
FUTURES IN SOFTWARE TESTING

How to overcome the information deficit and build careers in software testing

INTRODUCTION

30% of people who said they were considering leaving software testing were doing so because of a perceived lack of information around where they can go in their career and how to get there. This is a disconcerting statistic even if it were only one person leaving the industry.

Do we have an information deficit and is it impacting our ability to develop successful careers in software testing? In my software testing career I have collected anecdotal evidence to suggest this is the case. In order to find out what the testing community felt I ran a survey¹ and asked software testers to contribute. I was only able to get a small sample of the community but two key demographics ended up responding:

- Those in online software testing communities and forums
- Those in the industry who do not participate in online communities but I was able to reach via a network of people. These tended to be localised to the Canberra, Australia region which has a high concentration of testers due to government work.

The research showed two things:

- 75% of testers said they were unaware of the different directions a tester could go in their career
- 64% of testers said that they did not have all the information they require to develop the capabilities required for their desired career path.

This suggests that the anecdotal evidence I have been given is true and that those who are leaving testing due to a perceived lack of information are justified.

This paper proposes a self assessment model for software testing education and career development to resolve these issues.

A LACK OF DIRECTION

At present there are two predominate certification programs within the world of Software Testing; ISTQB and CSTE. Both of these programs provide a graduated way of developing software testing education. Both of these certifications are focussed on defining processes for the act of software testing. The Association of Software Testing also contains no resources² regarding establishing a career in software testing

None of the above organisations define what is a software tester? Or, what a tester is responsible for testing given a system? I can understand how a new software tester can feel lost and overwhelmed by the lack of clarity, definition and information necessary to determine areas of interest and identify what they need to learn. I have an email from a tester who is currently following the proposal I will introduce during this paper and she reports on how she struggled to define what she did as a job.

How can we create a future if we can't even describe what we do now?

¹ The appendix contains the survey questions and a summary of the results

² <http://resources.associationforsoftwaretesting.org/>

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

One of the survey questions was “have you undertaken any formal software testing training or certification program?” and this was followed up with “Since completing your formal software testing training or certification program do you have an increased understanding of the available directions of software testing?”.

The majority (71%) felt that having completed the training they had an increased understanding but we still have 64% of respondents feeling as though they don’t have enough information to develop their capabilities.

SIMPLISTIC MODELS FOR ADVANCEMENT

Currently the software testing profession; or the organisations that hire software testers are using vague notions of competency like Junior and Senior. In some organisations³ these are tied to a rank within the organisation rather than the any measure of actual ability. Therefore if the tester interviews for a position that is deemed Senior and is successful then that person is now a Senior Tester. This process is insufficient for gauging ability as it is a subjective measurement that is defined by the organisation and its needs rather than any level of capability as defined by the profession or the tester.

We need to move beyond junior and senior for indicating ability as they are not tied to any real measure.

In order to overcome these issues we need a structure that provides some clarity about what we do when we test and we need a structure that allows us to realistically assess our capabilities

Learning testing has complications that do not make it easier. When we are responsible for testing a product there are many aspects of the system that we should be checking, testing and validating. Some of these are quantifiable whilst others are qualitative. The new tester doesn’t get the luxury of only testing a subset of the system. The new tester has to test everything from day one and do it well.

SOFTWARE TESTING DISCIPLINES

It is my belief that it is not possible for any one person to be an expert on the different types of testing that are performed against a system under test and to have the time necessary to design and execute all of the test cases. Traditionally we have used system and automation as career development pathways as well as test management and performance testing. This suggests that the point where one person could cover all aspects of testing have long past if they ever existed.

Now that we know we must have directions for specialisation within software testing, we need to have robust definitions of each specialisation. What does automation mean and what is defined by system testing? The ISTQB definition of what is performed on during system testing encompasses functional testing and non-functional testing a wholly exhaustive definition of everything but the definition goes only a little further to suggest a subset of the types of non-functional testing to be undertaken⁴.

We have already seen that more than 70% of testers surveyed felt that when they started testing they were unaware of the different directions they could take. This means that we need to have a set of software testing disciplines and that they need to be readily available.

After considering what testing should be completed to thoroughly test a software system, without the constraints of timeframes, I came up with the following list of eight disciplines.

- Behaviour & Functionality
- User Interaction

³ The Australian Federal Police is one

⁴ ISTQB V1.04 Delegate Manual; section 2 page 20

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

- Management
- Business Domain Knowledge
- Performance
- Security
- Automation
- Infrastructure & Integration

What I want is that when a new tester begins their career they are given information regarding the disciplines of software testing and can identify the areas that interest them. With this information in hand they can work towards becoming an expert in those disciplines. How the new tester goes about this approach I will discuss later as now I will explain each of the disciplines and what they mean.

SCOPE

Each of these disciplines defines a testing specific set of concepts, responsibilities, techniques, artefacts and aspects.

- **Concepts** - theoretical constructs that are directly relevant to the act of testing
- **Responsibilities** - what the tester is responsible for
- **Techniques** - approaches to solving problems or achieving a task
- **Artefacts** – what is produced by the tester
- **Aspects** – testing foci of a discipline

The definition, once is has been comprised of the above states what **can** be done by a practitioner of the discipline. This provides clarity to the tester and a clear map of what they need to learn to achieve their career goals within software testing. The discipline definition does not cover **how** testing is to be done. This is process and implementation dependant so we leave this for context to define and relevant training courses to provide.

Any roles and responsibility in and around the testing effort such as Configuration Management, Environment Management or Team Administration are not covered here. The need for these positions in each organisation is different and is outside the scope of this paper.

TECHNICAL VS. NON-TECHNICAL

Four disciplines; behaviour & functionality, user interaction, management and business domain knowledge, are non-technical whilst the remaining four; performance, security, automation and infrastructure & integration, are technical. When I say technical I mean that the tester requires more in depth knowledge of the implementation to be effective rather than having an understanding of any programming language. This I feel is an important distinction to make as, anecdotally I have some fellow testers that are keen to get into technical testing and have no desire to become programmers.

NOT SPECIALISED AND NOT FUNDAMENTAL

Each discipline is not an isolated silo where the tester only focuses on one kind of work and nothing else. Each discipline covers what is to be tested in different ways and each tester is encouraged to learn about all of the disciplines that interest them.

I will now provide an overview of each discipline. For more detail including the set of applicable concepts, responsibilities, techniques, artefacts and aspects please see the Appendix.

THE BEHAVIOUR & FUNCTIONALITY DISCIPLINE

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

The behavioural and functionality tester embodies the traditional testing scope with an emphasis on the behaviour of the system rather than just meeting functional outcomes.

Functionality is defined as a specific task that the system performs. Behaviour is a system response to interaction. Behaviour can be implemented through the combination of functions (functional integration) or by a set of discrete functions. By this I mean that each function implements the behaviour in isolation.

Behaviour should be consistent and predictable. A system can be functionally correct and still have each function implemented with a different behaviour. A system with this kind of behaviour is less than ideal because it becomes more difficult to predict the outcome of an interaction.

Behavioural and functionality testing does not mandate a user interface, as not all systems require user interaction. For example batch applications and windows services.

THE USER INTERACTION DISCIPLINE

The user interaction tester is concerned with the human-computer interface. User interaction testing factors in the user experience, usability and testing the implementation of the user interface itself.

User interaction testing covers all interaction mechanisms from keyboard and mouse to touchpad and console as well as training and documentation. The last two are both considered ways in which the system interacts with the user.

The user interaction tester is aware that the success of a system extends beyond its feature set or functional correctness and that an application that help can assist a user to achieve their goals can be more successful.

User interaction testing finds potential issues as well as defects.

THE MANAGEMENT DISCIPLINE

The test management discipline focuses on two roles. The test lead, the person who is in charge of a test project, and the test manager, the person in charge of the future direction of testing in an organisation.

Administrative positions like team leader are not considered in this paper as they are not directly related to the act of testing.

THE BUSINESS DOMAIN KNOWLEDGE DISCIPLINE

The business domain knowledge discipline concerns itself with applying the testing skill set to the business domain. The business domain is all knowledge, processes and information stored, documented or surviving as tacit knowledge that pertains to the business.

Domain knowledge is a limiting factor in software testing; without knowledge of the business domain the ability to apply testing skills effectively is constrained.

THE PERFORMANCE DISCIPLINE

The performance tester is concerned with how well a system or system component behaves under a variety of controlled circumstances. The performance tester analyses the system architecture and runtime environment to devise scenarios and then execute them against the system. After execution has completed, the performance tester will analyse the results to determine whether or not there is a potential or realised performance issue.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

The performance tester is interested in the whole solution including architectural design, software implementations and infrastructure provisions.

THE SECURITY DISCIPLINE

The security tester is concerned that the system implement is secured against vulnerabilities both within itself and with respect the production environment. The security tester understands the business and the levels of security required for internal usage and accountability as well as protecting against external attacks.

There are six aspects of security testing: Availability, Authorisation, Authentication, Integrity, Confidentiality and Non-Repudiation.

THE AUTOMATION DISCIPLINE

Automation can be used in two contexts; for the facilitation of testing and for the verification of behaviour. Testers who are not dedicated automation testers can leverage the facilitation of testing aspect whilst dedicated automation testers will be interested in both.

The dedicated automation tester usually takes one of two roles; Interface testing or UI automation. The former can be subdivided into the application components that tested (batch processes, services, APIs, etc) but as a group they directly interact with the application component interface to test it. The latter is primarily used for creating build verification tests and regression suites.

The automated discipline is also responsible for developing new testing tools or using automation techniques to make their testing effort easier.

THE INFRASTRUCTURE & INTEGRATION DISCIPLINE

The Infrastructure and Integration tester is a technical tester that focuses on the integration of systems and how the systems interact with the environment.

The infrastructure discipline involves having knowledge of physical components like network topologies, virtualization and architecture design, and how this environment interacts with the system under test.

The integration aspect focuses on ensuring the system components integrate with each other irrespective of environmental influence.

THE FUNDAMENTALS OF SOFTWARE TESTING

Across all disciplines sits a common set of techniques, concepts and responsibilities that are specific to software testing. These are the fundamentals of software testing. The fundamentals cover the analysis of systems and requirements to create test scenarios; reporting and management of defects; testing techniques and test execution.

Software testing fundamentals are so important that without developing them you cannot progress within a discipline. Just as our business domain knowledge impacts on our effectiveness to design test cases; so do our fundamental competencies impact on our ability to become skilled within a given discipline. If you can't design a test case, you are not going to be able to design a performance test case.

The list of software testing fundamentals follows the same configuration as the disciplines; it provides a set of concepts, techniques and responsibilities. All fundamentals must be exclusive to all disciplines; so the analysis of requirements into functional tests does not belong to a testing fundamental.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

The full set of concepts, techniques and responsibilities of the software testing fundamentals can be found in the appendix.

ESTABLISHING A CAREER IN SOFTWARE TESTING

We now have reached a point where we have a definition for each of our disciplines and we have defined the fundamentals of software testing. One of the three problems revealed by my research was that the majority of testers surveyed felt that when they started their career they were unaware of the different directions a software tester could go.

We now have a definition of the eight discipline of software testing. The second and more important problem is that the majority of survey respondents felt that they had insufficient information to help them further develop their capabilities and achieve their desired direction.

The next part of this paper attempts to resolve this issue.

The first step that a software tester must take is to become aware of the fundamentals and disciplines of software testing. The fundamentals and each of the disciplines listed in the appendix are also road maps. Each road map represents an end state within the discipline. To become an expert within the discipline then the tester must achieve mastery over the set of techniques, responsibilities and concepts.

So what defines an expert and how do these road maps provide a tester with enough information to help them further their capabilities?

The road maps when combined with an assessment model, allow the software tester to identify areas of improvement in their skill set. The tester can then take their areas of improvement and find training courses, books or mentors to help them improve.

ASSESSMENT MODEL

What I'm proposing is a formative self assessment model to measure competency against each item in the road maps. The self assessment is done to remove the need for an organisation to run and monitor the development of each and every tester. It also allows the tester to choose the implementation of the techniques and responsibilities to suite their preferred toolset.

The assessment model measures levels of competence based on how much guidance the tester requires and how confident they are in applying each of the skills within the discipline or fundamentals. For a tester that feels their self-assessment is not accurate they can ask a mentor to work through their assessment with them.

Within each discipline and testing fundamentals are six levels of competency:

LEVEL 0 (UNAWARE)

The tester is completely unaware of the concept, technique or responsibility or has no formal knowledge. The tester cannot explain what the concept means or why it is applied.

LEVEL 1 (AWARE)

Fundamentals – The tester is aware of testing and understands why testing needs to be done, but not how they are done. This is the first level of competence where the tester can engage in discussions on the why of testing without getting into implementation details.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Discipline – The tester is aware of the discipline and understands what types of testing are performed, why it is done, the types of defect identified and the benefits of such a testing approach. A tester at this level does not know how the testing is performed.

Results – Not applicable. The tester is not yet capable of producing output.

LEVEL 2

Fundamentals – The tester is learning the concepts but is not yet competent enough to apply the concepts without continual attention. The tester should not be in charge of any part of a project and all work should be done with, or reviewed by, a mentor.

Discipline – The tester is able to identify what needs to be tested for a small subset of aspects within the discipline. The tester is able to execute existing test cases but may not be skilled enough to design all but the simplest of test cases.

Results – Not applicable. No work should be signed off without being reviewed by the test lead.

LEVEL 3

Fundamentals – The tester understands the concepts and processes and is applying them with regular review and guidance. The tester should not be in charge of a test project but the tester can be given ownership of a subsection of the project. All effort should be regularly reviewed by the experienced test lead.

Discipline – The tester is able to identify what is to be tested within the discipline for any particular project but may not be able to fully to execute the test cases. The tester should not be in charge of a discipline within a project but the tester can be given ownership of a subsection of the project relating to that discipline. All effort should be regularly reviewed by the experienced test lead.

Results – Not applicable. No work should be signed off without being reviewed by the test lead.

LEVEL 4

Fundamentals – The tester understands the concepts and processes and is able to apply them with infrequent review. The tester can be responsible for a test project. The tester does not need to be given guidance. At this point the tester is focusing on the application of their skill set. Testers of this level should be able to provide guidance.

Discipline – Is able to build the test plan and test suite for the discipline on any given project and execute the majority of tests. The tester may need occasional guidance to overcome more complicated test scenarios. Testers of this level should be able to provide guidance.

Results – Ideally a competent tester should not be responsible for more than 3 major avoidable defects being found in production per project.

LEVEL 5 (EXPERT)

Fundamentals – Satisfies all requirements for the fundamentals of software testing and should be able to provide sound guidance to any tester.

Discipline - Is able to build the test plan and test suite for the discipline on any given project and execute all tests. The expert disciplinarian should be able to overcome the most complicated of testing scenarios within the discipline.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Results - The expert tester should not be responsible for any major, avoidable defects being found in production.

A NOTE ON RESULTS

I've included the concept of results for a very specific reason. What I have set up emphasises that at the lower levels, the tester should not be responsible for their performance in production. They have already indicated that they need some degree of guidance and therefore someone should be mentoring them and verifying their work.

The aim is that an inexperienced tester who is less concerned with their performance in production will be more relaxed about trying, experimenting and learning. This is also to try to stop accusations of insufficient testing or unqualified testing when bugs are found in production. I don't like to hear reports that testers are being blamed for defects occurring in production when they have less than 6 months of experience.

At the high levels I think results are important and some may argue that 3 major avoidable defects is a major failing by a tester. The count of 3 is just an arbitrary number and is not for assessment.

The results component is not for scoring of testers. There are too many factors involved for a scoring system to work; define avoidable, define major, how much time did the tester have, how much churn was there, was the system trivial or complex, were new technologies being tried, etc.

The goal of the results component is for the tester, who when performing the self assessment, sees that they could have avoided a number of defects from getting into production. They can then use this to identify potential areas where they can improve.

The mental work flow is this:

The tester considers themselves capable within the fundamental or discipline. A number of bugs were released which contradicts their previous assessment and challenges them to review their self assessment in order to find a place where they perhaps need more guidance.

If a capable tester releases a bug into production I still believe that tester is capable but after an arbitrary number of bugs in one release then there is cause for reassessment.

AN INITIAL ASSESSMENT

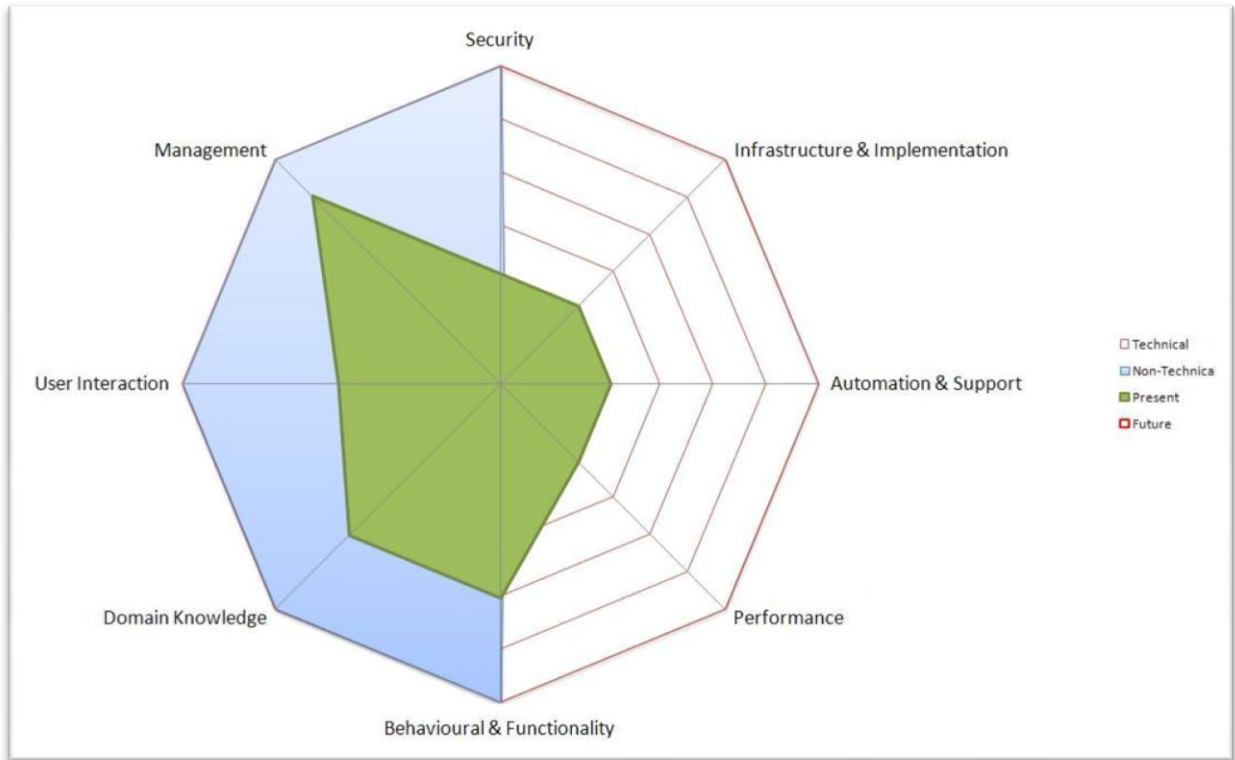
The tester, now familiar with the assessment approach takes the fundamentals road map and the road map for all disciplines. The tester then performs their self assessment. This should not be a laborious process. Feedback from testers who have recently completed their self assessment report that it takes less than one hour.

This initial assessment defines where they currently see themselves as well as identifying where they want to be in the future. As a visual aid, the tester can calculate an average level for a discipline and plot it onto a radar graph. This creates a representation of their skill-set within the domain of software testing.

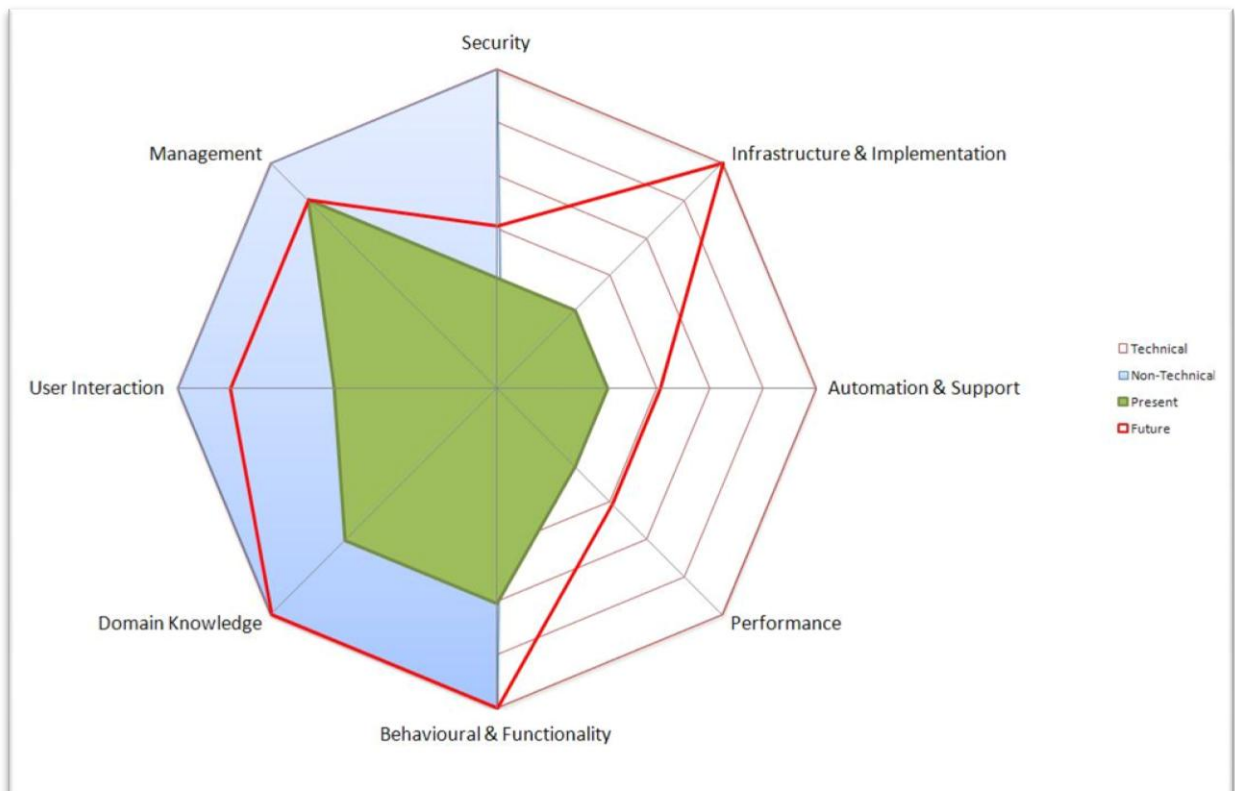
Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>



The tester can then specify where they want to be; this is their current goal for software testing learning. The tester can indicate on the chart their desired level of ability and a second shape is drawn over the top of the first. This second shape represents their future state.



The tester can now visualise their goals and with their completed assessment they will have a discrete set of techniques to practice and concepts to understand. This can be incorporated into a training program within an

organisation or if the tester is self motivated this means reading books, blogs and articles. It means doing research and discussing with their peers and mentors. It means finding a mentor. It means going on training courses. But most importantly it means practicing.

The difference between Level 1 and Level 5 is practice. Once you have an understanding of the concepts and can apply them it only takes practice to reduce your dependency on the guidance until you are at a point where you can guide others.

INFORMATION TO DEVELOP CAPABILITIES

This was the final question on the survey. Do you have all the information needed to further develop the capabilities of your desired direction in software testing? The majority of testers surveyed felt that they didn't have that information. The road maps aim to provide clarity around what testers can do, so that from the start of a career to the end, a tester can identify areas for improvement in their own skill set. From here the tester will know what has to be achieved and can find a mentor, training course or other resource to guide them.

IMPLEMENTATION & IMPROVEMENT

So far this process has been trialled in the Test & Quality Assurance team at the Australian Federal Police. It hasn't been long enough to warrant a follow up survey and the evidence of continued management buy in is lacking but the responses from the testers have been uplifting.

All this content was developed by me and peer-reviewed by a couple of smart testers to whom I am grateful but I know that this is not sufficient in the long run. This content needs to be taken by the software testing community as a whole, expanded and solidified to create a definitive syllabus of who we are and what we can do. Only then will there be sufficient information in the profession of software testing to develop the kind of futures we want.

CONCLUSION

There were three problems that were originally identified anecdotally and were revealed after a small scale survey was completed of two demographics; testers that are self motivated and operate in online communities and testers that are not experts that work in testing teams and don't actively ...

The first problem was that 70% of testers when they started testing were unaware of the there were different directions a software testing career could take. Our existing categorisations of software testing disciplines are not suitable which is why I proposed the fundamentals and disciplines as a replacement. The final step is to increase awareness and deliver this content that now meets the requirements to new testers and those struggling to find their feet in software testing.

The second problem is that the majority software testers surveyed felt they didn't have sufficient information to develop their capabilities and achieve their career goals. I have defined a model for assessment and a road map for each discipline and the fundamentals of software testing. These combined allow any tester to evaluate where they are now and identify what information they need to achieve their career goals.

The third problem is that of those that are considering leaving software testing are doing so because of the lack of information regarding professional development. This is closely tied to the first two issues and is the biggest of the problem. We can't develop futures in software testing if new testers are leaving because of a lack of information. Software testing is about providing information about problems to concerned stakeholders.

I now believe we have this information.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

ACKNOWLEDGEMENTS

Jess Kirwan

Mark Lawler

Kristen Schofield

The Australian Federal Police

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

APPENDICES

[table of appendices]

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

SOFTWARE TESTING FUNDAMENTALS ROAD MAP

The fundamentals of software testing are what we do as testers that are relevant to all disciplines. The fundamentals cover the analysis of systems and requirements to create test scenarios; reporting and management of defects; testing techniques and test execution.

Software testing fundamentals are so important that without developing them you cannot progress within a discipline. Just as our business domain knowledge impacts on our effectiveness to design test cases; so do our fundamental competencies impact on our ability to become skilled within any given discipline.

A software tester should review the list of skills and indicate their competency as per the Disciplined Software Tester competency guidelines.

The following table outlines each of the skills that are fundamental to software testers.

	Level 1	Level 2	Level 3	Level 4	Level 5
Core Skills					
Understands the role of software testing and is able to explain the who, what, where, how, why and when of software testing Concept					
Understands the difference between technical and semantic compliance and how that impacts the scope of testing. Concept					
Use positive path testing techniques to derive test cases that prove requirement implementation has been successful Test Design					
Use negative path testing techniques to derive test cases that exercise the inverse of the specified requirements Test Design					
Produce test procedures that provide additional information necessary to complete a test case. Test procedures document the how of a test case. Test Design					
Design tests that are discrete; this means that they only test one thing. Test Design					
Design tests that consider and cater for all relevant input data combinations Test Design					
Design tests that explicitly state the expectations of the tester Test Design					
Design tests that can be executed by another competent tester without manual intervention Test Design					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Core Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Analyse system behaviour to identify state data that will influence the expectations outcome and from this derive test cases that exercise these expectations Test Design					
Analysis of test scenario outcomes to identify defects Test Execution					
The ability to risk asses the test suite to identify cases that should be executed first because they present the biggest risk to the organisation if they fail. Test Execution					
The ability to create both input and state data to satisfy the test case Data Creation					
Application of the divide & conquer approach to failed tests to identify the root cause of a defect Skill					
Can analyse documented requirement and determine whether they are unambiguous Technique					
Can analyse documented requirement and determine whether they are testable Technique					
The creation of defect reports that are accurate, reproducible and have an appropriate severity. Responsibility					
Perform test case design reviews to ensure appropriate test coverage has been provided Responsibility					
Perform test case reviews to ensure test cases are written to the agreed standard Responsibility					
Perform test execution review to ensure test execution logs are correctly recorded Responsibility					
Fulfil the obligation of software testing by providing information to stakeholders regarding the quality of the solution Responsibility					
Communicate effectively by providing metrics to management regarding the testing effort Responsibility					
Escalate risks before they develop into issues Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

BEHAVIOURAL & FUNCTIONALITY TESTER ROAD MAP

The behavioural and functionality tester is the traditional testing scope with an emphasis on the behaviour of the system rather than just meeting functional outcomes.

The behavioural and functionality tester inspects the requirements documentation to create test cases and then executes these in whatever way is most appropriate for the system.

Functionality is defined as a specific task that the system performs. Behaviour is a system response to interaction. Behaviour can be implemented through the combination of functions (functional integration) or by a set of discrete functions. By this I mean that each function implements the behaviour in isolation.

Behaviour should be consistent and predictable. A system can be functionally correct but each function is implemented with a different behaviour. A system with this kind of behaviour is less ideal because it becomes difficult to predict the outcome of an interaction.

Behavioural and functionality testing does not mandate a user interface as batch applications and windows services do not require user interaction.

	Level 1	Level 2	Level 3	Level 4	Level 5
Core Skills					
Understands equivalence partitioning to assist in the creation of test cases Concept					
Understands exploratory testing and can use it as an approach to testing the system Concept					
Use boundary analysis to identify states in complex business objects that may impact expectations Responsibility					
Analyse the documented requirements, user expectations use cases, user stories, marketing material and help and training material to identify test cases Responsibility					
Can test system behaviour by combining functionality implemented within the system Responsibility					
Creation of end to end behavioural testing scenarios that exercise multiple systems within the organisation. Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Aspects

	Level 1	Level 2	Level 3	Level 4	Level 5
Functionality – what the functions of the system are and testing to ensure the system meets the specified expectations; this is done in both a positive and negative sense					
Behavioural – system behaviour is a feature that the system supports, potentially via functional integration or potentially through multiple discrete, isolated functions. Behavioural testing seeks to ensure that the expected behaviour is supported by the system and that behaviour is consistent across the system.					
Compliance – testing the system to ensure that it meets any required standards for functionality or behaviour.					
Regression – the identification of test cases that should be run after subsequent system changes to ensure the system has not regressed					
Data Integrity – inspection of system data after each function test is executed to identify incorrect storage of data					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

USER INTERACTION TESTER ROAD MAP

The user interaction tester is concerned with the human-computer interface. User interaction testing factors in the user experience, usability and testing the implementation of the user interface itself.

User interaction testing covers all interaction mechanisms from keyboard and mouse to touchpad and console as well as training and documentation, the last two are sometimes forgotten ways the system interacts with the user.

The user interaction tester is aware of the impact that the success of a system extends beyond its feature set or functional correctness and that applications that help user achieve their goals can be more successful.

User interaction testing finds potential issues as well as defects.

	Level 1	Level 2	Level 3	Level 4	Level 5
Core Skills					
Understands the difference between user testing and user feedback Concept					
Understands exploratory testing and can use it as an approach to testing the system Concept					
Can apply the A/B testing technique in order to quantify an aspect of a design. Technique					
Can apply the ladder interview technique to identify user motivations. Technique					
Can apply one of several usability inspection techniques (cognitive walkthrough, heuristic evaluation, pluralistic walkthrough) to identify design issues. Technique					
Can apply the 'think aloud' technique for identifying user interaction issues Technique					
Can apply the hallway testing technique for identifying user interaction issues Technique					
Analyse the any potential requirements to develop workflows and identify potential user interaction issues Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Analyse user interface wireframes and identify potential user interaction issues. Responsibility					
Read and review the training & help to ensure correct usage of business language; that it has correct spelling, grammar and tone; has correct content; is easy to learn and provides task oriented concise manuals. Responsibility					
Can organise and run User Acceptance Testing sessions Responsibility					
Organise and run User testing sessions and understands the implications of user demographic on such sessions Responsibility					
Produce the User Acceptance Testing report after the UAT sessions have completed and all feedback received. Responsibility					
Analyse product marketing and training material to develop test scenarios that exercise user expectations Responsibility					
Analyse the requirement specifications to develop user interaction test cases Responsibility					
Analyse user interface wireframe and develop user interaction test cases Responsibility					
Design tests that ensure the cultural background, psychological mindset and physical attributes of a user do not impact the user experience Responsibility					
Design tests that ensure all potential invocation mechanisms are catered for and support full user interaction with the system Responsibility					
Is able to debate user interaction issues calmly and intelligently. Responsibility					
User Accepting Testing Report – This report details the outcomes of the UAT session and details the input from the users. The report can contain an outline of what will be fixed in the current release and will detail which concerns will not be resolved. This report is relayed to the owners for signoff. Artefact					

Aspects

	Level 1	Level 2	Level 3	Level 4	Level 5
Accessibility – testing the user interaction so that it meets accessibility requirements for the intended users, organisation, country or standards committee. WCAG2.0 is the current W3 standard for web content accessibility.					
Consistency – the system should be consistent in its behaviour and interaction mechanisms. Consistency testing should also ensure that common UI components are used in ways that they user would expect.					
Internationalisation & Localisation – internationalisation testing covers the adaption of the system for potential use. Testing should focus on the mechanisms for internationalisation rather than the outcomes. Localisation testing covers the adaption of a software system for particular locale. It covers language and dialect translation, use of symbols, aesthetics, sort order, subtitling, cultural values and writing conventions.					
Structure – The structure of the user interface should be consistent and relevant to the functional purpose. It is making sure that the layout of the user interface has logical groupings of information, it contains sufficient information without being busy and workflows between screens make sense.					
Training & Help – the training and help testing focuses on ensuring that each UI component provides contextual help where necessary.					
Trustworthiness – focuses on ensuring that the system is forgiving of user errors and respond quickly to user input. A trustworthy system should allow the user to interact with it without fear of penalty.					
Usability – testing the usability of a system covers concepts such as efficiency, memorability, learnability, satisfaction and tolerance. An efficient system should be easy to use whilst satisfaction encompasses the usefulness of the system.					
Visibility – how well the system allows the user to construct a mental model and predict the effect of their actions. The user interaction tester should be able to identify when system information will assist users in making the best decision. Clear & simple navigation, good feedback, predicable behaviour, etc.					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

TEST MANAGEMENT ROAD MAP

The test management discipline focuses on two roles. The test lead, the person who is in charge of a test project, and the test manager, the person in charge of the future direction of testing in an organisation.

Administrative positions like team leader are not covered as they are not directly related to the act of testing.

The section is broken up into two sections; one for test lead and one for the test manager.

	Level 1	Level 2	Level 3	Level 4	Level 5
Test Lead Skills					
Is aware of and understands each of the other disciplines so that the test plan can include accurate resource provisions and estimates Concept					
Provide input into processes for test design peer reviews Responsibility					
Provide input into processes for test case specification peer reviews Responsibility					
Provide input into processes for test execution peer reviews Responsibility					
Provide input into the defect workflow process Responsibility					
Produce a test plan without relying on a template Responsibility					
Provide input into the project initiation phase by supplying a high level testing strategy, test estimates and test resource needs. Responsibility					
Produce the test summary report at the completion of a testing project. Responsibility					
Allocate effort to members of the test team relative to their discipline Responsibility					
Negotiate sponsor signoff with the project owners Responsibility					
Discuss, mentor or train current employees on testing fundamentals Responsibility					
Communicate effectively with the project manager on project progress Responsibility					
Report to test management the performance of the project team Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Test Lead Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Produce metrics to test management regarding performance of the test team and the project Responsibility					
Effectively communicate with the development lead to negotiate test engagement Responsibility					
Effectively communicate with the development lead to negotiate defect resolution processes Responsibility					
Test Plan – outlines the testing effort to be undertaken on a given project. Includes scope, estimations, risks and issues, environment configuration, resourcing and required disciplines. Artefact					
Test Summary Report – details the testing that was undertaken on the project and the state of the testing Artefact					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Test Management Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Understands different estimation models, how they relate to testing within the organisation Concept					
Understand the various models that exist for software testing and which is one is more appropriate for the organisation Concept					
Understands the different test environment configurations Concept					
Is aware of and understands each of the other disciplines so that the test strategy can include accurate resource provisions and estimates Concept					
Can estimate effectively for future testing effort Responsibility					
Define processes for test design peer reviews Responsibility					
Define processes for test case specification peer reviews Responsibility					
Define processes for test execution peer reviews Responsibility					
Define the defect workflow process Responsibility					
Plan the test environment configuration is best given the organisational resources Responsibility					
Define the structure for teams working within the testing area Responsibility					
Review current skill sets and allocate training or acquire new resources to cover current skill shortages Responsibility					
Define the role testing plays in the organisation Responsibility					
Recruitment and retention of skilled testing personnel based on current and future testing needs Responsibility					
Planning, deploying, and managing the testing effort for any given engagement / release. Responsibility					
Produce the short term test strategy Responsibility					
Produce the test future direction plan Responsibility					
Report to upper management the performance of the testing team Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

<p>Short Term Test Strategy – defines the testing resources, training, approaches and utilisation over the coming 12-24 month period</p> <p style="text-align: right;">Artefact</p>					
<p>Test Future Direction Plan – defines the testing resource, training and approaches to meet the expected organisational testing needs based on the CTO Future Direction</p> <p style="text-align: right;">Artefact</p>					

BUSINESS DOMAIN KNOWLEDGE ROAD MAP

The business domain knowledge discipline concerns itself with applying the testing skill set to the business domain. The business domain is all knowledge, processes and information stored, document or surviving as tacit knowledge that pertains to the business.

Domain knowledge is a limiting factor in software testing; without knowledge of the business domain the ability to apply testing skills effectively is constrained.

<i>Core Skills</i>	Level 1	Level 2	Level 3	Level 4	Level 5
Understands the language used by the business Concept					
Understands the business and the role software plays within the business Concept					
Analyse the requirements to develop test scenarios that model how the business is going to use the system Responsibility					
Analyse the requirements in the context of the business to ensure that the specified requirements are going to meet the needs of the business. Responsibility					
Analysis of semantic contracts, both explicit and implicit within the system to identify potential defects Responsibility					
Discuss the impact and severity of the defect from the perspective of the user. Responsibility					
Documentation of tacit knowledge as it is discovered Responsibility					
Identify systems that due their impacts on society or life require failure evident capabilities but do not feature them Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

PERFORMANCE TESTER ROAD MAP

The performance tester is concerned with how well a system or system component behaves under a variety of controlled circumstances. The performance tester analyses the system architecture and runtime environment to devise scenarios and then execute them against the system. After execution has completed the performance tester will analyse the results to determine whether or not there is a potential or realised performance issue.

The performance tester is interested in the whole solution including architectural design, software implementations and infrastructure provisions.

<i>Core Skills</i>	Level 1	Level 2	Level 3	Level 4	Level 5
Understanding why performance testing is done Concept					
Understanding the observer effect and its impact in your performance test Concept					
Understanding controllers and agents and their usage in performance testing Concept					
Understanding protocols and how they are relevant for different types of performance problems and solution architectures. Concept					
Has a solid understanding concepts of software development and can apply them in development performance test cases or a performance test framework Concept					
Understands the implications of virtualisation on performance testing Concept					
Understand the difference between clustering and load balancing and the implications each has on performance testing Concept					
Understands background noise and implications it has on performance test results Concept					
Understands the implications of different operating systems on performance Concept					
Understands the implications of virtual machines on performance Concept					
Understands how infrastructure components communicate and can therefore devise test scenario based on this information Concept					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Core Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Is aware of the available profiling counters and knows which ones to use to diagnose a particular performance problem. Concept					
Understands the difference between throughput (transactions per interval) and latency (responsiveness) and can identify when performance tests should measure either. Concept					
Understands what a resource bound component is how that impacts performance. Is able to articulate why a resource bound component can be an issue. Concept					
Understands what a CPU bound component is and how that impacts performance. Is able to articulate why a CPU bound component can be an issue. Concept					
Understands what CPU starvation is and how that impacts performance. Is able to articulate why a CPU starved component is an issue for the current architecture. Concept					
Understands what handshake issues are and how they impact performance. Is able to identify handshake issues. Concept					
Understands the difference resources that are available within an environment and how these are used (e.g. threads, handles) Concept					
Understands how different resource allocation policies can impact or benefit performance. Concept					
Understands how difference caching policies can impact or benefit performance. Concept					
Understands the concept of thrashing and can identify it within performance testing results. Concept					
Understands the difference between message size policies and how that can impact on performance. Concept					
Understands the horizontal and vertical scaling models, tradeoffs between both and can perform scalability testing on either. Concept					
Understands IP Switching or IP Spoofing and how that impacts on performance tests. Concept					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Core Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Understands the current and intended growth to be used in load and volume scenarios Concept					
Use the organisation's performance testing tool set to execute performance tests Responsibility					
Use programming languages associated with the automation tool. Responsibility					
Debug automation test code during development and fix broken performance tests. Responsibility					
Create or identify state and input data needed to satisfy a performance testing scenario and is aware of the implications of data creation during performance testing Responsibility					
Understand the difference between identity and anonymous data and the impacts it has on automated data creation Responsibility					
Analyse system architecture design to identify potential bottlenecks Responsibility					
Read network topologies and devise test scenarios from them Responsibility					
Create end to end test scenarios that better mimic production usage and understands the difference between isolated and integrated performance tests Responsibility					
Make recommendations for improving architectural design with respect to performance issues Responsibility					
Is able to evaluate multiple potential performance testing tools and evaluate them against criterion to determine which tool is best for a given organisation. Responsibility					
Produce the performance test report at the completion of all performance testing activities for a project Responsibility					
Produce the performance test plan using available knowledge of project and current architectural design. Responsibility					
Produce a 'state of the system' benchmark document that identifies each infrastructure component and their 'load' levels during standard, peak and trough times. Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Core Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
<p>Establish and manage a performance testing environment so that it best replicates production whilst allowing the isolation of new components.</p> <p style="text-align: right;">Responsibility</p>					
<p>Test the system failure evident capabilities using performance testing techniques.</p> <p style="text-align: right;">Responsibility</p>					
<p>Performance Test Plan – The performance test plan <i>details what performance testing is going to be undertaken on a project, where it will occur and who is doing it.</i></p> <p style="text-align: right;">Artefact</p>					
<p>Performance Report – The performance report is produced after all performance testing has been completed. It outlines how the testing went and if there are any pending issues or future performance constraints.</p> <p style="text-align: right;">Artefact</p>					
<p>State of the System Report – The state of the system report should be compiled after each production deployment. It documents each infrastructure component and the amount of ‘load’ that it experiences. The load should be separated into min, peak and average. This is used with performance testing to determine issues that may occur when the system is deployed to production.</p> <p style="text-align: right;">Artefact</p>					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Aspects

	Level 1	Level 2	Level 3	Level 4	Level 5
Performance – the fundamentals of performance tests are to measure two things; throughput and latency. This is the first step where basic timings are recorded without load.					
Load – how does the system operate under difference usage scenarios? Load testing looks at three basic tests; the spike test where many users access the system at once; the soak test where there is a high load activity over an extended period of time and the expected usage scenario.					
Volume – volume testing is the inverse of load testing; where rather than changing the concurrency of users we alter scope of their effort. A search test may be performed with both one million and ten million records to see how the volume change impacts the performance					
Stress – stress test is done to determine the maximum capacity of the system. Aside from pushing volume and load to their maximum possible levels there are other ways to stress the system: resource restrictions, resource competition and bandwidth throttling.					
Reliability – similar to the soak test, the reliability test is performed to determine whether or not the solution meets its uptime requirements given expected usage.					
Scalability – here we look at how well the system can scale out. We can do this by throttling server capacity and seeing how easily the system scales horizontally or vertically.					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

AUTOMATION TESTER ROAD MAP

Automation can be used in two contexts; for the facilitation of testing and for the verification of behaviour. Testers who are not dedicated automation testers can leverage the facilitation of testing aspect whilst dedicated automation testers will be interested in both.

The dedicated automation tester usually takes one of two roles; Interface testing or UI automation. The former can be subdivided into the application components that tested (batch processes, services, APIs, etc) but as a group they directly interact with the application component interface to test it. The latter is primarily used for creating build verification tests and regression suites.

The automated discipline is also responsible for developing new testing tools or using automation techniques to make their testing effort easier.

The automated skill set is broken into three lists; common skills and then skills associated with interface testers and UI automation testers.

	Level 1	Level 2	Level 3	Level 4	Level 5
Common Skills					
Has a solid understanding of the concepts of software development and can apply them in development automation test cases or a automation test framework Concept					
Understands when automation is the ideal approach for behaviour verification and when it should not be used at all Concept					
Understands how change impacts automation tests Concept					
Understands the value of automation with respect to repeated execution against changing environment states Concept					
Understand the difference between identity and anonymous data and the impacts it has on automated data creation and automated test execution Concept					
Understanding the difference between data dependant and data independent tests and when to use each one Concept					
Understands the difference between testing and checking Concept					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Common Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Can communicate with other infrastructure components programmatically to assist in testing. E.g. calling a database to verify state data Technique					
Understands mocks and how they can be used for isolating systems under test. Technique					
Understands stubs and how they can be used for isolating systems under test Technique					
Write automation tests that are resilient against change Responsibility					
Create both state and input data for use within automated tests Responsibility					
Use programming languages associated with the infrastructure to assist in testing. Responsibility					
Use programming languages associated with the automation tool. Responsibility					
Debug automation test code during development and fix broken automation scripts. Responsibility					
Use the organisation's automated testing tool set to execute automated tests Responsibility					
Writing new automation tools Responsibility					
Evaluate difference automation tools to identify which one is best for the organisation given specific criterion Responsibility					
Define processes for automation code reviews Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Interface Testing Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Understanding the role the component plays in the organisation and the impact that has on testing Concept					
Understands the difference between RPC, Message and Message Bus invocation mechanisms and how that impacts testing. Concept					
Understands component configuration based on technologies selected by developers Concept					
Understands the difference between SOAP, WCF and RESTful services and how they impact service testing Concept, Specialised					
Understands service discovery and how that impacts service integration testing. Concept, Specialised					
Understands the difference between service orchestration and service choreography and how they impact service integration tests Concept, Specialised					
The analysis of component contracts to identify design issues Technique					
The analysis of component documentation to identify design issues Technique					
Analyse the component contract to create test scenarios Responsibility					
Analyse component documentation to create test scenarios Responsibility					
Identify a change in component contract and update test accordingly Responsibility					
Automation the configuration of a component for testing Responsibility					
Configuring the component tool to work with non-trivial component configurations Responsibility					
Understands the standards associated with component development (SOAP, WCF standards, etc) Responsibility					
Communicate effectively with developers to rely potential design or implementation issues Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Interface Testing Aspects

	Level 1	Level 2	Level 3	Level 4	Level 5
Input Validation – testing the component contract with varying forms of invalid data. Considers invalid data in terms of the contract (technical) and invalid data in terms of the business (semantic).					
Functionality – testing the functionality of the service in isolation					
Security – working with the security tester to test the security characteristics of the service from infrastructure and message security to business security implementations.					
Compliance – testing the component to ensure it complies with applicable standards.					
Exception Reporting – testing the component to ensure that it reports exception information correctly and appropriately.					
Auditing – working with the security tester to ensure the component correctly audits requests made to it and that the audit information is secure.					
Data Persistence Integrity – ensuring that any information persisted by the component is done in a consistent manner.					
Negative Testing – executing a positive test path scenario over an incorrectly configured component to ensure the component gracefully handles the error.					
Interoperability – Testing the component so that it can be used with different technologies (Java vs. dotNet, etc)					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

UI Automation Testing Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Understands the complexities associated with writing checks for different UI technologies and how that impacts check authoring. Concept					
Understands the impacts of change on UI automation checks and how to alleviate these issues. Concept					
Understands the purpose of UI automation and that it is not always focused on identifying bugs that exist now Concept					
Understands the additional complexities identity data places on UI automation checks especially when it results in custom workflows Concept					
Is able to identify when to use UI automation to create data and when to use non UI based techniques. Technique					
Liaise with the other test disciplines to devise appropriate automation scenarios Responsibility					
Develop automation scripts that are resistant to changes in the content and layout of the user interface Responsibility					
Work with performance testers to develop automation performance tests that with the user interface Responsibility					

There are no aspects associated with UI Automation as it is covered under the User Interaction tester or under the Behavioural and Functionality Tester.

INFRASTRUCTURE & INTEGRATION TESTER ROAD MAP

The Infrastructure & Integration tester is a technical tester that focuses on the integration of systems and how the systems interact with the environment.

The infrastructure discipline involves having knowledge of physical components like network topologies, virtualization and architecture designs and how this environment interacts with the system under test.

The integration aspect focuses on ensure the system components integrate with each other irrespective of environment influence.

	Level 1	Level 2	Level 3	Level 4	Level 5
<p>Core Skills</p> <p>Understands the different integration techniques: Top-down, Bottom-Up and Big Bang and when to apply it</p> <p>Concept</p>					
<p>Understands what infrastructure is deployed in the environment and how that impacts the system under test</p> <p>Concept</p>					
<p>Understands what systems are deployed in the environment and how that impacts the system under test</p> <p>Concept</p>					
<p>Understands the concepts of configuration management</p> <p>Concept</p>					
<p>Understands the production administration teams are users of the system and that the developed solution is usable for them.</p> <p>Concept</p>					
<p>Understands difference backup strategies and their uses</p> <p>Concept</p>					
<p>Understands component configuration based on technologies selected by developers</p> <p>Concept</p>					
<p>Can apply fuzz testing for the purpose of robustness testing</p> <p>Technique</p>					
<p>Factor the impacts of platforms have on test outcomes during test design and prepare cases that exercise the different browsers, virtual machines and operating systems</p> <p>Responsibility</p>					
<p>Factor the impacts of environments on test outcomes during test design and prepare cases that determine the impact of relevant hardware, peripherals and virtualisation.</p> <p>Responsibility</p>					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Core Skills

	Level 1	Level 2	Level 3	Level 4	Level 5
Analyse the system architecture in order to determine test scenarios Responsibility					
Perform administration of infrastructure components for the purposes of test execution and data creation Responsibility					
Communicates with system administration teams to get their perspective on the system within the environment. Responsibility					
Verify organisational system operations can monitor system behaviour and can respond accordingly to system failure. Responsibility					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

Aspects

	Level 1	Level 2	Level 3	Level 4	Level 5
Deployment – focuses on the act of deployment whether via user installation or desktop rollout techniques. Deployment testing looks at whether the application can be installed easily with no manual steps, the workflow for installation is correct and that the application works on all supported hardware and software configurations.					
Failover – Failover is the capability for infrastructure components to switch over to secondary components in the event of failure. Testing looks at whether the setup failover configuration works correctly by running failover scenarios.					
Backup & Recovery – tests the backup and recovery procedures for an application or infrastructure component. Backup & Recovery should investigate all components that can fail and ensure that the system can be restored in the event of failure.					
Disaster Recovery – tests the disaster recovery plan. Simulate disasters by disabling infrastructure in a controlled way and then following the plan to ensure that it works.					
Integration – tests the integration of system components with respect to a project or change. Includes both project delivered components, existing components, 3 rd party components (browser upgrades, security patches, etc) and ‘Off the Shelf’ products					
Instrumentation – Instrumentation is data provided by an application during runtime. This can be used to diagnose or identify developing issues. Instrumentation testing focuses on ensuring the content is correct and written at appropriate times.					
Robustness – Looks at how well a system handles invalid or incorrect configuration. Robustness of input is covered in User Interaction or Business & Functionality Testing so we focus exclusively on the interactions between the system and its components or the system and the environment.					
Interoperability – focuses on testing system components can be communicated with using different technologies. For example interoperability testing can be done on a service to ensure it can be consumed by both .NET and Java clients.					

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

SECURITY TESTER ROAD MAP

The security tester is concerned with that the system delivered is secured against vulnerabilities both within itself and with respect the production environment.

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>

MISC CONTENT

Things to take away

- 1. When a tester starts they are aware of all the directions and will chose what areas they would like to learn and specialise in.**
- 2. A tester can change whenever they feel the need to change but the options are standard and available**
- 3. A tester who selects a discipline to learn then they can take the road map to work against. A list of concepts, techniques, responsibilities, artefacts and testing aspects for them to focus on.**
- 4. The road map does not provide how you should achieve or learn these tasks; only what you can do. The how is determined by context and appropriate practices and the learning can be done via self development, training courses, and real world experience.**

Ryan Boucher

email: ryan.boucher@distributedlife.com

web: <http://www.distributedlife.com>