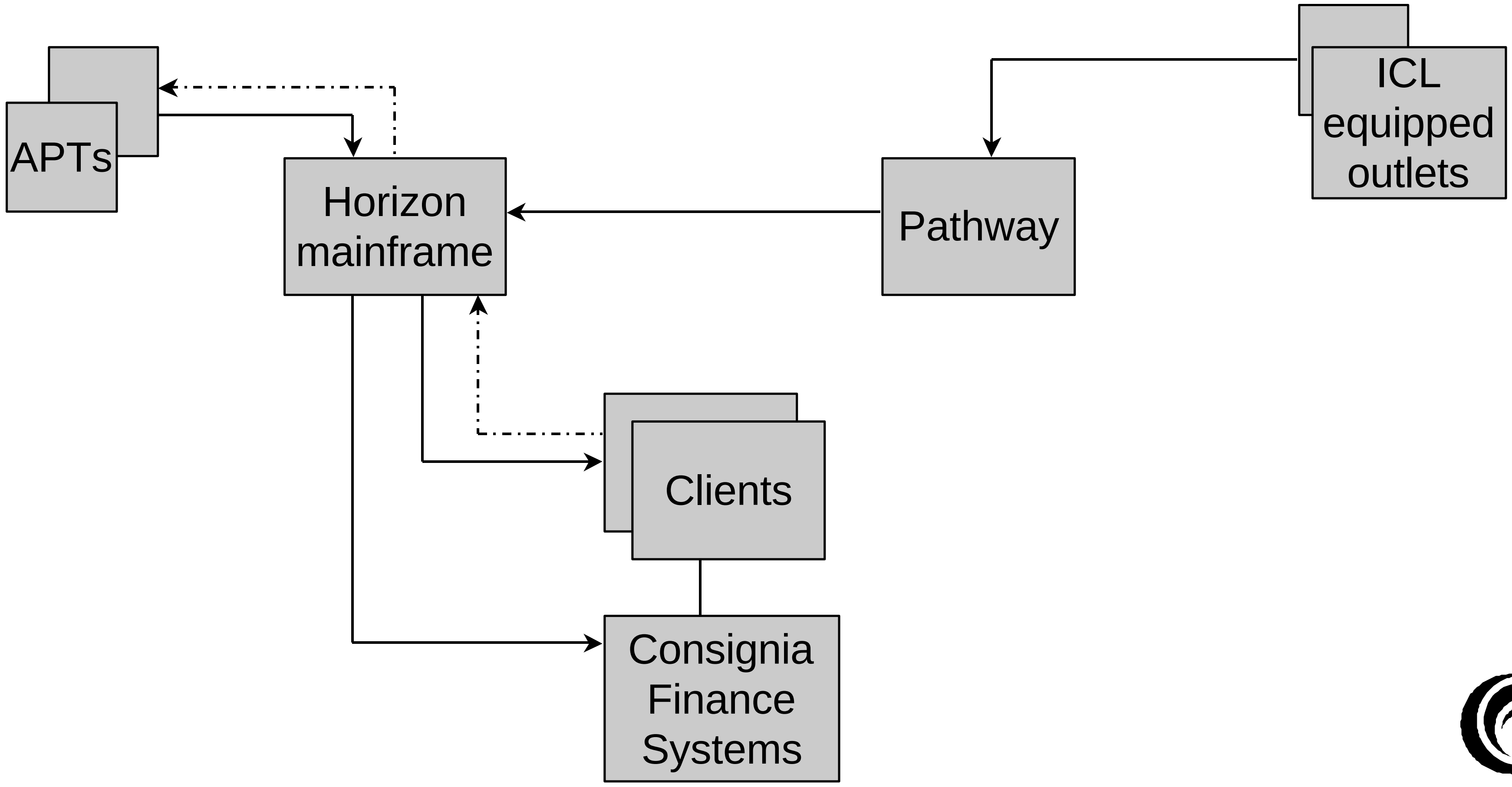
Horizon System

- Client server, transaction processing.
- Based on Tandem mainframe (Cobol) and APT equipment at post offices (C).
- Functionally relatively simple.
- In existence for around 10 years.
- Due to be phased out into replacement ICL Pathway system in two years.

Horizon System at start

——— transaction data

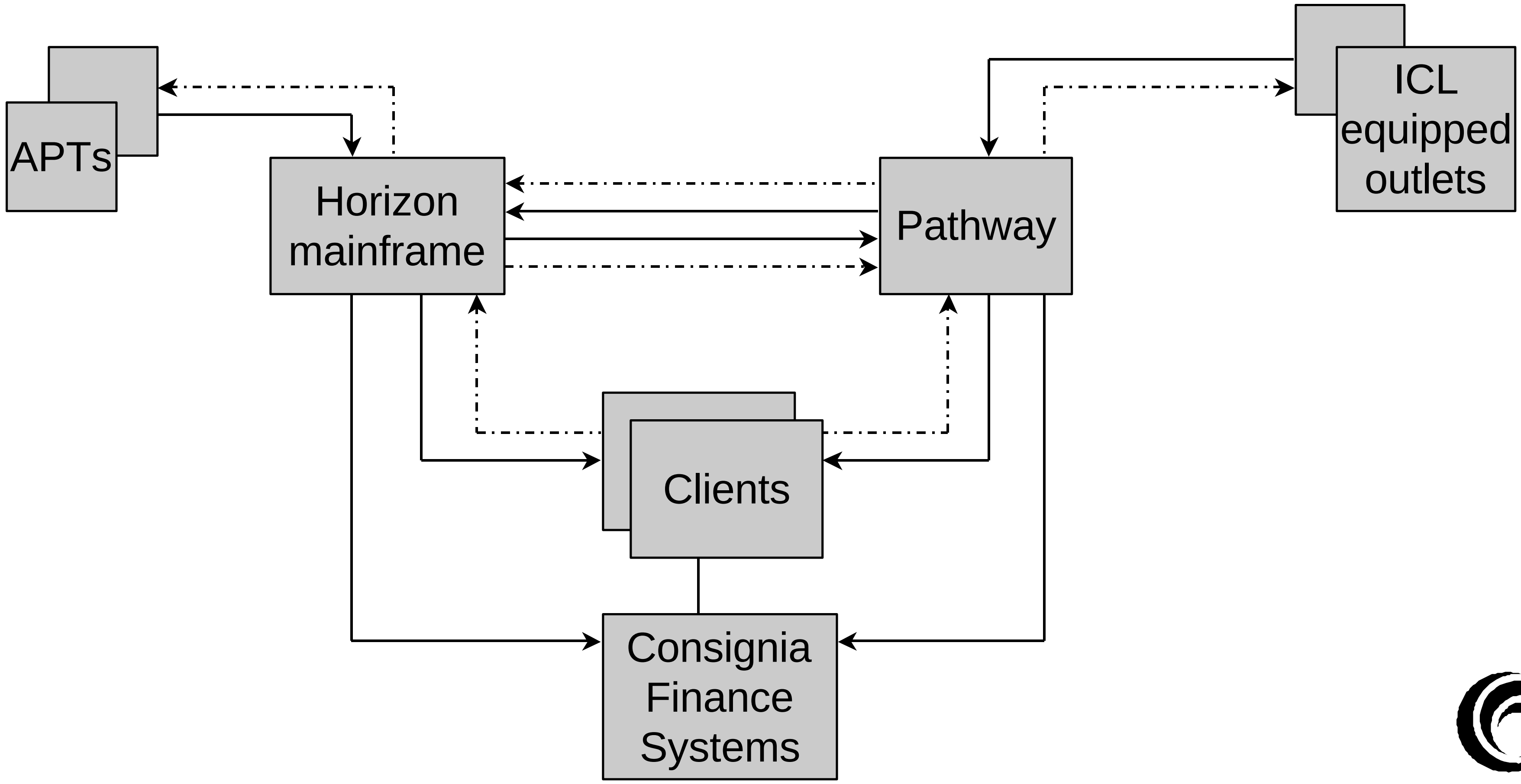
-·-·-·- client supplied information



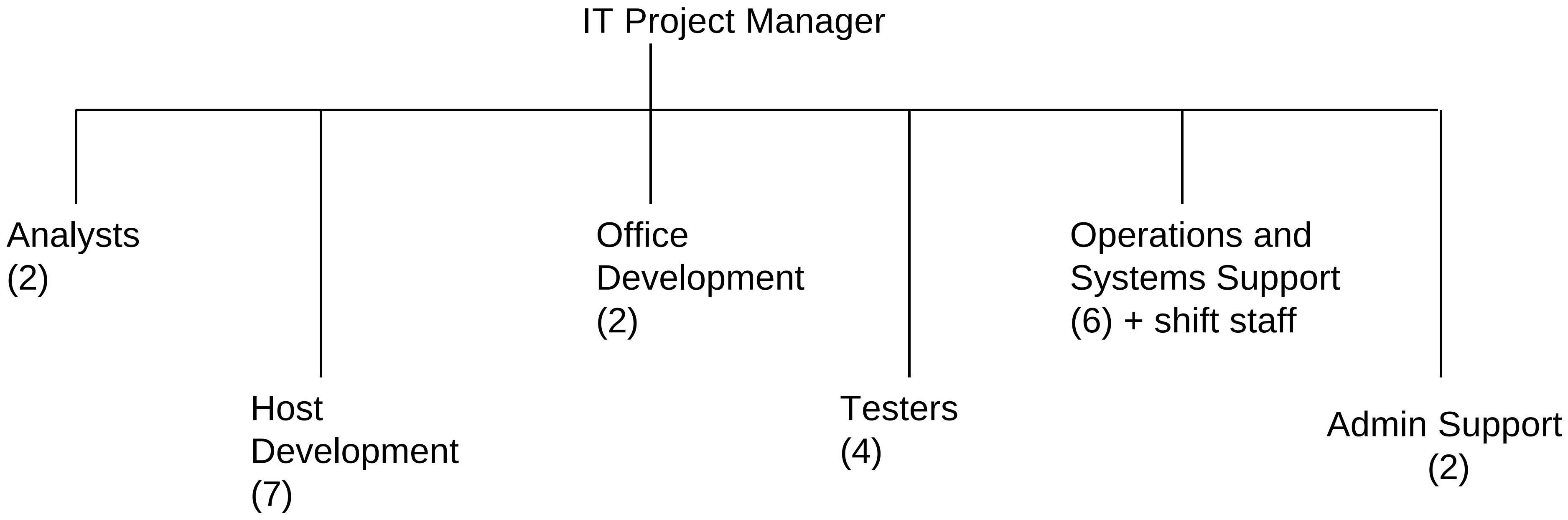
Horizon System at its most complicated

——— transaction data

-·-·-·- client supplied information



Horizon team structure



Each functional area had a manager responsible to the IT Project Manager

What the testers said:

- Last phase did not go too well.
- We did not really understand the changes that were made.
- We were never quite sure which program versions were in the test environment.
- No one (especially Development) seemed to have time to help us and explain.

and:

- We do not understand the Tandem.
- No one has told us how the system hangs together.

In addition

- The testers were from non-technical backgrounds.
- They were clearly upset about the implications of the outsourcing programme.

What the Host Development Manager said:

- The testers do not understand <u>my</u> system.
- X is a waste of space, the others try but do not know what they are doing.
- Testers are not testing the right things.
- Testers take up too much of our time, they need so much help and support.

and:

- When they find faults they often are not really faults at all and they do not give us enough information about what they were doing.
- Testing should be under development control.

- He would be delighted if testers got involved at the high-level design stage.

The Office Development Manager:

was kinder about the individuals, but thought they did not really get beyond the obvious.

The Analysts:

felt the testers tried hard but maybe did not get they support they needed

The Operations Manager said:

- They are OK, but do not stand up for themselves.
- They do not seem to know what really happens on the live system.

In addition I had discovered that

- The Project Manager thought testing could be done better.
- One of the test team thought he should be doing my job.
- The Host and Office Development Managers did not seem to know much about each other’s areas.

and:

- Fault reporting was paper-based and not even held centrally.
- The next phase of the project was business critical, it would be high-profile with real financial penalties for late delivery.
- Everyone seemed to have had problems with the previous Test Manager.

What next?

Things I could not deal with directly:

- Morale surrounding outsourcing.
- The reluctance of the testers to stand up and be counted.
- Other people’s views of the testers.
- Issues about the phase just ending or the previous Test Manager.
- Personality differences.

Things I could do something about:

- Basic understanding of how the Tandem worked
  - ⇨ attend supplier course.
- Knowledge of how the existing systems worked
  - ⇨ presentations by development managers
- On-line fault reporting system
  - ⇨ educational opportunity - build our own.
- Build confidence in testing capability
  - ⇨ testing techniques course and ISEB.

Test Specification

Was previous documentation an acceptable model?

Previous Conditions

- A set of statements with no expected outcomes.
- Impossible to tell how much had been covered.
- Cross-referencing was difficult to follow.

Previous Scripts

- Steps were a mixture of specific and generic.
- Referred out to ‘how to’ documents (which were equally flawed).
- Expected results were too vague, often just ‘check results’.
- Repeatability was by luck not design.

Something had to be done:

- The first and fundamental change
  - ☆ every condition and script step will have a clearly-defined expected outcome.
- Test techniques to be used to improve the extraction and presentation of conditions.
- New rules for script writing to be agreed.

Cause Effect Graphing

- easier to see what might be missing
- good way to show a lot of information in a simple format

Phase 4 (all clients migrated to ICL Pathway)

	Condition	1.31	1.32	1.33	1.34	1.35	1.36	1.37	1.38	1.39
Inputs										
transaction for client owned by	HAPS	no								
	ICL Pathway		Y	Y	Y	Y				
unknown client							Y	Y	Y	Y
transaction type	mag/bar		Y		Y		Y		Y	
	GEC watercard			Y		Y		Y		Y
collection office owned by	HAPS		Y	Y			Y	Y		
	ICL Pathway				Y	Y			Y	Y
sent to HAPS from ICL Pathway					Y	Y			Y	Y
Outputs										
HAPS sends transaction to	client		N	N	N	N	N	N	N	N
	ICL Pathway		Y	Y	Y	Y	N	N	N	N
HAPS sends to FCD via	Apachi		N	N	N	N	N	N	N	N
	OpTIP		Y	Y	N	N	Y	Y	N	N
Disowned notification sent			N	N	Y	Y	N	N	N	N
Undeliverable transaction			N	N	N	N	Y	Y	Y	Y
	Script reference	**	I4b	I4b	I4b	I4b	I4b	I4b	I4b	I4b

no = cannot happen

Narrative

- better for more complicated and special situations

Cond	action / comment	expected result	phase	script
5.33	HAPS collects transactions from AP terminals and ICL Pathway for a variety of clients. The data is processed (including rollover) and the transactions are notified to FCD via OpTIP. System is reconciled. (Create Transfer process CLS798CS)	Attempt to send transactions (successful or failures) and the volume and value of transactions are reported correctly in the Client Communications report. The values will match those in the Transaction Volumes report (which reports uploaded data) and those in the ALPS Reconciliation Report. The system will reconcile correctly. The number of transactions in and the header record of the transaction file are correct.	2,3	I2a, S2a
3.6	Attempt to set illegal value for CREATE-TRANSFER-GROUP for a 'real' client (eg to PNAG or GEC or spaces) is prevented.	This is not prevented by design. Operational procedures should prevent it happening when the system is upgraded in the test environment, and when client migration is required by the test scripts.	any	$$$

Boundary Conditions & Partition Analysis

Especially useful with rules for new card

- Minimum and maximum acceptable values for transactions and on card.
- Service charge.
- Calculations and rounding.

State Transition

- We needed to make sure all possible paths through the APT menus were covered and we ended up with a rather complex and confusing document.
- I have subsequently learned that state transition diagrams would have helped!

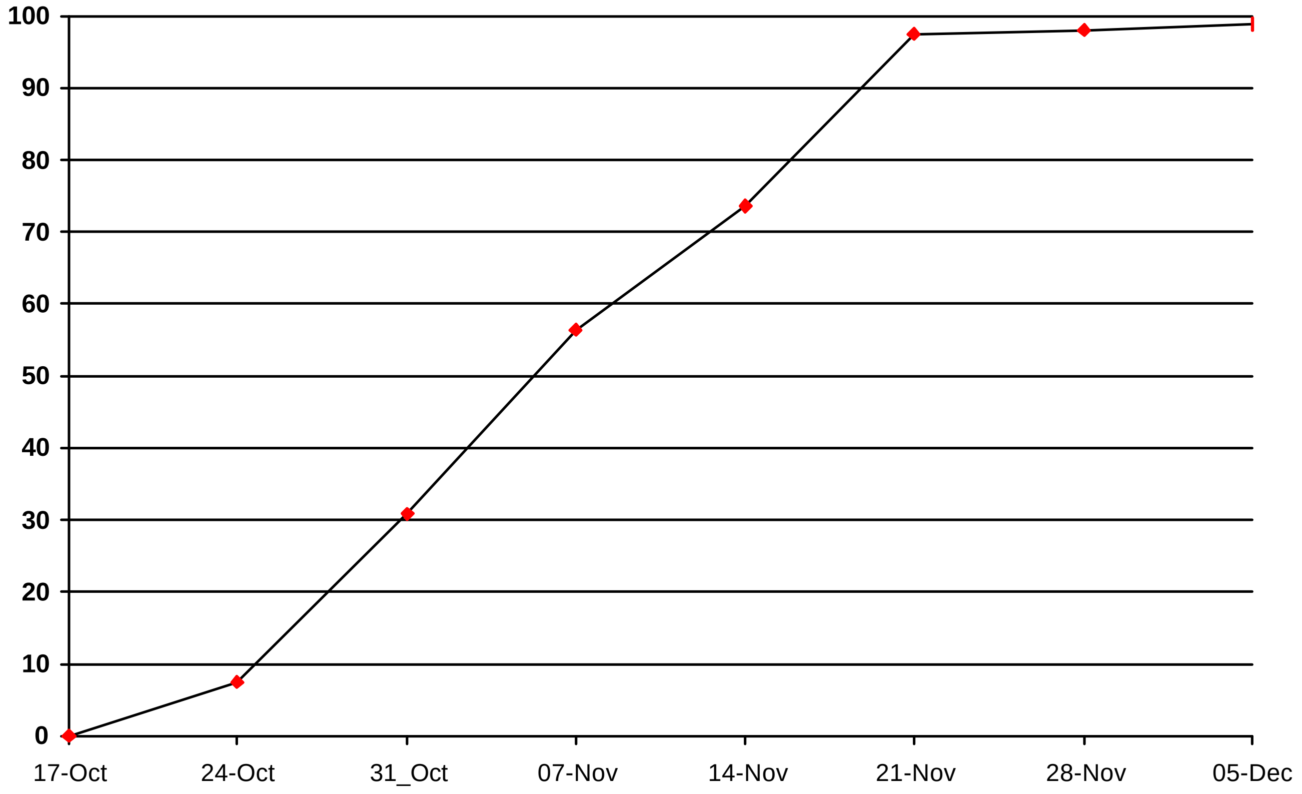
Comments from review

- “It is really useful to see clear expected results.”
- “Procedures are nothing to do with system test, you only need to test the programs.”
- “We had better do something about procedures.”
- “You don’t need to test all that, do you?”
- “I never realised there was so much analysis and preparation to do before you test anything.”

Rules for scripts

- Each script will be self-contained.
- Each script must contain enough detail to make it repeatable.
- Each step will be numbered.
- Each step will refer back to a condition.
- Inputs will vary across client and value, some duplication is desirable.

Progress measured as percentage of conditions successfully tested



We must have done something right!

- Previous phase:

  41 faults found in System Test

  39 faults found in UAT + Live.

- This phase:

  73 faults found in System Test

  0 faults found in business integration or UAT

  1 fault found in Live.

Achievement

- Significant increase in faults found, leading to a better quality product delivered to the Customer.
- A major improvement in the professionalism and confidence of the testers themselves.
- A definite improvement in the relationship between the test team and the rest of the project team.

All for

- The recognition that improvements were needed and could be made.
- A small amount spent in training.
- A bit of energy to drive change through.
- The application of good testing practice.

It’s just common sense!

It's Just Common Sense

Barbara Gorton

Contact:

email: barbara.gorton@consignia.com

phone 01252 528036

address: Consignia plc
Concept House
250 Farnborough Road
Farnborough
GU14 7LU

Abstract:

Large legacy systems still exist and are being regularly enhanced and tested. Such systems often have staff who have been assigned to the system for some time and who have become used to working in a particular way; frequently without the benefit of testing or management tools. Organisations are generally reluctant to spend anything extra on such projects and it is tempting for the staff to accept things as they are and not look for improvement.
Based on practical experience of such a system, we show how a significant improvement in the quality of changes delivered into User Acceptance Testing can be achieved by approaching the situation with common sense and the application of basic good testing practice.

Biography:

Barbara joined Consignia plc (formally The Post Office) in 1987 after performing a variety of IT roles in public sector computing. After some time in the Quality and Standards areas, during which Consignia achieved ISO9001 certification, Barbara was introduced to Testing as a separately identifiable activity in 1995 and has been happily testing on various large mainframe systems ever since. She is a regular supporter of SIGIST and holds the ISEB Foundation Certificate in Software Testing. Currently she manages a small testing group supporting the TV Licensing Project.

The last few years have seen an exponential increase in the number of testing tools designed to help in the execution and management of testing and there is no doubt that, when the right tool is chosen and correctly used, it can make a significant impact on the cost and quality of testing. However, even without tools, it is still possible to improve testing by making simple, cheap changes based on good practice and common sense. I would like to illustrate how this happened during the testing of a particular project phase with which I was involved.

I am employed by Consignia plc (formally The Post Office) in its Business Systems section, which is responsible for providing IT solutions to the rest of the organisation. A while ago, I had just finished the Y2K testing on a large project when I was invited to join the Horizon System project to manage the testing of the next release. The project had a good reputation and the IT project manager was respected, so I was happy to accept.

Y2K testing had been a very hectic and challenging time, so I was looking forward to returning to more traditional testing and, hopefully, less stressful timescales. I also thought I would be able to enjoy the benefits of working on an old, established system where the project team members would already know the Customer's personnel, business, vocabulary and priorities; where there would be an established documentation set and style and a good understanding of the impact of any proposed changes to the system.

Of course, there are possible disadvantages of working on old systems, for example, the relationship with the Customer may be too friendly; documentation may be written to old standards and may not be adequately maintained; there may be a lack of tools and no budget to buy one; and a mind set may have developed that just accepts the status quo and does not look for process improvements.

A little bit of information about the project.

At post offices throughout the UK, Consignia accepts payments from members of the public on behalf of clients such as telephone, gas, electricity and water suppliers, local authorities, and TV licensing, using bills (with reference details or bar-codes), magnetic cards or smart cards. The customer receives a receipt detailing his transaction. Later in the day, the client receives information about the transactions and, eventually, settlement. Each post office outlet receives a daily update of tariff charges and a reconciliation report to help the end of day cashing-up process. The Horizon System provided the IT capability for this service.

The Horizon System was essentially a client server, transaction processing system based on a Tandem mainframe (programmed in Cobol) and special purpose-built equipment called APTs (programmed in C) installed at post office outlets. The APTs were used to collect and verify information about customer transactions. The data was uploaded by the mainframe which sorted all the transactions and sent information to each client about its customers. In addition, the information was also sent to the central finance systems, for further processing and money transfer.

The Horizon System was due to be phased out over the next couple of years and replaced by a new project, the ICL Pathway System, which would replace all the APTs and the mainframe system. The previous Horizon release had started the process by establishing links with the Pathway mainframe, allowing transactions collected by Pathway equipment to be passed to the Tandem for processing. The new release would introduce a new type of payment card and also put in place all the code to allow the Horizon System gradually to transfer all the outlets, central processing and interfaces with other Consignia systems to the Pathway System so the Horizon System could be switched off. There were four separate phases to full implementation, all to be tested in the one release.

The amount of change needed for the new release was roughly equivalent to the previous one.

The project team structure was fairly typical. The twenty three or so permanent and contract staff were split into functional teams (analysts, mainframe developers, office developers, testers, operations and systems support, and administration staff), each headed by a manager, responsible to the IT project manager.

I met all the project team on my first day, at a team building exercise during which we were informed that the Horizon System had been selected for an outsourcing programme. This did little for morale! During the course of the day, I had noticed that the testers did not seem to have a close relationship with the rest of the project team, and that the testers' inputs were often disregarded during the exercises.

The next day, I sat down with my new team (whose members had been part of the project team for between two and seven years). I discovered that the testing of the last phase had not gone too well. The testers felt that they had not really understood what the changes to the system had been about. No one (especially in the development team) seemed to have had time to help them and explain things, and they had never been completely sure which versions of which programs were in the test environment. They also said that they did not know much about the Tandem and did not really understand how the systems hung together - courses had been promised but had not materialised. I was unsure if they were unwilling or unable to identify the issues and do something about them. None of the testers had come from a technical background and they were all clearly upset about the implications of the outsourcing exercise.

When I talked with the other managers, trying to find out what the working relationships were and how they might need to be improved, I got plenty of feedback, although not quite what I was expecting!

The mainframe development manager felt that the testers did not understand his system. The testers took up far too much of his team's time, they needed so much help and support. They were not testing the right things and, when they did find faults, they often were not really faults and, if they were, the testers did not give the developers enough information about what they had done to help in the fixing process. In all, he felt testing should come under development control. He did say that he would be delighted if testers got involved at the high-level design stage.

The office development manager was kinder, but thought testing never got beyond the obvious. The analysts felt the testers tried hard, but maybe did not always get the support they needed.

The operations manager said that the testers were 'OK', but did not stand up for themselves. She added that they did not seem to know what really happened in the live system.

Clearly, there were some people problems here!

I had already discovered that the project manager was dissatisfied with the testing of the previous phase. She thought it could have been done better because too many faults had been missed. The fault reporting system was paper-based and not held centrally, so there was no way of analysing past faults.

On the plus side, everyone on the project was used to reviews being held on almost every document that was produced, and the project manager always allowed time for reviews in her plans. There was a comprehensive set of project documentation that was kept up to date and under version control.

I had not expected so many concerns in a well-established project. Things needed sorting out and, as the next phase of this project was business critical and would be high-profile with real financial penalties for late delivery, so we could not afford to get it wrong!

I could do nothing much about the impact of outsourcing on morale; nor could I do much about the phase just ending. Personality issues seemed best left alone.

It seemed to me that the only way the testers were going to get respect was to give them the knowledge, and vocabulary, to do their jobs better. My first action was to get all the team on an Introduction to Tandem course. I then approached the mainframe development manager and asked him to give an overview of his part of the system and to tell us everything he thought we ought to know. Despite his reservations, he gave a really good overview at the right level, including much of the live running information. The office development manager was equally helpful.

During this period, I reminded my test team that it was each individual's responsibility to get as much as possible from the information being given and that I expected them to ask questions and not just sit there. I dismissed criticism that there was too much information being provided, testers needed to understand as much as possible about the system to be able to test it properly and they would not get another chance.

Another issue to be addressed was the lack of a mechanised fault reporting system, especially as there was no budget to buy one. Some improvement work had been started on the old paper-based system and I was keen to get the team involved by building on the earlier work, especially to clarify and simplify the classification of incidents. One benefit from the outsourcing review was that budget was available for staff to train for new work. As one of my team wanted to get into PC development, I was able to arrange for him to attend an Access course and then to build us a basic fault reporting system. It was true win-win. He acquired a new skill and could demonstrate he had used it. We got a fault reporting system. The team learned that defining requirements is not easy! I had also invited the development managers to get involved, I needed their support once the system went live to supply fix information and to agree a way of defining program versions. It also allowed me to take back control of what was put into the test system and when.

While this was happening, I sent the most junior team member on a testing techniques course (the others had already attended) and persuaded one of the others to go on the course leading to the ISEB Foundation Certificate in Software Testing (which he obtained). It was not until later that I realised what a good decision that had been. Both came back more motivated than before and, when I publicised the success, the project team members started to realise that testing was just as special a skill as programming or analysis.

By this time, the requirements had been defined, reviewed and signed off. I had raised the testing profile by attending and contributing to all the high-level design reviews. We were ready to start specifying the tests. All the team members had been on testing training courses, so I imagined that my problems would be over. They were not!

I asked the testers for examples of test specifications they had produced previously. If the design and layouts that the testers had been used to were adequate, I did not want to change them. What I found were conditions that were nowhere near specific enough with no expected outcomes. It was almost impossible to judge what had been covered. The test scripts barely justified the description, they were imprecise, often referred out to 'how to' documents and rarely gave expected results - most of the time if just said 'check results'. They had been produced following guidelines that were outdated.

So we went back to basics, starting with defining test conditions. The first and fundamental change was that every condition would have a clearly-defined expected outcome. We looked at techniques we might use to make sure we could have a confidence we had considered all the functionality to be tested, both from an individual process and the whole system viewpoint. (Processes throughout both systems and their interfaces with each other and with the 'outside world' were being changed).

This is where the table part of Cause-Effect Graphing came into its own. There were a limited number of types of easily identifiable inputs (eg, transaction type, media, client type, outlet type, processing mainframe, client owners, etc), and each had a limited number of options. (eg three different client owners, two groups of transaction types). We were able to construct tables showing the possible combinations, although we slightly 'customised' the technique to suit the circumstances. The patterns that emerge with this technique certainly helped to ensure coverage was complete.

Some of these tabulations lead naturally to a look at boundary conditions and partitions, particularly at the APT end of things.

We also used a narrative type of condition, particularly when Cause-Effect Graphing became clumsy. This allowed a consideration of the flow of data through the system, from APT to client. It was also very useful where a complicated set of circumstances applied, particularly in the timing or sequencing of transactions being recorded at the APT, or looking at procedures.

Lastly, we looked at the navigation through the screens on the APT and tried to map the various combinations. Had we been better familiar with them, we could have used the state transition techniques, which would have made a better job of it.

We also included conditions to test for the various procedures that would be needed to move from one implementation stage to another. This was a bit difficult as the procedures had not yet been written!

Once identified, all the conditions were assembled into a document which developers, analysts and the Consignia Customer were invited to review. Some of the constructs were new and needed to be explained to various participants, most of whom were interested enough to learn them. I also informed the review body that the test scripts would not be formally reviewed because it can be very time consuming and, if the conditions were correct and complete, the script is best 'reviewed' by running it and carefully examining any discrepancies.

The feedback from the review meeting was interesting. There was general surprise at how many tests had been identified. The Consignia Customer and project analyst realised they needed to do something about procedures. The common response was that it was really useful to see clear expected results (especially as our test analysis had identified some unexpected results!). The non-review of scripts was endorsed.

A set of scripting rules were agreed: scripts will be self-contained; they must be repeatable; each step will be numbered and will refer back to the condition(s) it is testing; transactions would be made using varying amounts for a mixture of clients.

All this was going along really well. A good working relationship was developing between the testers and the rest of the project team and, more importantly, my testers seemed to start to believe in themselves and take on issues rather than expect me to sort out every problem.

I then discovered that 'testing' had become a high risk project issue! This was because the project manager had no confidence that this 'new' way of testing was going to work or be completed in time. Usually, test running would have started by now and she would have expected to see some plans and progress graphs. With all the other improvement activity, I had neglected one rather important area, that of formally reporting progress to keep the project manager happy.

It was easy to put together a plan for when each script should be ready and when we expected to start and finish running it. I had no project or test management tools, so just used Microsoft Word. It was not pretty, but achieved the result.

Once test running started, I needed to report progress. The previous phase had produced lots of little scripts, we had fewer, much longer scripts, each covering a business day. Our first scripts were designed to exercise as many parts of the system as possible. These early runs exposed quite a few faults (including the inevitable environmental ones) which had to be fixed and retested before we could move on. Reporting the number of scripts completed (the method used on the previous phase) made progress look bad and did not match my 'gut feeling'. I realised that what really indicated progress was the proportion of conditions successfully tested. This produced a traditional 'S' curve and seemed to satisfy the project manager. I also graphed the incidents raised, fixed and retested each week.

We managed to complete the initial testing and all the retesting just within the project target. This was more pleasing given the amount of additional work, which was not part of the original plan, in which the test team had become involved.

At the end of the system test, I felt that we had injected a much more professional approach to testing into the project and it was proving effective. The testers seemed to have far more confidence in what they were doing, to the extent of being prepared to discuss faults with the developers and not be fobbed off, and I felt they had earned, and were receiving, the respect of the rest of the team.

We must have been doing something right. The previous phase had found 41 faults during system test. Another 39 had been found during UAT and live running. In our phase, we found 73 faults during system test (including one procedural one that, had it gone live, would have brought the system to its knees). No faults were found during the subsequent UAT and business integration tests, and only one fault during the first three months of live running.

In conclusion. I had not gone looking for improvement opportunities, they cropped up all on their own. It just needed some digging to establish what the real issues were and then to deal with them, one by one, by spending a relatively small amount of money, using a bit of energy to drive change through, applying good testing practice - and a heap of common sense.

So - how much better could your testing be if you went looking for improvements?

Friday 23 November 2001

F12

It’s Just Common Sense

Barbara Gorton

Barbara joined Consignia plc (formally The Post Office) in 1987 after performing a variety of IT roles in public sector computing. After some time in the Quality and Standards areas, during which Consignia achieved ISO9001 certification, Barbara was introduced to Testing as a separately identifiable activity in 1995 and has been happily testing on various large mainframe systems ever since. She is a regular supporter of SIGIST and holds the ISEB Foundation Certificate in Software Testing. Currently she manages a small testing group supporting the TV Licensing Project.