Integrity Software Testing

ABN: 50 141 155 185

Queensland Health Payroll System Commission of Inquiry
C/- Ashurst Australia
GPO Box 9938
BRISBANE QLD 4001

Expert Report

I refer to your letter dated 17 May 2013 seeking my expert opinion in this matter.

I understand that I have been retained to prepare a report in respect of matters before the Queensland Health Payroll System Commission of Inquiry (**"Commission"**).

Qualifications

Please find attached a copy of my current Curriculum Vitae.

Documents Provided

The documentation which has been provided to me for the purposes of providing this report is as follows:

1. Report of Brett Cowan of K.J. Ross & Associates in respect of User Acceptance Testing conducted by Queensland Health, dated 27 January 2010
2. Witness Statement of Brett Cowan in the matter of the Queensland Health Payroll System Commission of Inquiry, dated 16 April 2013
3. Report of Dr David Manfield, dated 30 April 2013
4. The System and System Integration report of 27 April 2009
5. Transcript of hearing of the evidence of Brett Cowan before the Commission on 2 May 2013
6. Extracts from transcript of hearing of Dr David Manfield before the Commission (15 May 2013 pages 30-49 – 30-65; 16 May 2013 pages 31-7 to 31-10)
7. Letter of Clarification, 3380_001 dated 21 May 2012

References

In preparation of this report, I have referred to the following reference books and journal articles:

1. Using Requirements Management to Speed Delivery of Higher Quality Applications, Alan M. Davis and Dean A. Leffingwell 1996.
2. The International Software Testing Qualification Board (ISTQB) foundation syllabus 2012.
3. Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them, Journal of Object Technology, Vol. 6. No. 1, January-February 2007, Donald Firesmith.
4. Institute of Electrical and Electronic Engineers. IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). New York, NY: Institute of Electrical and Electronics Engineers, 1990.
5. A concise introduction to software engineering, P. Jalote, 2008.
6. The Practical SQL Handbook, Using Structured Query Language, Judith S. Bowman, Sandra L. Emerson, Macy Darnovsky, 1996.

Opinion

My opinion about the specific questions raised in your letter dated 17 May 2013 is as follows:

(a) As a testing manager, would one need to read and understand the system scoping documents to ensure that test scripts were developed and run against the scoped system?

Yes. Every software development project has a definition of scope, i.e. you need to know what it is that you are building and the boundaries to which that scope extends. Scope of a project is frequently subject to change and the level of change is dependent on a number of factors, but the one that has the greatest impact is the complexity of the solution. Complexity can appear in a project through a number of factors:

1. Requirements Complexity: by both the number and complexity of the requirements being addressed. Complex requirements are related to the size and number of business rules being implemented. A payroll system, such as that implemented for Queensland Health with a large number of awards, staffing types and salary categories etc, is a complex system of business rules when compared to other systems such as superannuation and banking.

2. Architectural Complexity: where solutions need to interface with multiple systems in real time or via batch processing techniques.

3. Technological Complexity: through the combination of technologies being brought together as well as the newness of the technology being implemented.

In standard testing practice, the test manager of any phase of testing needs to know what the scope is for each test phase to be performed for the project. This is a fundamental aspect of testing and is defined within the ISTQB Foundation Syllabus [2], the minimum level of Tester Accreditation.

All testers are taught the importance of understanding the testing scope and the requirement for defining and refining what is and is not in your scope. The Test Manager has the responsibility for ensuring that the testing being performed aligns with changes in scope of the solution.

(b) Assume that a high number of defects are identified during User Acceptance Testing on a highly complex system (such as the payroll system).

(i) Is that necessarily an indicator of poor, or inadequate Systems and Systems Integration Testing?

(ii) Are high numbers of defects common in highly complex systems?

(iii) Would more information than a severity indicator be required?

Point (i)

Defects are a core metric tracked by projects to provide an indicator of a system's quality. Like all metrics, they need to be looked at in context and used to trigger further analysis to quantify what the defect metrics mean.

Defects, as a number, are only an indicator that there is a potential problem. Large defect detection rates are an indicator that "something" is not right. While the immediate candidate for the cause of high defect detection rates can be the quality of the system under test, this is not always the case.

There have been a number of studies performed around the different types of root causes for defects. Donald Firesmith [3] noted that:

> *"Poor-quality requirements greatly increase development and sustainment costs and often cause major schedule overruns. As long ago as the early 1990s, it was well known that defects discovered once a system is fielded cost 50 to 200 times as much to correct as they would have had they been found during requirements evaluations [Boehm and Papaccio 1988], and these depressing figures have not changed significantly since then. As noted in [Wiegers 2001], "Industry data suggests that approximately 50 percent of product defects originate in the requirements. Perhaps 80 percent of the rework effort on a development project can be traced to requirements defects."*

The above analysis supports my own experiences of large projects with large numbers of defects: frequently the root cause of the defects is found to be requirements errors.

In order to improve the overall software development processes, understanding where defects are introduced is critical to ensuring that testing is focused on the right areas to mitigate risk as early as possible.

While there are numerous questions in the transcripts concerning the number of defects found during UAT, there is no data that indicates what the root cause for the defects were. The use of the term "functional defect" is used frequently. The definition of a functional defect is purely that the system did not perform a function or operation as expected. The root cause of "functional defects" can be:

1. Incorrect coding;
2. Requirements errors which include missed or poorly expressed requirements;
3. Environment issues including configurations;
4. Test errors due to an incorrectly written test scripts; and
5. The test data is wrong.

One method of measuring the effectiveness of the testing performed in previous test phases is to perform Defect Leakage Analysis. Defect Leakage is used to determine in which phase the reported defects should have been found, i.e. should the defect have been found during unit testing instead of being detected in UAT. Defect leakage analysis would have shown where the defects found during the UAT cycle should have been detected.

Defects are caused by errors introduced by humans and therefore a defect in and of itself is only one dimension of quality. Taken in isolation, the raw number of defects does not support the conclusion that "the previous test phases had inadequate testing performed". If defect leakage analysis had been performed and that indicated that a large number of the defects detected in UAT should have been found in Systems Testing, then the statement that "the previous test phases had inadequate testing performed" may have had a proper foundation.

Point (II)

Yes. Complex projects, by their very nature, frequently have high levels of defects, but what is seen as high in one organisation could be low in another. Software Development maturity affects the number of defects in total that are found.

Complex business systems in my experience embody a large number of requirements, with a large proportion of these being complex business rules. If these requirements experience a high rate of "churn" (change), this will generally lead to higher numbers of defects and I have found that this is frequently the outcome in these projects. This problem can result in code being written to implement a requirement specification that is actually incorrect and test cases verifying behaviour that is in fact not the correct expected behaviour.

Point (III)

Measured by itself, the number of defects at a specific severity level, except those at the highest level, e.g. severity 1, do not necessarily indicate how a system will perform after it goes live.

The standard defect management process uses attributes to define impact (severity) and importance (priority). Severity is usually viewed from a customer, business or end user perspective, i.e. if this function or process doesn't work what is the impact to the end user of the software? Priority indicates how quickly the defect needs to be fixed.

The defect management process is fundamental to both testing and software development. It is taught in ISTQB and every methodology has some definition of the way this process should be performed. In all these processes the key factor that is considered is the severity of that defect.

Testing can and frequently does provide an initial assessment of that severity. In these cases severity is frequently related to how it impacts the ability of the tester to perform more testing i.e. is the defect blocking the testing team from performing further testing? Defects are generally managed through triage processes.

Defect triage processes are meetings that include representatives from key stakeholders to review the defects raised. The triage meeting starts to change focus the closer to the delivery date of the system. That focus changes from a "tester centric" view to a "business centric" view. That is, where the triage focuses on whether the business requires this defect to be fixed for "go live" because, for example, it presents an unacceptable risk to the business.

(c) Assume that:

(i) System and Systems Integration Testing was conducted, audited and accepted in the manner set out in the System and System Integration Testing Report dated 27 April 2009;

(ii) You have not personally reviewed the System and Systems Integration test scripts or processes;

(iii) User Acceptance Testing was not conducted (by the UAT Test Manager or individual testers) by reference to a Requirements Traceability Matrix;

(iv) a high number of severity 2 "functional" defects (for instance, the test returns an error or the system does not complete that process) are discovered during a final phase of UAT,

would you consider it appropriate or safe to reach the conclusions expressed in the K J Ross Test Completion Report dated 27 January 2010 (at p 20) that "system testing was not thorough enough" and "integration testing was not thorough enough".

Based on the documentation I have and the above assumptions, I cannot see in this report any definitive data that would support the statements that "systems testing was not thorough enough" and "integration testing was not thorough enough".

Defect leakage analysis would have provided improved insights and the facts to show whether the defects found had "leaked" from previous test phases into UAT. Without this analysis to substantiate the statements, they are assumptions about the causes rather then statements based on concrete substantiated facts.

(d) On the assumptions expressed above, what conclusions could you reach about the possible causes of the high number of defects?

As I set out above, more information would be required to draw particular conclusions.

A lack of clear requirement specification will cause increased numbers of requirements specification errors and changing requirements scope, i.e. new requirements being added, can further introduce additional requirements specification errors. Based on industry studies, observations and my own experience, a situation where the requirements were not clear would contribute to higher defect numbers

The development team uses the requirements to create software code that implements the condition or capability the requirement defines. If that requirement is incorrectly specified then the code written to meet the requirement will inherit the errors.

The manifestation of these requirement errors is system behaviour that isn't expected by the tester or customer. While the behaviour may not have been expected, it doesn't mean that the code written was not of sufficient quality or that the development team wrote the wrong software code. They wrote code to meet a requirement that was specified.

In Section (f) of this report I address the impact of not having a requirements traceability matrix. In short, the lack of this document prevents effective impact analysis being performed on any change to the requirements. Not understanding the impact of change will result in increased defects, as you do not know what requirements trace to which test cases and therefore which test cases need to be retested or modified to reflect the change.

(e) Assume further that data imported into a proposed system in the course of UAT contains errors of a kind not normally encountered in production.

(i) What impact could that have upon test results?

(ii) Should a system ordinarily be sufficiently robust during UAT that it ought to cope with any bad data which is so imported?

(iii) What if such data is generally *incapable* of being encountered in production (for example, 'child' records without a parent, where such a record could not be added without a 'parent' record, etc).

Point (I)

Importing data into a system that does not match what the system expects to receive can result in issues ranging from data integrity problems through to complete system failure. Any data inconsistency in a test environment will manifest itself as testing failures as the system does not behave in a manner that the test expects, e.g. in batch systems, data problems frequently manifest as batch processing failures.

These test failures can result in the raising of defects that are not an application failure. This leads to increases in the testing effort from the investigation required to determine whether the failure is a system problem, a data problem or a test script error.

The defect metrics reviewed in the K.J Ross Test Completion Report dated 27 January 2010, do not detail how many of the defects raised were caused by environment, data or test scripting errors.

Point (II)

In my experience, any system that imports data needs to have a level of robustness to deal with a variety of data errors that the system typically expects to experience. There are common error classes for data importing that are typically catered for in systems where data import functionality is provided. The most common of these is checking that the import file is in the correct format. The next most common is that the data being imported contains the expected data types.

How well a system handles the data being imported is related to how much robustness the system needs to have to cope with unexpected data and it's permutations. I did not review any non-functional requirements so I cannot comment on what level of robustness the system was expected to have.

UAT is where the system is being tested to determine whether it is "fit for purpose". Part of being fit for purpose is whether the system handles the files it expects to import. The formats of the import files are usually specified as part of the requirements, often along with how to handle any expected errors. If the requirements do not define that the system should handle import file errors, then it would be typical to review whether the requirements were correctly specified or a requirements omission has been detected.

I have not seen any documentation on whether a data migration strategy was included in the project, but a project of this size, which is replacing a legacy systems should have had a data migration stream to prepare the data for migration to the new system. This would typically require a period of data cleansing to eliminate as many errors in the legacy data as possible. Once this has been run the data migration stream would normally run tests on that migration in preparation for migrating the data in the final production system.

Point (III)

Clarification was sought concerning point III of this question. The use of the words "generally incapable" was ambiguous as depending on the data storage mechanism employed, the management of child and parent relationships can vary considerably. The "Letter of Clarification, 3380_001, dated 21 May 2013" has clarified my request concerning point E, sub point III, with the following reframed question:

By this we ask that you assume a situation where:

1. **Test data has been loaded into the data stores / database being used by the relevant system (SAP or Workbrain);**
2. **That data was either:**
    a. **Loaded without referential or other (principally integrity) checks being used or enforced during the loading; or**
    b. **Loaded with checks and then subject to deletions or other modifications which were not subject to referential integrity or other (principally integrity) checks;**

    Such that system data was in a state that could not be input through the front-end.

In short, the two scenarios described at 2(a) and 2(b) ought be avoided. This is because of the high probability that these approaches can introduce data inconsistency errors into the test data. These will likely lead to the failures and false defects set out below.

Business Rules & Functionality

Whether considering SAP or Workbrain, either application utilises a (separate) database to store the data that the application processes. A database implements a data model that reflects the business rules on how the data is to be stored in the database.

Both SAP and Workbrain have specific business rules and functionality that determines where the data that the application receives is saved. Such rules and functionality would include:

- how the system performs data updates and data deletion in the database;
- the relationships between the data (stored in separate "tables" in the database);
- which of those data relationships need to be enforced by constraints (through "keys" inside a table, and through the use of "foreign keys" between tables) to ensure that the relationships maintain referential integrity. These concepts are explained below.

The definition of these relationships is usually defined via a data model.

Database Concepts

It is helpful to set out an explanation of the following database concepts to explain my answer. First, it is useful to explain how applications store data within a database and how that data is viewed, inserted, modified or deleted using Structured Query Language (SQL), which how Oracle databases would normally work. The following definitions are from The Practical SQL Handbook [6]:

1. A Primary Key is an "attribute (or group of attributes) that you can use to uniquely identify any row in the table". For example, every employee who works in an organisation has an employee number that is unique to that individual therefore the employee number can be used as the primary key as this value will always be unique.

2. A Foreign Key is a "column (in one table) that matches a primary key in a related table". The tables (entities) in a database can have relationships between each other. The foreign key can be used to cross-reference tables. For example, using our employee's table example from point 1, the employee's table contains a field called "Managed By". This field references the primary key column of a record in the Managers table. The primary key value stored in the "Managed by" column of the employee's table is a foreign key that points to a record in the managers table.

3. Referential Integrity is the "consistency among pieces of information that are repeated in more then one table". "It is vital that when information is changed in one place, it is also changed in every other place it appears".

An example of maintaining referential integrity is that a child record must contain a reference to the required parent record. If the child record does not contain the reference to the parent then we now have a database with data that is not consistent. Inconsistent data in the database could result in unexpected application behaviour, as this condition is something that the application considered it would never encounter.

The following are the kinds of data integrity that can be implemented by database systems to ensure the consistency of the data stored in the database:

1. Data Constraints - All database systems should guarantee that a value being stored in a field in a table of the database is of the correct data type, e.g. you should not be able to store a word or sentence in a field that only accepts numbers.

2. Domain Constraints – "is the set of logically related values from which the value in a particular column can be drawn" [6]. For example, the value of a field that stores the state of an Australian customer can only select from the values of valid Australian states. These types of domain constraints are often referred to as reference data.

3. Entity Integrity – "These require that no components of a primary key be allowed to have a null value. That is, a single-column primary key can't accept nulls, nor can any of the columns in a composite primary key" [6] (a composite primary key is where a number of fields are used to create the unique identifier for the row of data in a table) it cannot be empty.

4. Referential Integrity – "For each distinct non-null foreign key value in a relational database, there must exist a matching primary key value from the same domain". For example, consider the employees and manager example used to describe the definitions of a foreign and primary key. Since each employee has a manager, the employee's table must contain a foreign key that references to the primary key in the manager's table. These two tables now have a defined relationship between employee records and Manager records. Maintaining the relationship between the two tables ensures referential integrity.

Referential integrity is critical for managing data integrity within the database. If at any time the referential integrity is broken or not followed, the data integrity of the database is no longer assured. As data integrity declines, the higher the likelihood will be that the application will start experiencing errors or failures, leading to further declines in the data integrity of the database.

How a database and its application manage the integrity of its data depends on how it implements referential integrity. The following describes three ways that an application can implement and manage referential integrity:

1. The integrity is created and maintained at a database level by "cascading" the data update or delete to any tables that have a matching foreign key. For example, if the manager's id in the manager's table is changed, all records in the employee's table that reference that value must be updated. The database understands the relationship between the manager's table and the employee's table and will automatically "cascade" (update) the new manager ID value in the employee's table.

2. The application is designed with business logic that understands the relationships between the data it captures which is then stored in the database. When the application stores the data, it manually maintains the relationships between tables by adding the appropriate data values in the fields that require a reference to a record in another table. The relationship exists because the application knows about that relationship but it is not using the database to enforce the referential integrity by cascading changes through the database. The application is usually designed assuming that any data to be entered into the database is processed by the application's functionality and that the database data is not manipulated outside of the application. That is, the data is not manipulated directly at the database level, bypassing the application completely by the use of SQL statements or database importing mechanisms.

3. The application uses a combination of both method 1 and method 2 to implement data integrity using the database to manage some of the data referential integrity while the application manages other relationships through the applications business logic.

To define how data is related to each other, the applications use a technique called data modelling. The data model is verified by functional testing during system testing while for UAT, the data model is the embodiment of the business rules the system implements to support the required business processes. For example, with the Employee and Manager tables (in data modelling these are called entities), we can show how a data model is applied to describe a business rule that expresses the relationship between Employees and Managers. The data model can define a business rule between Managers and Employees, that is expressed, using the language of data modelling as, *"A manager can be responsible for managing multiple employees, but an employee can have one, and only one, manager"*.

UAT would look at the process of a new employee starting with the organisation and follow the process of creating the new employee in the system. One step in this process would be to define who their manager is.

Test Data

The next key aspect to understand in relation to the question in point 2 is how we approach the creation of test data for a test environment. There are a number of ways this can be done and some of these methods have higher chances of introducing invalid and incorrect data into the test database, which can then have undesirable outcomes when testing the application. The follow are the most common test data creation approaches employed in projects:

1. If the system is already in production, a copy of the production database is made into the test environment's database and the application uses this data to execute the required test cases.

2. If the system is new and no production system exists, then there are a number of options available to populate the database in the test environment:

   (a) If an old system (legacy) is being replaced by the new system, a Data Migration project is usually created to focus on migrating the legacy data correctly into the new system. The approach usually uses Structure Query Language (SQL) scripts to insert the data directly into the test database, but this approach should be subject to quite rigorous testing and validation to ensure the data is inserted correctly and doesn't cause data inconsistencies that subsequently cause the application to fail or behave in an unexpected manner.

   (b) Using the application (in this case SAP or Workbrain) to create the data required through its front end. Depending on the application complexity and the amount of data required, manually creating the data can be an incredibly time consuming process.. Automated test scripts are frequently used in this case and substantially speed up the process of creating the required test data.

   (c) Importing data via functionality provided by the application. Using the provided application functionality ensures that the data is saved into the correct tables and with the correct relationships as the system knows what to do with the data it is importing.

   (d) Importing data directly into the applications database using data import functionality provided by the database technology. This method should only be used in certain circumstance and used with caution. In simple systems with simple data models, the method maybe appropriate, but in complex systems with complex data models (complexity here is determined by the number of tables and the number of relationship that exist between these tables) this is not advisable. The risk of creating data that is completely inconsistent with what the application expects is very high in this approach. Testing must be performed to prove that the data is ending up in the right tables etc.

   (e) A more structured approach to loading test data directly into the application's database is to use SQL scripts, in an almost identical approach to that used by Data Migration projects. This approach gives you more control over where the data is inserted and the ability to create the relationships between the data that the application expects to see. To be successful with this approach, you must have a thorough understanding of the applications data model and database structure and every script must be verified to ensure that the SQL scripts insert the data into the correct places. You must also verify that the application works with the data created using the SQL scripts otherwise the system will exhibit unexpected behaviour as a result of data integrity errors. If this approach is used without verifying that the SQL scripts insert and modify that data as expected, it will significantly impact testing by causing unexpected behaviour of the application, increased number of defects being raised and test phases that take longer to complete due to the time required to investigate the increased numbers of defects.

Understanding how data is managed by and application and how test data is loaded into a test environment provides the background from which to answer the question posed in point 2, part (a). The data loading approach I have been asked to assume is similar to those I describe in points 2d and 2e immediately above. Using this approach to loading data into the database without any checks on whether it would insert the data into the correct tables or create the correct relationships once inserted, can lead to the test database becoming corrupted due to data errors and data inconsistencies.

If SQL scripts or data import mechanisms are used without any verification off their expected outcomes, you will corrupt the test data very quickly.

The outcome is an application that exhibits unexpected behaviour, crashes, or has major stability problems usually leads to questions concerning the level of quality that the developed solution is demonstrating. An application in this state would result in significantly higher defect numbers being raised then initially expected, which without root cause analysis would support the view that the solution hasn't reached the quality standard that was expected.

Point 2, part (b), of the clarified questions, describes a scenario in which data is deleted directly from the database and not by either the system's front end or via any other application specific functionality provided to remove one of more records from the database. Removing data from the database without using the applications functionality is very risky unless you have a thorough understanding of the application's data model.

If the relationship between all the data is not understood, the ramifications of deleting a record could result in the database records being corrupted. This will result in the same outcome as described above, i.e. an application that isn't behaving as expected.

In either case described in the clarified questions, there maybe appropriate reasons why you would perform such operations. This is usually as part of systems testing where we may create data inconsistencies to test an aspect of the system functionality, such as error messages being correct under particular conditions.

It should be noted that a primary assumption of UAT testing is that the database and the data should be as close to production like as is possible. Any manipulation of data directly via the database and not the application should be avoided during UAT. If the database is required to be manipulated as part of a business process, i.e. a work-around, then this operation should be performed using a SQL script/s which have previously been verified as working as expected before being applied to a UAT database. The integrity of the database is critical to the successful execution of UAT.

Executing a UAT phase with faulty test data would be extremely challenging due to the increased levels of unexpected application behaviour.

(f) Should User Acceptance Testing be conducted by reference to a Requirements Traceability Matrix? Why? What consequences might follow is a Requirements Traceability Matrix is not used?

All testing should be traced to some form of requirement. It is normal practice to trace UAT test cases to the business requirements, as defined by the V-model, as it is these requirements that define what the business expected the system to do.

I noted that while reading the various documents, there was no mention that UAT was driven by business requirements. The overall approach to UAT was driven by workshops with Subject Matter Experts (SME) who defined what business processes were to be included in the UAT phase. What I could not find in the UAT Test Completion Report I reviewed was how these business processes that were selected by the SME's for UAT execution mapped to the original business requirements that were specified during the scoping and planning stages of the project (subject to any changes that were agreed during the project).

The use of a traceability matrix enables the mapping of the business requirement defined in the scoping and planning phase of the project to the business processes selected by the SME's for execution during UAT. The traceability matrix is a key tool used in determining the impact of change on scope by identifying which test cases need to be modified or deleted and any new test cases that would need to be created.

The consequences of not having a traceability matrix are that you do not know what test cases cover which requirements and therefore how much of the system has been covered by the testing performed. The coverage aspect of the traceability matrix enables a more accurate assessment around what level of residual risk still exists in the system based on what the test execution covered. The traceability matrix would have enabled the UAT testing teams to manage change through the assessment of scope variations against the defined requirements.

Summary

A summary of the conclusions I have reached is as follows:

1. The signed off QHIC System Test and SIT Completion Report, dated 27 April 2009, shows that the software was accepted from these test phases with only Severity 3 and 4 defects outstanding and a rectification plan for these defects was agreed. The report also listed a number of change request in Section 3.2.6 Change Requests – "baseline 4" that were included in the system and system integration testing. This would suggest that the system was still experiencing requirements specification activities. It's not clear whether this level of requirements specification activities continued to occur into the UAT phase, but the chronology of project events in Dr Manfield's report does refer to a number of change requests, for example CR184, raised 26 June 2009 which indicates change was still occurring in June 2009. Regardless of what was changing, it is the responsibility of the test manager to ensure that they always understand what the current scope of the project is at any point in time.

2. Use of Severity is a key part of defect management and is used to triage the defect to enable the defect to be prioritised for fixing. Every defect should be investigated to determine what caused it. Defects raised which were caused by test data errors, environment errors and test script errors should also be tracked and captured for process improvement, but these defects should be distinguished and separated from defects which are found to be an error in application functionality. No test report reviewed contained any details of whether the defect numbers being reported were only functional application defects.

3. Given the QHIC Systems and SIT Completion Report, dated 27 April 2009, was signed off and meet the stated exit criteria, if the next phase of testing experienced higher numbers of defects than expected, this is not an automatic pointer that the previous test phases didn't deliver the level of quality expected in the solution. High numbers of defects should be a trigger to understand what is causing the number of defects being detected to be higher than expected. Unless you perform root cause analysis to determine what caused the defect, combined with defect leakage analysis, to determine which test phase the defect should have been detected in, you will not have sufficient data to substantiate any stated cause for the increased defect numbers, especially not a statement about the thoroughness of the previous test phase.

4. Root causes analysis of the defects would have provided insights into how many errors were introduced as a result of incorrect requirements. Requirements issues are a very common source of defects when the system scope is changing. While there is no specific data that reports the level of defects raised which were caused by requirements errors, in projects where change requests are being raised, throughout systems and SIT testing, frequently exhibit higher levels of requirements errors. Change requests being implemented while UAT is in progress is both an expensive impact to testing and increases the risk to project delivery.

5. Traceability Matrices are used in testing to trace which test scripts cover which requirements. It is extremely useful in determining the impact of any change request on testing by identifying what test scripts need to be reviewed against the changed requirements. It also can identify any new test cases that need to be created and what test cases can be marked as not required. It can be an effective way of managing scope and ensure that project management understand the impact a change request can have on the testing effort. All test scripts should be traced to a requirement to ensure that every requirement has been verified. In the V-Model, different requirements documents trace to different test phases. Business requirements define the core business processes the system must implement and these are traced to user acceptance tests that verify these business processes work correctly.

6. Data integrity impacts as a result of entering data directly into the database and not via the applications built functionality, i.e. the front end of the application, can be very destabilising for any application. This approach to test data creation is extremely error prone and depends on how well the database base model, data base structures and the application logic is understood when manipulating the data directly in the database. The impact of getting this wrong is an aborted test phase and subsequent delivery delays, while the test data is fixed. At worst, testing an application on the wrong test data significantly increases the risk that when the system "goes live", it actually fails when real data enters the application.

Declaration

(a) The factual matters stated in this report are, as far as I know, true;

(b) I have made all enquiries which I consider to be appropriate;

(c) The opinions stated in this report are genuinely held by me;

(d) The report contains reference to all matters I consider significant; and

(e) I have sought to act impartially in reporting on the issues the subject of this report.

I recognise that I have been requested to exercise an overriding duty in preparing this report, and any further report in the same matter, and in giving evidence, to the Commission rather than to the party or solicitors who commissioned me or pay my fees.

I have provided within my report:

(a) Details of my relevant qualifications;

(b) Details of any literature and other significant material that I have used in arriving at my opinions;

(c) Identified any person, and their qualifications, who has carried out any data selection, data inspection, tests or experiments upon which I have relied in compiling my report; and

(d) Details of any instructions that have affected the scope of my report.

I have used my best endeavours in my report, and will endeavour in any evidence that I am called to give:

(a) To confine myself to expressing opinions as an expert within those areas in which I am specially knowledgeable by reason of my skill, training or experience;

 (i) To distinguish among the data upon which I have relied, the assumptions which I have made, the methods that I have employed, and the opinions at which I have arrived;

 (ii) To indicate those dates, assumptions and methods upon which I have significantly relied to arrive at my opinions;

 (iii) To give succinct reasons for each of the opinions which I express;

 (iv) To be objective and unbiased;

(v) To make the opinions which I express clear, comprehensible and accessible to those not expert in my discipline;

(vi) To be scrupulous in terms of accuracy and care in relation to the data upon which I rely, my choice of methods, and the opinions which I express arising from those data;

(vii) To indicate whether I have been provided with all the data necessary for me to arrive at the views which I have expressed and whether I need further information;

(viii) To indicate whether I have been apprised of any data or choice of method which might entail opinions which are inconsistent with the opinions which I have expressed; and

(ix) To indicate whether I have been unable for any reason to employ the methodology that I would prefer to use before expressing an opinion.

If I become aware of any error or any data which impact significantly upon the accuracy of my report, or the evidence that I give, prior to the legal dispute being finally resolved, I shall use my best endeavours to notify those who commissioned my report or called me to give evidence.

I shall use my best endeavours in giving evidence to ensure that my opinions and the data upon which they are based are not misunderstood or misinterpreted by the Commission.

I have not entered into any arrangement that makes the fees to which I am entitled dependent upon the views I express or the outcome of the case in which my report is used or in which I give evidence.

Yours faithfully

[illegible]

Shane Parkinson

Integrity Software Testing

RESUME: SHANE PARKINSON

20 July 2011

59/95 Euston Road
Alexandria, NSW, 2015

+614 39318353

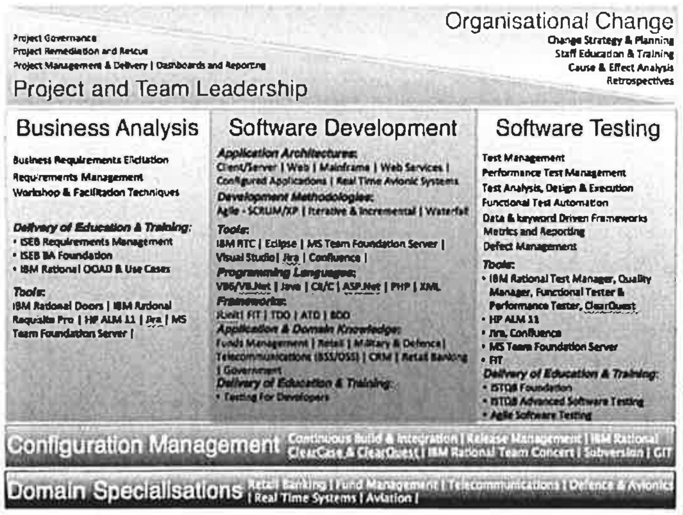
shaneparkinson@me.com

CAREER OBJECTIVE

To work with an organisation with vision, foresight and leadership that embraces, enacts and enables collaborative, team-based behaviours, while supporting and encouraging staff to explore a culture of learning.

CAREER SUMMARY

- **25 years experience in Software Development** across Waterfall, iterative & incremental specifically AGILE (SCRUM, XP) & RUP
- **Experience in Defence, Finance, Telecommunications and Australian Government industry sectors**
- **15 years experience in Test Management** including Performance Test Manager, Test Manager and Test Director roles defining strategy, manager delivery of software testing activities and test execution
- **Consulting roles with Capgemini, Planit Test Management Solutions, & Rational Software covering the following engagements**:
  - **Software development process assessments** (for both Agile and waterfall) & facilitating the implementation of recommendations
  - **Organisational change management of process improvements** encompassing staff mentoring, training and customised course development to support these change initiatives
  - **Software tools assessments** detailing recommendations for improvement (or change)
  - **Development of software tools and strategies** for Tools Selection and implementation including the management of the implementation
  - **Implementation of performance and test automation processes** and practices
  - **Test Environment Management** (including configuration and release), problem investigation and resolution
  - **Business Analysis training and assessments of business analysis practices** including working with teams on agile approach to requirements definition
- **Training course Development and Delivery** in:
  - ISTQBFoundation and Advanced in Software Testing
  - ISEB Business Analysis Foundation & Requirements Engineering,
  - Agile Software Development
- **Roles Performed:**
  - [Current role] Director - Testing Services - Capgemini Australia
  - Director of Technical Testing Services - Planit
  - Software Trainer - Planit
  - Test Environment Manager - PlanIt
  - Software Tester - Planit
  - Software Developer,
  - Test Automation and Performance Testing Engineer,
  - Software Process Definition and Testing Practices Lead,
  - Configuration Manager
- **Other career achievements include**:
  - Co-author of the syllabus and course for the **Certified Agile Tester** (CAT)
  - **Presented at ANZTB conference 2010**:
    - **Tutorial**:- Agile Testing
    - **Paper**:- "Who makes the better tester, BA or a Tester"
  - **Presented at the Business Analysts Master Class Auckland 2010**:
    - **Paper**:- "Who makes the better tester, BA or a Tester"
  - **Keynote presenter at the ANZTB conference 2011**:
    - **Paper**:- "All we need to do is define the process, then magic happens"
  - **International Presenter at Business Analyst Master Class Auckland 2011:**
    - **Paper**:- "Are We Just Blinded by the Shiny New Toys of Epics and User Stories"
    - **Paper**:- "Buyers beware! Things you need to know before you buy a requirements management tool"



Organisational Change
Change Strategy & Planning
Staff Education & Training
Cause & Effect Analysis
Retrospectives

Project Governance
Project Remediation and Rescue
Project Management & Delivery | Dashboards and Reporting
Project and Team Leadership

Business Analysis
Business Requirements Elicitation
Requirements Management
Workshop & Facilitation Techniques
Delivery of Education & Training:
• ISEB Requirements Management
• ISEB BA Foundation
• IBM Rational OOAD & Use Cases
Tools:
IBM Rational Doors | IBM Rational Requisite Pro | HP ALM 11 | Jira | MS Team Foundation Server |

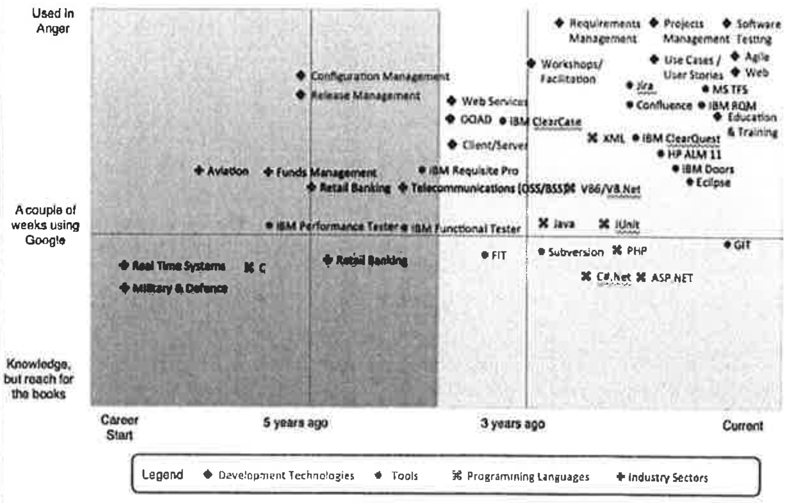
Software Development
Application Architectures:
Client/Server | Web | Mainframe | Web Services | Configured Applications | Real Time Avionic Systems
Development Methodologies:
Agile - SCRUM/XP | Iterative & Incremental | Waterfall
Tools:
IBM RTC | Eclipse | MS Team Foundation Server | Visual Studio | Jira | Confluence |
Programming Languages:
VB6/VB.Net | Java | C#/C | ASP.Net | PHP | XML
Frameworks:
Xunit | FIT | TDD | ATD | BDD
Application & Domain Knowledge:
Funds Management | Retail | Military & Defence | Telecommunications (BSS/OSS) | CRM | Retail Banking | Government
Delivery of Education & Training:
• Testing For Developers

Software Testing
Test Management
Performance Test Management
Test Analysis, Design & Execution
Functional Test Automation
Data & keyword Driven Frameworks
Metrics and Reporting
Defect Management
Tools:
• IBM Rational Test Manager, Quality Manager, Functional Tester & Performance Tester, ClearQuest
• HP ALM 11
• Jira, Confluence
• MS Team Foundation Server
• FIT
Delivery of Education & Training:
• ISTQB Foundation
• ISTQB Advanced Software Testing
• Agile Software Testing

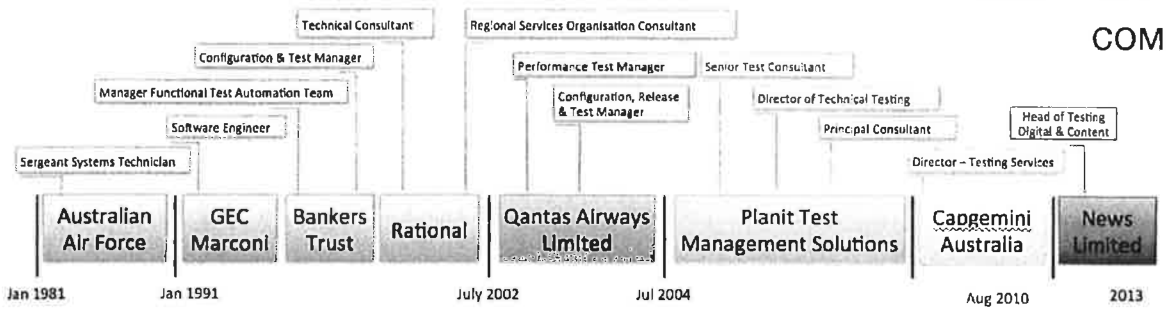
Configuration Management Continuous Build & Integration | Release Management | IBM Rational ClearCase & ClearQuest | IBM Rational Team Concert | Subversion | GIT

Domain Specialisations Retail Banking | Fund Management | Telecommunications | Defence & Avionics | Real Time Systems | Aviation |

SKILLS



COMPETENCIES


Technical Consultant
Regional Services Organisation Consultant
Configuration & Test Manager
Performance Test Manager
Senior Test Consultant
Manager Functional Test Automation Team
Configuration, Release & Test Manager
Director of Technical Testing
Software Engineer
Principal Consultant
Head of Testing Digital & Content
Sergeant Systems Technician
Director – Testing Services
Australian Air Force
GEC Marconi
Bankers Trust
Rational
Qantas Airways Limited
Planit Test Management Solutions
Capgemini Australia
News Limited
Jan 1981
Jan 1991
July 2002
Jul 2004
Aug 2010
2013

EXPERIENCE

Head of Testing - Digital and Content Jan 2012 to Current

News Limited Australia

The role is to set strategy and direction for the testing of the Digital aspects of the News Limited presence with the responsibility to ensure software testing practices and processes align with the business outcomes for the delivery of the News Limited Digital Strategy.

The role's delivery component was focused on providing:

- Delivery of testing outcomes aligned with business goals and objectives
- Management of the Testing team consisting of both full time testers and contractors
- Ensuring consist testing approaches applied across all projects
- Ensuring all testing staff are skilled and component in delivering testing outcomes across the current SDLC changes introducing agile software development practices and processes to the Digital and Content department.
- Representing testing at steering committee and Programme management meetings
- Reporting progress and current software quality based on Defect data, Story completion, burn up chats and mapping of trends
- Introduction of improved automated testing strategies through the combination of tester developer skills and building in functional test automation as part of the continuous integration practices and processes

Performance Test Manager - News Digital Media Aug 2011 to Jan 2012

News Limited Australia

The role of Performance Testing Manager was part of the Paid Content Programme which delivered a subscription based model for The Australian and the Herald Sun mastheads.

I was asked to come on board to remediate the performance testing strategy and approach and to align all key stakeholders to gain agreement on the approach and risks being mitigated by the strategy. At the time of joining, significant issues existed with the lack of agreement on the non functional requirements and the discordant perspectives held by the key stakeholders.

The performance testing strategy and the non functional requirements were redefined and the focus placed on the risks that needed to be mitigated by aligning the stakeholders on risks, subsequently redefining the strategy and approach for delivery of the performance testing of the solution.

The project performance test strategy was delivered and an overall new performance testing regime was developed which tested continually and iteratively across all major systems. This change has now resulting in up front engagement of the performance testing group and has resulted in a significantly improved focus on mitigating risks associated with applications and systems performance, early in each projects development cycles

EXPERIENCE

Director - Testing Service — Aug 2010 to Current

Capgemini Australia

The role was newly created to give the Australian Testing Services division a local thought leadership focus to leverage the global Capgemini and Sogeti thought leadership. It was to also focus on innovation and global trends in software testing.

The role was to be the enablement channel for internal staff, as well as Capgemini's clients to broaden the adoption, implementation and improvement of testing practices, skills and processes, through knowledge management, training and skills and competency career frameworks.

The role's delivery component was focused on providing:

- Organisational Test Process Improvement (TPI) reviews
- Testing Tools current state reviews
- Implementation and support of Testing Centre's of Excellence (TCoE) for clients
- Supporting the Testing Services business through:
  - Providing pre and post sales support
  - Creating RFP/RFQ responses
  - Development of Client proposals to match specific testing solutions to custom needs
- Supporting Test Managers on site
- Remedial Testing project support and action where required to rectify testing issues
- Test Strategy and planning
- On site staff mentoring and training in testing practices and testing tool usage
- Education and Training consisting of:
  - In house training and presentations
  - Delivery of ISTQB Training/Mentor for certification
  - Test management Approach (TMAP) and (TPI) training

Engagement within Capgemini

Test Strategy Development

BIO Security Transformation Program, Livestock Health Management System, the NSW Department of Industry and Investment.

Development of the test strategy Considerations for the BIO Security information transformation programme and then subsequently the completion of the Test Strategy for the Live Stock Health Management System, (LHMS). This has also flowed on to the creation of the high-level Acceptance testing objectives and Test Objectives Matrix to enable the future enactment of the testing strategy and usage of the high-level test objectives to guide user acceptance testing.

Test Process Improvement (TPI) Assessment

Telstra - Architecture, Online and Media

Test Process Improvement (TPI) review focused on testing, but covering a holistic, end to end view which encompassed review development and business analysis processes and products and the interrelated relationships between their practices and processes and subsequent influence on testing approaches and practices

Testing Tools Review

QLD Department of Education and Training (DET)

Review the implementation, adoption and usage of the selected Tools Suite across testing, business analysis and development. The tools discussed for the scope of the review were MS Team Foundation Server, IBM Rational Doors and SpiraTest, but additional tools were found and these factored into the review. The primary focus was on impediments to usage and adoption.

Testing Case Analysis and Design

Fosters

The review was requested as a result of a client audit and was focused on the level of detail and coverage against requirements of the test cases created for systems testing.

EXPERIENCE

Principal Consultant Jul 2004 to Aug 2010

Planit Test Management

Held three roles during the 6 years with the organisation:

- **Principal Consultant**
- **Director Technical Testing Services**
- **Senior Test Consultant**

Responsible for:

- **R&D Focus to Navigate the Business for Future Trends** by creating plans for staffing and education development to prepare for emerging trends in software development and testing, e.g. Agile
- **Pre and Post Sales Support** to the sale team
- **Account Management of Client and Staff** engaged on client sites
- **Focused on Delivery** to ensure success in client outcomes across software testing and tools engagements
- **Education and Training** involving the development, maintenance and delivery of all courses on the Planit public training schedule.
- **Developing and Delivery Client In-House Training Courses** which included customisation of courses for specific change initiatives, skills uplift and capability enabling

Highlights of working with the training division include:

- **The successfully implementation of the ISEB Business Analysis programme within Australia** (the first such programme to bring ISEB BA courses to Australia)
- **Acted as the Planit Business Analysis liaison with the Australian Institute of Business Analysts (AIBA)** in the development of training to support the AIBA's initiatives
- **I successfully developed and implemented the Planit Agile Training workshop** to introduce agile concepts and practices to software teams
- **Instrumental in the development of the Syllabus for the Certified Agile Tester (CAT)** which was based on the Agile Workshop format or practical, experiential approaches to learning. The course is now adopted globally for Agile testing certification
- **Engaged to Develop Client Specific Training Material to Support Organisations Change Initiatives**

A broad rage of consulting engagements with clients across the 6 year period:

- **Implementation of the IBM Rational Team Concert Agile ALM tools solutions**, including staff training and client relationship management
- **Working with Teams on Implementing Agile Practices**, mentoring and applying scrum master techniques to facilitate the team's success
- **Development of Test Strategies** for client software development projects, aligned to product and project risks, project goals and business and stakeholder objectives and the introduction of earlier engagement of the testing team
- **Definition and Management of Test Environments** through alignment with configuration management and release practices as well as alignment with testing processes and practices
- **Review and Recommendations for the Implementation of Test Automation Strategies** as well as capturing and defining the tool requirements for Test Management, Functional and Performance test automation tools, Configuration and Release Management and tools that support Agile software development and Collaboration
- **Test Management of Large System Integration Projects** - Multiple vendors across multiple test phases including Performance, Systems, Systems Integration and User Acceptance Testing with a team of over 20 testing staff, business users and vendor management
- **Review of the Current State of Client's SDLC.** These reviews result in a set of improvement recommendations which often cover all the SDLC due to the inherit coupling of SDLC components using a holistic approach that encourages "quick wins", but without detriment to the longer-term objectives and goals. These engagements have included major organisational change initiatives to enable successful implementation of the recommendations

Development Staff Skills and Competency Matrix to improve the matching of Client's needs to staff which resulted in the development of a series of in-house technical training and assessments programmes to objectively assess the skill levels of staff in key technology areas

SKILLS

Software Testing Skills

All Aspects of Software Testing:

- Test management and Leadership
- Test Strategy and Planning
- Tester Education and Training
- Test Analysis, Design and Execution
- Metrics and Reporting
- Estimation and Planning
- Project Management competencies as related to test management

Testing Tools:

- HP ALM 11, QC10, 9
- IBM Rational RQM, Test Manager
- IBM Rational ClearQuest
- MS Team Foundation Server

Development Skills

Coding skills cover:

- VB 6/VB.net/C#/ASP.Net
- Java
- PHP
- TDD/BDD

Configuration & Release Management:

- ClearCase
- SVN
- Continuous Build and Integration Tools (Hudson, IBM Rational RTC)
- Atlassian Jira (inlcuding GreenHopper), Confluence and Crowd
- IBM Rational ClearQuest
-

Development Skills

Business Requirements elicitation techniques of:

- Workshops (various techniques)
- Interviews
- Questionnaires

Requirements Documentation Techniques:

- Use of Tools for Requirements Captures
- Use Cases
- User Stories

Requirements Management and Modelling tools:

- IBM Rational Doors
- IBM Rational Requisite Pro
- IBM Rational Rose
- MS Visio
- MS Word

Technologies

J2EE
MS.NET
Web
SOA, REST services
MS SQL
MYSQL
MS Access
PHP Web and Open Source Web based tools of Joomla, Wordpress, Concrete5
ZEND Technologies

SDLC

Covered

- Waterfall
- V-Model
- Iterative and Incremental
- Agile, including:
  - SCRUM
  - XP
  - LEAN

Domain

Financial:

- Fund Management
- General Deposits and Banking
- Financial Advisers

Telecommunications:

- Billing and Rating
- Client/Customer systems
- Provisioning

Defence:

- Avionic Real-time Systems

Government:

- Education and Training
- Primary Industry's
- Australian Bureau of Statistics

Aviation:

- Booking Systems
- Membership Programs
- Freight

EDUCATION

ROYAL AUSTRALIAN AIR FORCE TRAINING

Royal Australian Air Force Instrument Fitters Apprenticeship

Royal Melbourne Institute of Technology (RMIT) in Aircraft Engineering

RATIONAL TRAINING COURSES (ISEB)

- Requirements Management and Use Cases
- ClearCase Administration
- ClearQuest Administration
- Object Orientated Design and Analysis
- Performance and Functional Testing

INFORMATION SOFTWARE EDUCATION BOARD (ISEB

- Software Testing Foundation Certification
- Software Testing Practitioner Certification
- Business Analysis Foundation Certification
- Requirements Engineering from Business Analyst Diploma

AUSTRALIAN INSTITUTE OF MANAGEMENT

Developing Strategies for Change

PLANIT TEST MANAGEMENT

International Software Testing Qualification Board (ISTQB) Foundation Certification

Creating Quality Requirements

Practical Test Management

Agile Software Development

Our ref: RJT\IRI\02 3001 0594

Partner: Robert Todd
Direct line: +61 2 9258 6082
Email: robert.todd@ashurst.com
Contact: Ian Innes, Special Counsel
Direct line: +61 7 3259 7142
Email: ian.innes@ashurst.com

Ashurst Australia
Level 38, Riverside Centre
123 Eagle Street
Brisbane QLD 4000
Australia

GPO Box 9938
Brisbane QLD 4001
Australia

Tel +61 7 3259 7000
Fax +61 7 3259 7111
DX 388 Brisbane
www.ashurst.com

17 May 2013

BY EMAIL

Mr Shane Parkinson
Integrity Software Testing

Dear Mr Parkinson

Queensland Health Payroll Commission of Inquiry

We act for IBM Australia Limited (**IBM**).

1. Retainer

The Honourable Mr Richard Chesterman has been appointed under the *Commissions of Inquiry Act 1950* (Q) to investigate matters related to a contract between IBM and the State of Queensland. A focus of the Commission arises out of the design, build and implementation of a finance and payroll system for Queensland Health.

A copy of the Commission's terms of reference is attached.

You are retained to prepare a report in respect of matters before the Commission. You may be asked to give oral evidence in relation to this report by the Commission.

2. Specific Questions

The questions which we would like you to address are as follows.

(a) As a testing manager, would one need to read and understand the system scoping documents to ensure that test scripts were developed and run against the scoped system?

(b) Assume that a high number of defects are identified during User Acceptance Testing on a highly complex system (such as the payroll system).

(i) Is that necessarily an indicator of poor, or inadequate Systems and Systems Integration Testing?

(ii) Are high numbers of defects common in highly complex systems?

(iii) Would more information than a severity indicator be required?

225048702.01

(c) Assume that:

(i) System and Systems Integration Testing was conducted, audited and accepted in the manner set out in the System and System Integration Testing Report dated 27 April 2009;

(ii) You have not personally reviewed the System and Systems Integration test scripts or processes;

(iii) User Acceptance Testing was not conducted (by the UAT Test Manager or individual testers) by reference to a Requirements Traceability Matrix;

(iv) a high number of severity 2 "functional" defects (for instance, the test returns an error or the system does not complete that process) are discovered during a final phase of UAT,

would you consider it appropriate or safe to reach the conclusions expressed in the K J Ross Test Completion Report dated 27 January 2010 (at p 20) that "system testing was not thorough enough" and "integration testing was not thorough enough".

(c) On the assumptions expressed above, what conclusions could you reach about the possible causes of the high number of defects?

(d) Assume further that data imported into a proposed system in the course of UAT contains errors of a kind not normally encountered in production.

(i) What impact could that have upon test results?

(ii) Should a system ordinarily be sufficiently robust during UAT that it ought to cope with any bad data which is so imported?

(iii) What if such data is generally *incapable* of being encountered in production (for example, 'child' records without a parent, where such a record could not be added without a 'parent' record, etc).

(e) Should User Acceptance Testing be conducted by reference to a Requirements Traceability Matrix? Why? What consequences might follow is a Requirements Traceability Matrix is not used?

3. Your report

The Commission is not a Court, but it operates in a manner similar to a Court.

To assist you in understanding the function IBM seeks to have your perform we have however extracted the relevant court rules which would apply were this report to be provided as evidence before a Court. You should use these as a guideline to assist you in understanding the role we wish to have you perform.

As you would be aware, an expert's primary duty when giving evidence to a court is to assist the court. We ask that you take the same approach in providing your report. When you are preparing the report, we ask that you please address the following.

(a) The report should be addressed to the Queensland Health Payroll Commission of Inquiry. The report itself is nevertheless to be forwarded to us, and not to the Commission;

(b) Please attach your curriculum vitae to the report;

(c) If you rely on any documents in the course of preparing your report, please list them in the body of the report, (excluding this letter of retainer unless you consider it necessary). We would prefer, if you consider that you need to refer to an assumption in the course of your report that this was stated in your report, rather than incorporated by reference to this letter. You do not need to list any documents you have <u>not</u> relied on;

(d) The report should include a statement at the end to the following effect:

  (i) the factual matters stated in the report are, as far as you are aware to the best of your knowledge, true;

  (ii) you have made all enquiries you consider appropriate;

  (iii) the opinions stated in the report are genuinely held by you;

  (iv) the report contains reference to all matters you consider significant; and

  (v) you have sought to act impartially in reporting on the issues the subject of the report.

In addressing those matters you may wish to give consideration to making statements, where appropriate, in similar terms to the following at some point in the body of the report.

I recognise that I have been requested to exercise an overriding duty in preparing this report, and any further report in the same matter, and in giving evidence, to the Commission rather than to the party or solicitors who commissioned me or pay my fees.

I have provided within my report:

(d) details of my relevant qualifications;

(e) details of any literature and other significant material that I have used in arriving at my opinions;

(f) identified any person, and their qualifications, who has carried out any data selection, data inspection, tests or experiments upon which I have relied in compiling my report; and

(g) details of any instructions which have effected the scope of my report.

I have used my best endeavours in my report, and will endeavour in any evidence which I am called to give:

(a) to confine myself to expressing opinions as an expert within those areas in which I am specially knowledgeable by reason of my skill, training or experience;

(b) to distinguish among the data upon which I have relied, the assumptions which I have made, the methods that I have employed, and the opinions at which I have arrived;

(c) to indicate those dates, assumptions and methods upon which I have significantly relied to arrive at my opinions;

(d) to give succinct reasons for each of the opinions which I express;

(e) *to be objective and unbiased;*

(f) *to make the opinions which I express clear, comprehensible and accessible to those not expert in my discipline;*

(g) *to be scrupulous in terms of accuracy and care in relation to the data upon which I rely, my choice of methods, and the opinions which I express arising from those data;*

(h) *to indicate whether I have been provided with all the data necessary for me to arrive at the views which I have expressed and whether I need further information;*

(i) *to indicate whether I have been apprised of any data or choice of method which might entail opinions which are inconsistent with the opinions which I have expressed; and*

(j) *to indicate whether I have been unable for any reason to employ the methodology which I would prefer to use before expressing an opinion.*

If I become aware of any error or any data which impact significantly upon the accuracy of my report, or the evidence that I give, prior to the legal dispute being finally resolved, I shall use my best endeavours to notify those who commissioned my report or called me to give evidence.

I shall use my best endeavours in giving evidence to ensure that my opinions and the data upon which they are based are not misunderstood or misinterpreted by the Commission.

I have not entered into any arrangement which makes the fees to which I am entitled dependent upon the views I express or the outcome of the case in which my report is used or in which I give evidence.

4. Confidentiality

We ask that the report be kept strictly confidential as it is to be used for the purpose of obtaining legal advice or for use in legal proceedings. You are only authorised to provide this letter or your report to representatives of the IBM or ourselves, and not to any other person or party.

We would encourage you to contact us to discuss any of the matters raised above before finalising your report.

If you have any queries with respect to this, please contact Ian Innes of our office on (07) 3259 7142.

Yours faithfully

Ashurst Australia

ashurst

EXPERT EVIDENCE – UNIFORM CIVIL PROCEDURE RULES

426 Duty of expert

(1) A witness giving evidence in a proceeding as an expert has a duty to assist the court.

(2) The duty overrides any obligation the witness may have to any party to the proceeding or to any person who is liable for the expert's fee or expenses.

428 Requirements for report

(1) An expert's report must be addressed to the court and signed by the expert.

(2) The report must include the following information—

- (a) the expert's qualifications;
- (b) all material facts, whether written or oral, on which the report is based;
- (c) references to any literature or other material relied on by the expert to prepare the report;
- (d) for any inspection, examination or experiment conducted, initiated, or relied on by the expert to prepare the report—
  - (i) a description of what was done; and
  - (ii) whether the inspection, examination or experiment was done by the expert or under the expert's supervision; and
  - (iii) the name and qualifications of any other person involved; and
  - (iv) the result;
- (e) if there is a range of opinion on matters dealt with in the report, a summary of the range of opinion, and the reasons why the expert adopted a particular opinion;
- (f) a summary of the conclusions reached by the expert;
- (g) a statement about whether access to any readily ascertainable additional facts would assist the expert in reaching a more reliable conclusion.

(3) The expert must confirm, at the end of the report—

- (a) the factual matters stated in the report are, as far as the expert knows, true; and
- (b) the expert has made all enquiries considered appropriate; and
- (c) the opinions stated in the report are genuinely held by the expert; and
- (d) the report contains reference to all matters the expert considers significant; and
- (e) the expert understands the expert's duty to the court and has complied with the duty.

Queensland Health Payroll System Commission of Inquiry

Terms of reference

COMMISSIONS OF INQUIRY ORDER (NO. 2) 2012

Short title

1. This Order in Council may be cited as the *Commissions of Inquiry Order (No. 2) 2012*.

Commencement

2. This Order in Council commences on 1 February 2013.

Appointment of Commission

3. UNDER the provisions of the *Commissions of Inquiry Act 1950* the Governor in Council hereby appoints the Honourable Richard Chesterman AO RFD QC from 1 February 2013, to make full and careful inquiry, in an open and independent manner, into the implementation of the Queensland Health payroll system with respect to the following matters, and having regard to previous reviews of the Queensland Health payroll system implementation, including the KPMG implementation review and the Auditor-General of Queensland's report titled *Information systems governance and control, including the Queensland Health Implementation of Continuity Project* (2010):
   a. the adequacy and integrity of the procurement, contract management, project management, governance and implementation process;
   b. whether any laws, contractual provisions, codes of conduct or other government standards may have been breached during the procurement and/or implementation process and who may be accountable;
   c. the contractual arrangements between the State of Queensland and IBM Australia Ltd and why and to what extent the contract price for the Queensland Health payroll system increased over time;
   d. any recommended changes to existing procurement, contract and project management (including governance) policies, processes, standards and contractual arrangements for major Queensland government information and communication technology projects initiated in the future to ensure the delivery of high quality and cost effective products and systems; and
   e. any other matter relevant to this review.

Commission to report

4. AND directs that the Commissioner make full and faithful report and recommendations on the aforesaid subject matter of inquiry, and transmit the same to the Honourable the Premier by 30 April 2013.

Application of Act

5. THE provisions of the *Commissions of Inquiry Act 1950* shall be applicable for the purposes of this inquiry except for section 19C – Authority to use listening devices.

Conduct of Inquiry

6. THE Commissioner may hold public and private hearings in such a manner and in such locations as may be necessary and convenient.

ENDNOTES

1. Made by the Governor in Council on 13 December 2012.

2. Notified in the Gazette on 14 December 2012.
3. Not required to be laid before the Legislative Assembly.
4. The administering agency is the Department of Justice and Attorney-General.

Our ref: RJT\IRI\02 3001 0594

Partner: Robert Todd
Direct line: +61 2 9258 6082
Email: robert.todd@ashurst.com
Contact: Ian Innes, Special Counsel
Direct line: +61 7 3259 7142
Email: ian.innes@ashurst.com

Ashurst Australia
Level 38, Riverside Centre
123 Eagle Street
Brisbane QLD 4000
Australia

GPO Box 9938
Brisbane QLD 4001
Australia

Tel +61 7 3259 7000
Fax +61 7 3259 7111
DX 388 Brisbane
www.ashurst.com

21 May 2013

BY EMAIL

Mr Shane Parkinson
Integrity Software Testing

ashurst

Dear Mr Parkinson

Queensland Health Payroll Commission of Inquiry - Request for Clarification

You have asked for clarification of the type of situation we refer to in our original letter of date 17 May 2013, wherein we asked:

> "(d) Assume further that data imported into a proposed system in the course of UAT contains errors of a kind not normally encountered in production.
>
> ...
>
> (iii) What if such data is generally *incapable* of being encountered in production (for example, 'child' records without a parent, where such a record could not be added without a 'parent' record, etc)."

By this, we ask you to assume a situation where:

1. Test data has been loaded into the data stores / database being used by the relevant system (SAP or Workbrain);
2. That data was either:
   a. loaded without referential or other (principally integrity) checks being used or enforced during the loading; or
   b. loaded with checks and then subject to deletions or other modifications which were not subject to referential or other (principally integrity) checks;

   such that system data was in a state that could not be input through the front-end.

We trust that this clarifies question (d)(iii). If you have any queries with respect to this, please contact Ian Innes of our office on (07) 3259 7142.

Yours faithfully

Ashurst Australia