An Investigation into the impact of Machine Learning on Software Testing

Karl Morrissey

A dissertation submitted to the University of Dublin in partial fulfilment of the requirements for the degree of MSc in Management of Information Systems **1st May 2019**

Abstract

Software Testing is an integral function of the Software Development Lifecycle (SDLC). Manual testing has long been the tried and tested approach whereby test cases are designed and executed by human resources. Automated testing reduces the effort required by software test engineers by transferring repetitive test case execution to software programs that can be run continuously to provide rapid feedback to project stakeholders. However, studies suggest that the adoption rate of automated testing is 30% or lower across the software testing industry.

Machine learning can reduce repetitive tasks and adapt to changing environments. The benefits of machine learning align closely with the goal of automated software testing.

This investigation researches how machine learning can be leveraged by software testing practitioners and assesses the capabilities of commercial testing services which incorporate machine learning. The potential impact on the software testing profession is found to have both positive and negative implications.

The forecast for the adoption rate of machine learning across the software testing industry is concluded to be low within a five-year timeframe but the potential presented by machine learning to resolve many of the issues found with standard automation tools is such that it is only a matter of time before ML-Driven automated tools become an industry standard.

Acknowledgements

I would like to thank my supervisor, Paula Roberts for her invaluable support during the creation of this dissertation.

I would also like to thank the respondents who gave up their time to share their knowledge to help me write this paper.

I would like to take the opportunity to recognise the support I have received from my employer, both financially and technically.

Lastly, I would like to thank my partner, Katharina Casey for her patience and advice during the long hours spent bringing this project to a conclusion.

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work, and has not been submitted as an exercise for a degree at this or any other university. I further declare that this research has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: 

Karl Morrissey

30th April 2019

Permission to lend and/or copy

I agree that the School of Computer Science and Statistics, Trinity College may lend or copy this dissertation upon request.

Signed: 

Karl Morrissey

30th April 2019

Table of Contents

List of Tables

List of Figures

Abbreviations

AI	Artificial Intelligence
CI	Continuous Integration
ML	Machine Learning
SDLC	Software Development Lifecycle
SUT	System Under Test

1. Chapter 1: Introduction

1.1 Context and General background

Software Testing is an integral function of the Software Development Lifecycle (SDLC). Testing ensures that software features meet the requirements of clients while also providing a level of assurance that software is delivered to agreed quality targets through the identification of defects residing within the code. Manual testing has long been the tried and tested approach whereby test cases are designed and executed by human resources. Automated software testing was introduced in the 1990s which provides software testers with the option to use automation tools and methods to record or code test cases that be executed programmatically.

Automated testing can be complicated and expensive to maintain. There are limitations on how effective they can be, particularly in a system that is updated regularly. This constrains emerging development processes associated with agile methodologies which demand rapid feedback from test processes in order to ensure Continuous Integration (CI) practices operate properly. CI requires that bulk sets of test cases are executed on a nightly basis. Manual tests cannot provide this volume of testing at this frequency, which leaves test automation the only viable option. With studies reporting that automation is only adopted by between 18% - Throvagunta, S., Olsen, B., Aymer, A. (2018) and 30% - Appvance (c.2018) of software development organisations currently, it is clear that there is a problem. Software testing practices are falling behind as development practices continue to evolve.

Machine Learning (ML) is a branch of Artificial Intelligence (AI) which incorporates sophisticated algorithms which can 'learn' from data sets and adapt to changes to the system in which it operates. The advantages of machine learning are that it can process large volumes of data and adapt quickly to change, while automating tasks that previously were only suited to human application.

The goal of test automation is to facilitate the execution of larges sets of test cases, which in turn allows for the rapid identification of defects in a system which is continuously changing. Machine learning capabilities seem ideal to disrupt software testing practices by providing the capability to automatically generate test cases for the System Under Test (SUT) and proactively respond to changes to underlying behaviour.

The intention of the investigation outlined in the following chapters is to assess the impact of machine learning on software testing within the context of the following areas:

- The value that machine learning offers software testing in terms of improved capability
- The commercial availability and adoption rates of machine learning services in the software testing industry
- Disadvantages to machine learning services which would impede adoption
- Positive and negative implications for humans working in software testing
- The five-year forecast for machine learning in software testing

It is assumed that the results of this investigation would be of interest to not only practitioners in the field of software testing but to anyone who is interested in improving software development delivery timelines and providing clients with new features at a quicker rate.

1.2 Scope of the Study

The main question for investigation is **'How can Machine Learning be Leveraged for Software Testing within a Timeframe of Five Years?**' This question was then broken out into three sub-questions:

Question 1: How is machine learning (ML) being used to impact software testing techniques?

This is an investigation into how machine algorithms are being adapted in software testing.

Question 2: How are commercial ML-Driven automated software testing tools different from traditional non-ML automated tools?

This is an investigation into the commercial availability of machine learning services and tools and what improvements, if any, they offer over standard automation tools. During this dissertation the term 'standard automation tools' refers to tools that do not incorporate machine learning.

Question 3: How will the rate of adoption impact on professional software testing from both a human and business perspective within five years?

This question aims to predict, using information gathered at each stage of the investigation, how ML-Driven automated testing tools and services will impact on the software testing profession from a human perspective. What are the negative connotations and what are the positive? An attempt is also made to forecast the adoption rate of ML-Driven tools within a timeframe of five years.
Each chapter is framed with these questions in mind. The conclusions to these questions combine to answer the primary question.

1.3 Research Questions/Objectives

Two complementary approaches comprised the overall research design. The first was to conduct semi-structured interviews with leaders in the field of software testing. This provided useful information as to the awareness of machine learning in the testing industry. It provided qualitative data which was used to assess the human impact of machine learning and provided data which was used to assess the adoption rate over the next five years. The potential value that the respondents associated with machine learning was then used to structure the data captured when reviewing the services currently available on the market that offer ML-Driven automated testing. An empirical analysis was done on five services which were selected through researching industry-related material. This data was then analysed using the online tool 'Dedoose', which provided a framework for qualitative analysis. Data was then correlated with respondents' data to provide a combined assessment of machine learning in software testing

The literature review was carried out from September 2018 to February 2019 and then refined over the remaining months to April 2019. Criteria for assessing the ML-Driven services were designed over March, and later modified after the semi-structured interviews were conducted. Ethics approval was granted on April 3rd, 2019 and semi-structured interviews commenced after that.

1.4 Chapters

The dissertation contains the following chapters:

Chapter 1: Introduction

This provides information as to the goal of the investigation, who might be interested in the findings and information regarding the structure of the investigation

Chapter 2: Literature Review

During the literature review an investigation is carried out within the context of the three sub-questions by reviewing available academic journals, industry-related articles, standards bodies, commercial reports and individuals' blogs. Common themes are identified, and a conclusion is drawn which is later expanded on by the research methodology.

Chapter 3: Methodology and Fieldwork

During this chapter the research methodology is explained starting from the philosophical concepts underpinning the approaches undertaken. The strategy and approaches are described in detail and the rationale behind each interview question and empirical evaluation are explained within the context of the sub-questions.

Chapter 4: Findings and Analysis

The results from the data collections are presented during Chapter 4 and findings from the qualitative analysis are discussed and used to provide conclusions as to the impact of machine learning on software testing.

Chapter 5: Conclusions and Future Work

Chapter 5 provides a conclusion to the dissertation and submits an answer to the question of the investigation. Future research is suggested which can extend on the findings produced from this investigation.

2. Chapter 2: Literature Review

2.1 Introduction

The primary question for investigation is: **How can Machine Learning be Leveraged for Software Testing within a Timeframe of Five Years?**

This section contains a review of selected literature related to the potential impact of machine learning on software testing practices. Information has been gathered from several different source types, including academic journals, industry-related articles, standards bodies, commercial reports and individuals' blogs.

The primary question is broken down into three sub-questions:

Question 1: How is machine learning (ML) being used to impact software testing techniques?

This is an investigation into how machine learning algorithms are adapted for the improvement of software testing techniques, metric gathering, test coverage and defect identification.

Question 2: How are commercial ML-Driven automated software testing tools different from traditional non-ML automated tools?

This is an investigation into commercially available software test tools and services which incorporate machine learning. Included is a review of the improvements they possess compared to standard automated tools.

Question 3: How will the rate of adoption impact on professional software testing from both a human and business perspective within five years?

Through a review of expert opinions, commercial forecasts and an assessment of the maturity of available ML-Driven automated testing tools an informed predication is proffered of the adoption rates and impact of the tools in the software development industry.

2.2 Background

2.2.1 Software Testing Theory & Practice

Software Testing is an integral part of the Software Development Lifecycle. Its purpose is to ensure that the system under test (SUT) behaves as intended. Verification involves the practice of identifying technical defects in the code and ensuring there are no mistakes in associated documentation. As Sharma, L. (2017), defines it in 'Difference between Verification and Validation', "Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful. Verification is concerned with whether the system is well-engineered and error-free." Validation describes the process of ensuring the SUT meets the requirements set out by the customers or end users. "It is a dynamic mechanism of validating and testing the actual product."- Sharma, L. (2017). Simply put Validation ensures we're building the right thing; Verification ensures we're building it right.

Software testing can be split into two categories: White Box Testing and Black Box Testing. White Box Testing 'is based on an analysis of the internal structure of the component or system.' - ISTQB Glossary (c.2018). For White Box Testing the code of the underlying application is known and usually involves unit testing and component testing. White box is an important phase of the testing lifecycle and ensures that the fundamental components of the code are tested to ensure correct behaviour. Black Box Testing involves testing the functionality of the software without access to the code underneath. This usually involves front-end testing, ensuring that the application behaves as expected from an end-user perspective and reflects business requirements. The ISTQB Glossary (c.2018). defines this as 'Testing, either functional or non-functional, without reference to the internal structure of the component or system'. Artificial Intelligence, and particularly the branch of AI called Machine Learning has potential applications across both White and Black Testing techniques.

2.2.2 Manual, Automation, Agile & Continuous Integration

Traditional waterfall type development methodologies usually include long release cycles, with costly manual regression testing phases. Manual testing is done by software testers who design, write and execute test cases to identify flaws in the design of the software and defects in the software code.

Automated testing involves a software tool or service that can be used to record or code a set of test cases. The tool can be used to execute these test cases and provide feedback on the stability of the SUT. Automated software testing has evolved since its introduction in the mid-1990s. Tools have become more intuitive and less expensive to employ. This evolution is further discussed in section 5 of this Chapter. As the software development industry has adopted more agile development practices, shorter release cycles have resulted in increased pressure to deliver new features to market at a faster. Agile Software Development is described in the ISTQB Glossary (c.2018) as 'A group of software development methodologies based on iterative incremental development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.' Incremental development practices require rapid feedback from test activities, which rely on automated testing to provide this test coverage within a Continuous Integration process.

Continuous Integration (CI) is a feature of modern software development practices. According to Radigan, D. (c.2018) in 'Continuous Integration Explained' this is 'the practice of routinely integrating code changes into the main branch of a repository, and testing the changes, as early and often as possible.' Stable, repeatable automated tests are a necessary component of a successful Continuous Integration framework.

Automated testing, when properly integrated, provides the rapid feedback needed for effective CI but it can be difficult to implement, which is reflected in low adoption rates across the software development industry. In a study involving 1700 executives, representing 10 different sectors Throvagunta, S., Olsen, B., Aymer, A. (2018) found that

> 'On average, 18% of functional test cases were generated using test generation tools, and 16% were executed using test automation tools. Similarly, 16% of all security tests were executed using automation tools, and automation was also applied to the execution of 16% of all performance test cases. Quite encouragingly, 15% of all end-to-end business scenarios were also being executed using test automation tools.'

These statistics suggest that most organisations are still relying heavily on manual testing, and are not investing in automated testing, for several reasons, discussed further in section 5 of this Chapter.

2.2.3 Artificial Intelligence and Machine Learning

The terms Artificial Intelligence (AI) and Machine Learning (ML) are often used interchangeably but there are distinct differences. AI describes systems that can make decisions based on the information presented to them. AI uses machine learning to process data to perform some operation for which the AI system was built. Machine Learning allows a system to learn from a data set that is changing constantly and is not known at the time of programming. Machine learning moves away from a constrained program and allows a system to adapt to the information or data that is presented to it at a given time, and through computational algorithms adapts and 'learns' without necessary further human intervention.

For the purposes for this investigation Machine Learning can be considered as a branch or subset of Artificial Intelligence. This is captured in the Glossary from PWC (2017) in Figure 2-1.

AI consists of a number of areas, including but not limited to those below:

Main AI areas	*Description*
Large-scale Machine Learning	Design of learning algorithms, as well as scaling existing algorithms, to work with extremely large data sets.
Deep Learning	Model composed of inputs such as image or audio and several hidden layers of sub-models that serve as input for the next layer and ultimately an output or activation function.
Natural Language Processing (NLP)	Algorithms that process human language input and convert it into understandable representations.
Collaborative Systems	Models and algorithms to help develop autonomous systems that can work collaboratively with other systems and with humans.
Computer Vision (Image Analytics)	The process of pulling relevant information from an image or sets of images for advanced classification and analysis.
Algorithmic Game Theory and Computational Social Choice	Systems that address the economic and social computing dimensions of AI, such as how systems can handle potentially misaligned incentives, including self-interested human participants or firms and the automated AI-based agents representing them.
Soft Robotics (Robotic Process Automation)	Automation of repetitive tasks and common processes such as IT, customer servicing and sales without the need to transform existing IT system maps.

FIGURE 2-1 GLOSSARY OF AI TERMINOLOGY. SOURCE: PRICE WATERHOUSE COOPER (2017)

Although Marr. B, (2016) would define the difference between AI and ML differently: 'Often referred to as a subset of AI, it's really more accurate to think of it as the current state-of-the-art.' This illustrates the ambiguity that surrounds the terminology related to AI and ML. This investigation refers to machine learning predominantly, except for some instances,

such as the research interviews where the term AI is used synonymously with ML to facilitate the discussion with respondents.

As machine learning, evolves, its adoption by organisations across various industries is predicted to increase. According to the report 'AI predictions. 8 insights to shape business strategy' by Price Waterhouse Cooper (2018) 'In the near-term, the biggest potential economic uplift from AI is likely to come from improved productivity. This includes automation of routine tasks, augmenting employees' capabilities and freeing them up to focus on more stimulating and higher value-adding work.' This further illustrated by the predicted growth of value derived from machine learning technologies over the coming decades as capture in Figure 2-2.

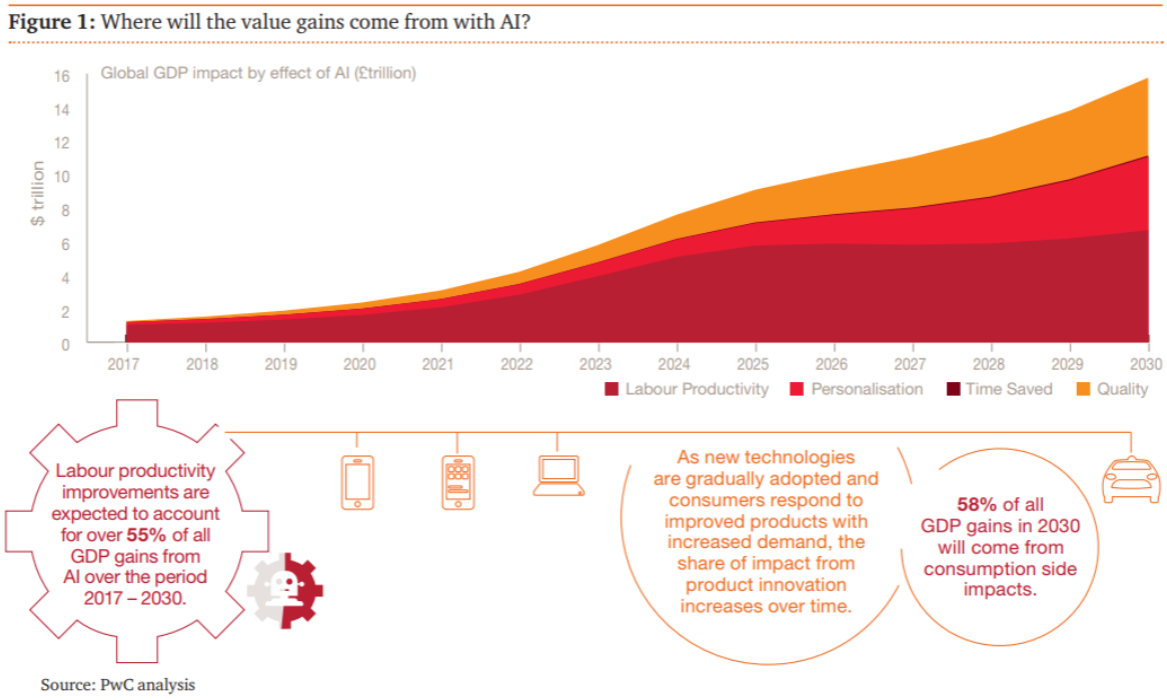

FIGURE 2-2 WHERE WILL THE VALUE GAINS COME FROM WITH AI? SOURCE: PWC (2017)

The goal of Software Test Automation is closely aligned with the vision of machine learning to automate routine tasks and free workers up for more creative work. Standard automation tools (tools that do not incorporate ML) rely heavily on a human understanding for design and creation of automated test cases. There is potential for ML-Driven software automation tools to reduce the need for human intervention through their use of machine

learning algorithms to produce automated test scripts that can verify the SUT with potentially faster feedback of results, automatic test case generation, reduced maintenance costs and automated user interface (UI) testing.

2.3 Machine Learning Algorithms use in Software Testing

How is machine learning (ML) being used to impact software testing techniques?

The learning capability available from machine learning algorithms offers software test practitioners a powerful advancement in automated testing. King, T M., Santiago, D., Phillips, J., Clarke, P. (2018) define Machine Learning as:

> 'a subset of AI that involves the construction of algorithms that can learn from and make predictions on data. Formally stated, if the performance P of a computer program at completing a task T improves with experience E, the program is said to have learnt.'

Software test tools that can self-learn and evolve have significant potential for a range of test activities. Machine learning can increase the speed and coverage of test automation while reducing or eliminating the need for testers to code and maintain large sets of automated tests scripts. Whereas human-driven automation techniques rely on software testers to design and generate test cases, machine learning can potentially be used to generate test cases automatically by coupling established software test methods with machine learning algorithms to produce tools that provide effective code coverage without relying on the same level of time-consuming and costly analysis done solely by human application.

2.3.1 Test Case Generation

In a study to automatically create test cases for Android mobile applications Rosenfeld, A., Kardashov, O. & Zang, O. (2018) presented *'a novel approach for the automation of functional testing in mobile software by leveraging machine learning techniques and reusing generic test scenarios.'* The approach split out mobile applications into a number of generic activities, which included: Splash Activity, Advertisement Activity, Login Activity, Portal Activity, Mail Activity, Browser Activity, To Do List Activity. Machine learning algorithms were then used to classify screens in a mobile application into these activities. They found the KStar machine learning algorithm most effective.

Through the accurate classification of the screens they were able to execute a set of generic test cases based on the activities. The researchers found that by utilising machine learning they were able to 'automatically cover a large portion of the human testers' work suggesting a significant potential relief in the manual testing efforts.' This is an example of ML being used to generate test cases automatically, which still required much effort and manual data entry in order to produce a working solution.

2.3.2 Model-Based Testing

Model-Based-Testing (MBT) is a relatively new test strategy where the SUT is represented as a system model, which is designed by developers and testers and used to generate test cases. According to Güldali, B., Mlynarski, M. & Sanca, Y. (2010) 'MBT uses abstract models (test models) of the system under test (SUT) or its environment as the source for test case generation. In addition to models of SUT and the environment, also the testware, e.g test execution environment or test cases, can be modeled'

The Environment-Application Interaction Model is an existing model that is used to represent interactions between a given system and the environment in which it operates. Fanping, Z., Ling, L., Juan, L. & Xufa, W. (2009) describe an experiment where an 'EAI model is used for anomalies simulation. At the same time, we give an idea of introducing artificial intelligence technology and status feedback to extend the original model. So our method can monitor and control the testing process.' The experiment uses a knowledge base, created from existing manual methods to test for vulnerabilities in the system. This model base is then used by a machine learning algorithm, which generates test cases to be run on the SUT. Machine reasoning is used to determine the test cases that will be run on the SUT. Test cases are then executed, results are fed back and analysed. The algorithm then learns from subsequent test runs. This provides continuous feedback on how vulnerable the SUT and is to attack. The experiment required the researchers to input data from a range of knowledge bases in order to provide the data that the ML would need, demonstrating the need still for much manual preparation to create a working ML-Driven solution.

The Angluin's $L*$ machine learning algorithm is used to construct a Fixed State Model in the experiment outlined by Groz, R., Simao, A., Bremond, N. & Oriat, C. (2018) where they 'have tried to make the most of AI-inspired and machine learning methods to get behavioural models of blackbox software systems without resetting them.' These fixed state models, which are generated by machine learning are then used to test that the SUT

behaves correctly. The use of ML reduces the requirement for testers to create a model to test, the models themselves are automatically created and the heuristic process creates more complex models as it is executed over time. The researchers conclude that their initial investigation is successful enough to scale the approach up for enterprise software systems.

2.3.3 Branch Coverage

Branch Coverage is a type of testing that verifies that each available functional path through a system is behaving correctly. The ISTQB Glossary (c.2018) describes this as:

> Test coverage criteria requires enough test cases such that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) taken each way, true and false. It helps in validating all the branches in the code making sure that no branch leads to abnormal behavior of the application.

Tools such as EvoSuite and Randoop can be used to generate these tests which provide test coverage metrics of the SUT at a unit level. These are low-level tests which ignore business-level behaviour and are a type of White Box testing. As they provide a wide test coverage of the SUT they can take a lot of time and effort to execute, particularly if the tests are required to be executed as part of a CI process. To improve on existing static analysis tools Grano, G., Titov, T. V., Panichella, S. & Gall, H. C. (2018) describe an experiment where they 'take the first steps towards the definition of features and machine learning approaches able to predict the branch coverage achieved by test data generator tools.' They were successful in using Huber Regression, Support Vector Machine, Multi-layer Perceptron machine learning algorithms to predict the branch coverage that would be achieved by a given tool, which gives developers a better indication of how effective the tools would be before deciding to use them to test their code. This allows for more informed test planning, leading to an improved test strategy and higher quality software.

2.3.4 Mutation Testing & Fault Identification

Mutation testing introduces a fault into the SUT to check whether the current test cases running on the system are effective enough to identify potential issues Bowes, D., Hall, T., Harman, M., JIA, Y., Sarro, F. & WU, F. (2016). created a study where they '*used four different classifiers (i.e., Na¨ıve Bayes, Logistic Regression, J48, Random Forest) which*

cover a range of different techniques' to create a model which improved on methods of fault detection. The benefit to software development is the identification of defects and faults in a system earlier and provides a tool which enables developers to establish how prone the system is to faults through the collection of metrics generated from their experiment. Bowes, D. et al found '*that mutation-aware fault prediction can outperform traditional fault prediction, for a range of predictive modelling machine learning algorithms.'*

Khosrowjerdi, H., Meinke, K. & Rasmusson, A. (2018) have devised another approach using Fault Injection (FI) which uses Learning Base Testing which 'combines: (i) machine learning to reverse engineer software models, and (ii) model checking to construct test cases that evaluate formalised safety critical software requirements.' The test method uses the machine learning Angluin's L∗ Algorithm to create a predictive model-based test strategy that aims to further ensure flaws found in critical systems such as the software systems used in the automotive industry are captured effectively. Khosrowjerdi, H. et al state that 'It was successful in finding previously unknown anomalies around safety requirements.'

2.3.5 Stabilising and Prioritising Existing Test Cases

A criticism often directed at test automation is that the tests can become unreliable after the SUT has been changed over time. 'Flaky' tests are automated which can provide both a fail and pass under the same test conditions, on the same SUT. This can happen for several reasons, for example:

- Tests might rely on an element to appear within a certain time frame and fails if it does not. That element could appear within the time frame under one test and appear outside of the time frame in a second execution run, producing a different result.
- Tests might be impacted on data that wasn't cleaned up properly from the first run or tests that are running in parallel from a different execution run.

King, T M., Santiago, D., Phillips, J., Clarke, P. (2018) have conducted research that 'fits into the category of AI for testing. It proposes a supervised ML approach that uses Bayesian networks to predict flaky automated tests. Flaky tests were identified and analysed. Metrics were associated with the causes of the 'Flakiness'. Tools were used to gather data on the flaky tests. Human experts were then used to classify data, and train a

model which then incorporated machine learning to develop the model which could be run on other systems, to identify flaky tests. The experiment was successful in that it proved that the approach was feasible, but further work was needed to provide a fully working solution. As with other experiments effort was required to define and input the data for the algorithms to learn from.

Tornhill, A. (2018) conducted an experiment using Codescene which 'combines repository mining, static code analysis and ML to prioritise potential code improvements based on the most likely return on investment'. This tool is intended for application-code analysis. The experiment was to assess the tool's effectiveness to identify issues with existing test-code. The tool is used to identify, through machine learning, the highest priority code that requires refactoring, based on development activity and behavioural analysis. The experiment used Microsoft's open source project ASP.NET MVC. By Using Codescene and machine learning the researcher was able to prioritise 2,501 from over 300,000 lines of code for refactoring.

Regression testing for large systems can be expensive and time-consuming. As testers usually have no visibility on what has changed in the code it is difficult to prioritise what tests to execute. This necessitates running all regression tests to provide confidence that potential defects will be detected. Lachmann, R., Nieke, M., Seidl, C., Schaefer, I. & Schulze, S. (2016) proposed an experiment in which their focus was on 'prioritizing test cases according to the decisions made by a test expert imitating the expert's behavior using ML.'

The experiment followed five steps:

- **Training Data Selection:** Creation of training data for the Machine Learning algorithm to build on
- **Dictionary Creation and Meta-Data Collection:** Creation of a dictionary that contains all-natural language words in test cases used in regressing testing the system. This dictionary is used to identify patterns to be used in the prioritisation model.
- **Classification Learning:** Machine learning algorithm SVM Rank is used to learn a classification model
- **Test Case Prioritization:** Using the Classification Model test cases for execution. Manual prioritisation would take a lot of time, but as Lachmann R. et al suggest,

'Our approach improves this by automatically prioritizing large number of test cases in reasonable time.'

- **Execution and Update:** The test cases are executed during the Regression phase and new Classifications are learnt, if necessary, depending on changes to code and the decision of test experts.

The experiment found that through the use of ML-Driven techniques they were able to identify the highest priority test cases to run rather than executing all of the tests or tests at random.

These experiments have proven that machine learning can be used to improve software testing for a range of techniques. Several algorithms and algorithm types are common across the experiments such as Angluin's L* and Support Vector Machines suggesting a convergence of hypotheses. The experiments required a high level of understanding to apply machine learning effectively, which perhaps the average software development organisation would not typically possess. For machine learning to be adopted widely in software testing, tools and services need to be made available which can make the benefits accessible and reduce the requirement for software test professionals to have a deep understanding of the complex underpinning mathematical concepts.

2.4 Commercial Availability and Adoption

How are commercial ML-Driven automated software testing tools different from traditional non-ML automated tools?

Digitalisation, or Digital Disruption is impacting public and private organisations across virtually all industrial sectors, which was implied by the growth suggested in Figure 2-2. Organisations are impacted by a growing need to incorporate digital solutions into existing business models, while providing more technologically advanced products and services, or risk obsolescence. Effective software testing is a fundamental requirement to ensuring the vision of industry leaders is met with the reality of the end-product. Machine learning has the potential to be an effective instrument for improving software testing techniques and tools.

Machine Learning algorithms are complex to implement which has made their integration into software processes prohibitive. If ML becomes easily accessible and readily available as a tool in the field of software testing then it would represent an inflexion point, providing

a powerful improvement to current automation practices and disrupting not only software testing, but the entire industry of software development.

Software automation tools which incorporate machine learning algorithms offer a range of advantages:

- Automatic Generation of Test Cases
- Self-Healing Tests
- Improved Continuous Integration Capability
- Improved Automatic UI Testing Capability

For these benefits to be realised it is required that machine learning can be implemented cost-effectively. One of the severest potential impediments to its adoption is the high level of skill required to leverage machine learning. Leaders in software testing currently would not possess the knowledge to leverage an ML-Driven test strategy without the assistance of a service or tool which abstracts the benefits and provides a cost-effective solution that can be adopted at an enterprise-level. This is reflected by Van De Ven, T., Dupjan, G. & Mamnani, D. in the World Quality Report 2018 which suggests that the

> 'traditional tester is no longer adequate, as working with AI requires professionals with a diverse range of competencies such as testing, mathematical optimization, neuro-linguistic programming, AI, business intelligence skills and algorithmic knowledge. At present, finding this combination of skills is difficult and experts suggest that challenges regarding the availability of qualified professionals will increase in the future as more organizations start experiment with AI'

This necessity has been identified by a number of vendors of software testing services who are positioning themselves to fill this market gap. These services have benefited from the evolution of machine learning, its move towards mainstream adoption and its growth in accessibility. The Price Waterhouse Cooper (2018) finds that 'AI is becoming more user friendly. Users no longer need to know how to write code in order to work with some AI applications. But more still demand far more technical knowledge than a spreadsheet of word processing program does.' ML-Driven test solutions are offered via a Software as a Service (SaaS) platform in which test cases are designed by an organisation's testers, then uploaded or programmed into the vendor's system which then uses ML to automate, execute and learn from this input. These services offer a range of valuable advantages.

2.4.1 Test generation

A prerequisite of standard automated testing is a level of design done by testers. The requirements of the SUT are analysed and test scripts are written that ensure the correct functional and non-functional behaviour, check for system compliance and identify defects. The design of these test cases will determine how effectively the SUT is tested. This involves a high-level of understanding from the tester to verify that system behaves as expected and to validate that business requirements are met. ML-Driven tools offer the capability to learn from the test cases uploaded by the testers to generate new test cases based on the behaviour patterns indicated by the uploaded test collateral. One ML-Driven test service, Appvance IQ (c.2018), describes their tool as using 'machine learning and cognitive generation to instantly produce thousands of scripts based on a thorough mapping of the application and an analysis of actual user activity'

2.4.2 Self-Healing

In software test automation one of the drawbacks is the cost of maintaining automated scripts. Black Box, front-end automated test cases commonly use screen elements with which to recognise the location of the controls to be used to execute a test script. If these elements are changed by updates to the SUT then the automation fails, and tests require a re-mapping exercise, done manually by testers. This can prove costly and time-consuming. If the automation approach adopted proves to be too brittle, then the benefit of these automated test may not outweigh the cost. If the root cause analysis of a failed test case too often indicates an issue with the test-code rather than a valid defect produced by the application-code then the viability of the automated tool is quickly brought into question as business users lose confidence in viability of the tool.

Self-healing is achieved by ML-Driven tests through the use of machine learning, which uses historical data to extrapolate information to identify the corrections needed to adapt to changes in the SUT over time. Functionize (c.2018), another provider or ML-Driven test services, describes how their tool uses machine learning for self-healing: 'ML models are used to generate scores characterizing the confidence level in element selection and action correctness. These scores are used by the expert system to help detect failing actions. ML models are also used to identify root causes by learning what successful actions look like over time, and using this training to identify failed actions'

2.4.3 Test Execution and Continuous Integration

Automated test scripts can be executed in parallel to simulate a large set of users, which increases the test coverage that can be covered by manual testing depending on the number of instances of the application that can be generated. This execution strategy requires hardware resources, usually achieved through a set of virtual machines. Selenium Grid for Selenium 2.0, for example offers the ability to execute automated scripts on a large set of environments, which allows for scaling of execution and provides faster feedback of test results. The Selenium Grid enables parallel execution on a range of different platform configurations, as Figure 2-3 illustrates. This solution can be costly to maintain and usually incorporates the services of other support teams such as System Administration or Development Operations (DevOps), depending on the organisational structure. Continuous Integration solutions require the capability to execute automated test cases, continuously, usually nightly, in order to verify that new code check-ins have not introduced any new defects. This rapid feedback is a characteristic of modern Agile development processes.

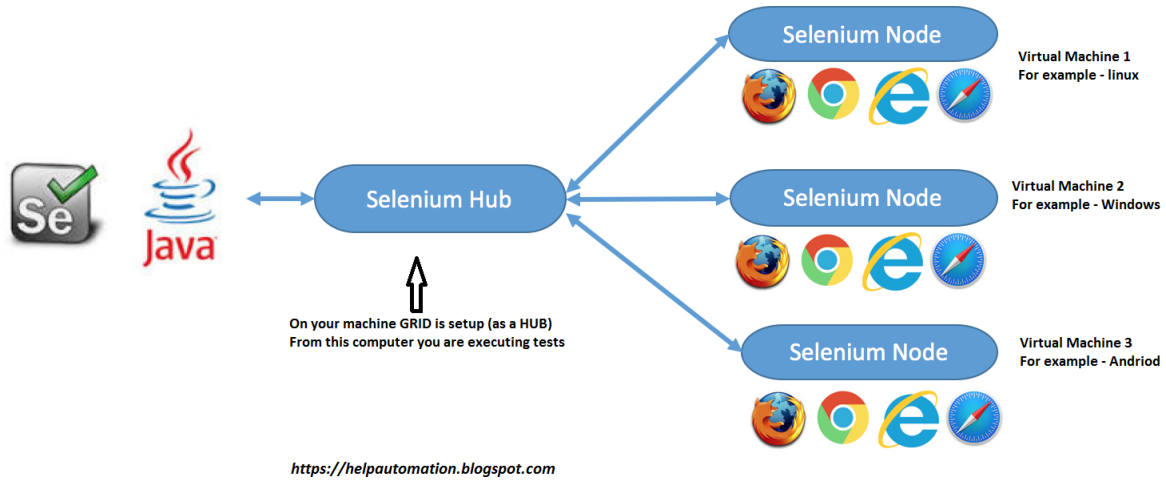

FIGURE 2-3 SELENIUM GRID. SOURCE: NARKHEDE, P. (2017)

Infrastructural costs relating to maintaining large test environments are reduced significantly by ML-Driven services. Test execution occurs on cloud-hosted servers. Test cases can be executed in parallel on the same environments where ML algorithms are employed to learn from the resulting patterns of output. ML-Driven test service Testsigma allows customers to 'Make use of Testsigma Lab to run your tests on thousands of

devices with different configurations available on the cloud.' This capability is offered by all of the services researched during this investigation (described in further detail in Chapter 5), and reduces the cost of simulating test users on the organisation's network and offers rapid feedback from the execution of large sets of automated test cases.

2.4.4 UI Testing

The inability to automate UI tests is a limitation of standard automation tools. While automated tools can recognise that an element exists it has been difficult to use an automation tool to verify that a screen looks right to an end user. Human confirmation has been the only reliable way of ensuring that the aesthetic of the application has not changed. Software containing a proliferation of UI defects becomes difficult to navigate, loses intuitiveness and draws many complaints from end-users. Several ML-Driven services offer the capability of automated testing to verify that the user interface has not been compromised by updates to the SUT. The Applitools (c.2018) solution describe their method of doing this: 'Baseline images define the expected appearance of your app at each step of the test. AI-powered computer-vision algorithms instantly detect and report any difference found between screenshots and baselines. By emulating the human eye and brain, our algorithms only report differences that are perceptible to your users.'

2.5 Human Impact and Five-Year Forecast

How will the rate of adoption impact on professional software testing from both a human and business perspective within five years?

Automated test approaches have grown in sophistication, effectiveness and usability since the first real experiments began during the 1990s. This evolutionary journey of test automation is illustrated in Figure 2-4. This timeline indicates that the future of testing belongs to ML-Driven solutions. ML is a growing trend in the automated software testing industry, but its capabilities are still limited (as demonstrated in Chapter 4). Machine learning in software testing is still a relatively new innovation and in an early stage of adoption.

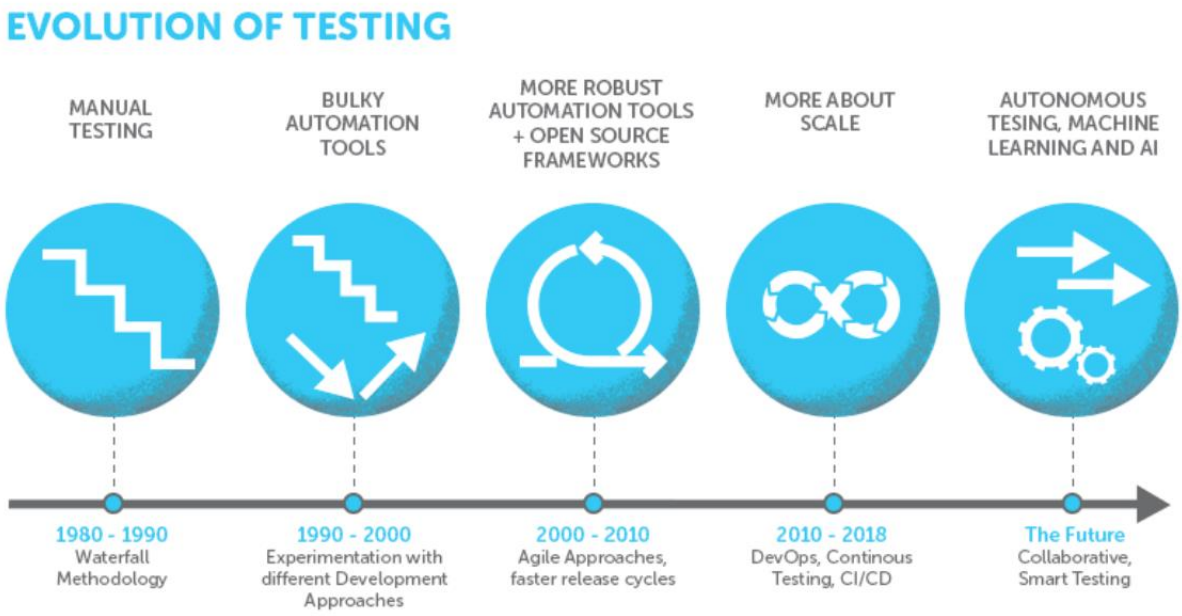

FIGURE 2-4 EVOLUTION OF TESTING. SOURCE: SUBRAMANIAN, R (2018)

This evolutionary progress of automation is represented by different levels described by Appvance (c.2018):

- Level 0 is manual testing, still 70% of the testing today.
- Level 1 is scripting at around 25% of testing. This methodology was introduced in the mid 1990's.
- Level 2 is codeless script generation or recording at around 5% of testing today.
- Level 3 introduces the first hints of machine learning such as self-healing scripts or monkey bots. Appvance was the first to introduce self-healing scripts in 2016.
- Level 4 is automatically generating scripts with no human intervention, scripting or recording. Appvance announced Level 4 in 2017.
- Level 5 is self-generating tests which are able to validate complex actions or data or elements generally at or above human capabilities and do so autonomously.

Similarities between both representations of the evolution of automated software testing suggest that automation is still predominantly designed and maintained by humans and that the Level 5 definition of self-generating tests is yet to be realised by commercial offerings within the Software Testing as a Service industry. Software test automation in even its simplest form has yet to be integrated into the majority of practices, as has been reported by Price Waterhouse Cooper (2018) who discovered that a mere 15% of all end-

to-end business scenarios carried out by survey respondents were being executed using test automation tools. Appvance suggest that only 30% of testing is automated.

Reviewing the disadvantages and challenges of automation reveals some of the reasons for the low adoption rates of test automation tools. These are listed by Ghahrai, A. (2018) as: 'False sense of quality, Not reliable, Automation is not testing, Maintenance Time and Effort, Slow feedback, Not many bugs found.'

Sharma, P. (2017) identifies 'ROI comes late, More skilled Testers are Required, More Maintenance is involved, Automation is not Testing, Unrealistic expectations from the tool' as the main disadvantages.

Throvagunta, S. et al (2018) collected data from 1700 executives across 10 different sectors. The challenges that were most commonly reported is represented in Figure 2-5.

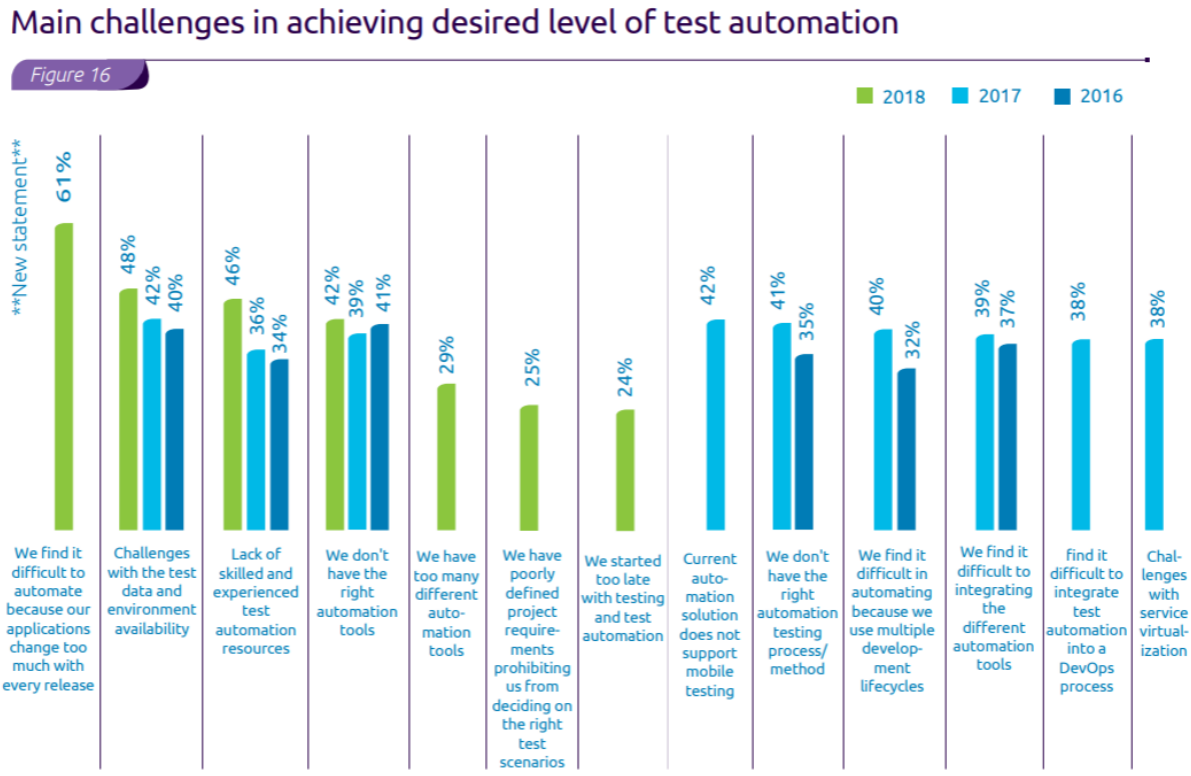

FIGURE 2-5 CHALLENGES IN AUTOMATION. SOURCE: WORLD QUALITY REPORT (2018)

Before the adoption levels of test automation grows these challenges and disadvantages need to be addressed. The introduction of machine learning into test automation can start to provide solutions to these issues:

Maintenance:

A common drawback cited is the cost of maintaining scripts. Changes in the SUT result in automated tests breaking due to updates to the underlying structure of the system, resulting in necessary modifications to existing automated code. Machine learning offers a 'Self-Healing' capability to counter this, characterised by algorithms that can identify when a test should be updated to align with changes to product code.

Lack of Skill and Experienced Test Automation Resources:

Services offering ML-Driven automated tools offer the ability to automate tests without the need for coding automating tests. Tests can be written in natural language and uploaded to the SaaS cloud. This removes the need for testers to be able to code, reduces the need for test professionals to understand coding practices and enables testers to focus on design and test analysis instead. However, with ML comes the need to understand to an extent what the ML algorithms are capable of. There will still be the requirement for testers to verify that the ML tools are returning the correct result, at least at the initial phase. ML-tools have the capability to generate large sets of test scripts. Testers will need to be able to decipher large results sets on a continuous basis.

Slow Feedback:

The time involved in scripting automating tests can mean that the value of those test scripts is not realised until late in the project. Manual testing is still needed to fill this gap to ensure new features are tested effectively while automated tests are still in development. ML-Driven tools offer a solution to this, whereby test cases can be generated once test cases are input that the algorithms can learn from. This however, still suggests that an initial phase is required whereby test cases are uploaded. Feedback might still be slow, depending on the complexity of the SUT and suitability of the test cases to be used as a basis for learning.

Challenges with the test data and environment availability:

Effective test automation in an Agile environment requires Continuous Integration environments, which still need to be managed, usually by a DevOps team. Test services will still need the organisation to provide an environment to execute test cases on. This cannot be solved by test automation, and needs investment in a fit-for-purpose CI solution.

Automation is not Testing:

Non-ML driven automated tests are only as effective as their design. There is still a need for testers to understand the business requirements and desired functionality of the end software product. As Validation requires a level of subjectivity it is still best suited to manual testing whereas Automated Testing is ideal for the objective static evaluation associated with test verification. Verification can be carried out by automated tools, but only within the test coverage in which they have been coded. Exploratory testing is still required by knowledgeable testers to carry out initial testing whereas automated testing is suitable for regression testing of existing functionality to ensure future releases of the software do not introduce defects. The Appvance model and Testim descriptions of the growth of software both suggest that ML-Driven has not arrived yet at a place whereby complex test cases can be generated without the need for manual testing to run complementarily.

Unrealistic expectations from the tool:

The challenges outlined above of not having the right tool, having too many tools to choose from and having unrealistic expectations from the tool, suggest a disconnect between the practical reality of what can be achieved by automation and what business leaders believe to be the potential of automated testing. While integrating machine learning with software testing does offer more powerful capabilities to software testing there is still a long way to go for it to be intuitive enough to replace testers altogether. Test Validation is still a requirement which is supported by a human interpretation of system and business design documentation. Test cases will still need to be designed by testers and results will still need to be analysed by humans. Limitations will need to be understood upfront by decision makers for ML-Driven tools not to be dismissed when unrealistic expectations are not met.

As further Digitalisation puts pressure on organisations to software solutions growing in complexity, at faster rates the feedback provided by test automation will speed up at a proportionate rate as illustrated in Figure 2-6. Platz, W. (2017) suggests that ‘AI, imitating intelligent human behavior for machine learning and predictive analytics, can help us get there.’

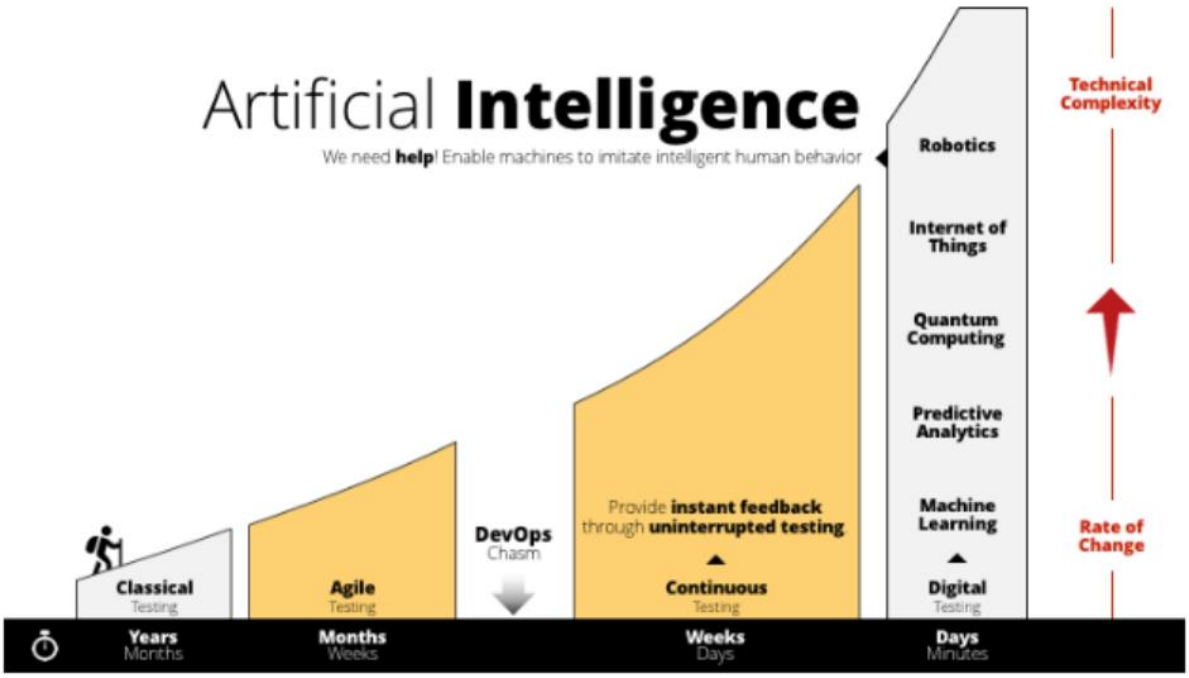

FIGURE 2-6 GROWING REQUIRED SPEED OF TESTING. SOURCE PLATZ, W.(2017)

This will require ML to evolve in technical sophistication while also improving in usability, maintainability and portability. The literature does reflect that the key to the future of test automation will be in machine-learning, but before ML-Driven tools are widely adopted there is still a gap that needs to be bridged between what's expected and what's available.

2.6 Conclusion

How is machine learning (ML) being used to impact Software Testing techniques?

The details of several experiments were available which sought to leverage machine learning to improve software test techniques. All experiments produced successful results, where the results indicated a viable improvement in software test techniques or provided the confirmation required to proceed to further investigation.

Experiments covered a range of machine learning algorithms. The same machine learning algorithms were used by more than one experiment, inferring that there are algorithm types that are more applicable to software testing, such as Angluin's L* and Support Vector Machines.

The experiments were characteristically complex, requiring a deep understanding of machine learning. The experiments invariably required a significant level of data set-up to

build the knowledge that the algorithms required to start learning. For the integration of these machine-learning techniques to be scaled up to an enterprise-level further research is required. Whereas they provided proofs of concept, few described a solution that could be adopted by an average software development organisation.

How are commercial ML-Driven automated software testing tools different from traditional non-ML automated tools?

Commercial tools are offering a range of features, including self-healing, automatic test-generation, out-sourced test execution capability and UI testing. These features can potentially remove some of the most common obstacles preventing software development practices from adopting automated testing.

The cost of maintaining broken tests should reduce. More straight-forward test creation should eliminate the need for testers to create code. Outsourcing test execution to a 3rd party cloud should reduce the cost of running large numbers of test cases concurrently on a nightly basis. Effective automated UI testing can identify usability defects in the system faster, improving end-user experience and customer satisfaction levels. Chapter 5 discusses the ability of current ML-Driven services to deliver on these potential benefits.

How will the rate of adoption impact on professional software testing from both a human and business perspective within five years?

These services are still in the early stages of development. Machine learning appears to be the next step in the evolution of software test automation, rather than the state-of-the-art. Obstacles still remain. Many standard automated practices use open source tools such as Selenium. The ML-Driven tools currently available incur a cost and require a subscription. ML-Driven services require tests to be stored in the cloud, and require access to internal networks, which can potentially create a security risk. Test cases can be generated, but require an initial knowledge base to learn, which still requires human input. Self-healing test will still need to be verified as healing 'correctly', for an initial period at least, until confidence grows in their ability to successfully distinguish between a defect and a broken test. It will take a number of successful case studies before business leaders begin to take the value of these services seriously. Given the current low adoption rates of standard automation it seems unlikely that these tools will disrupt the software testing profession significantly within five years. Widespread adoption will likely take longer.

3. Chapter 3: Methodology and Fieldwork

3.1 Introduction

This chapter explains the Research Methodology and Fieldwork that was undertaken to gather relevant data to explore the topic further and build on the knowledge gained from the literature review. Included is a discussion on the philosophical concepts which underpin the two approaches adopted to address each sub question. Saunders, M., Lewis, P. & Thornhill, A. (2009) compare research to the structure of an onion with data collection techniques and procedures at its core. This structure is displayed in in Figure 3-1.

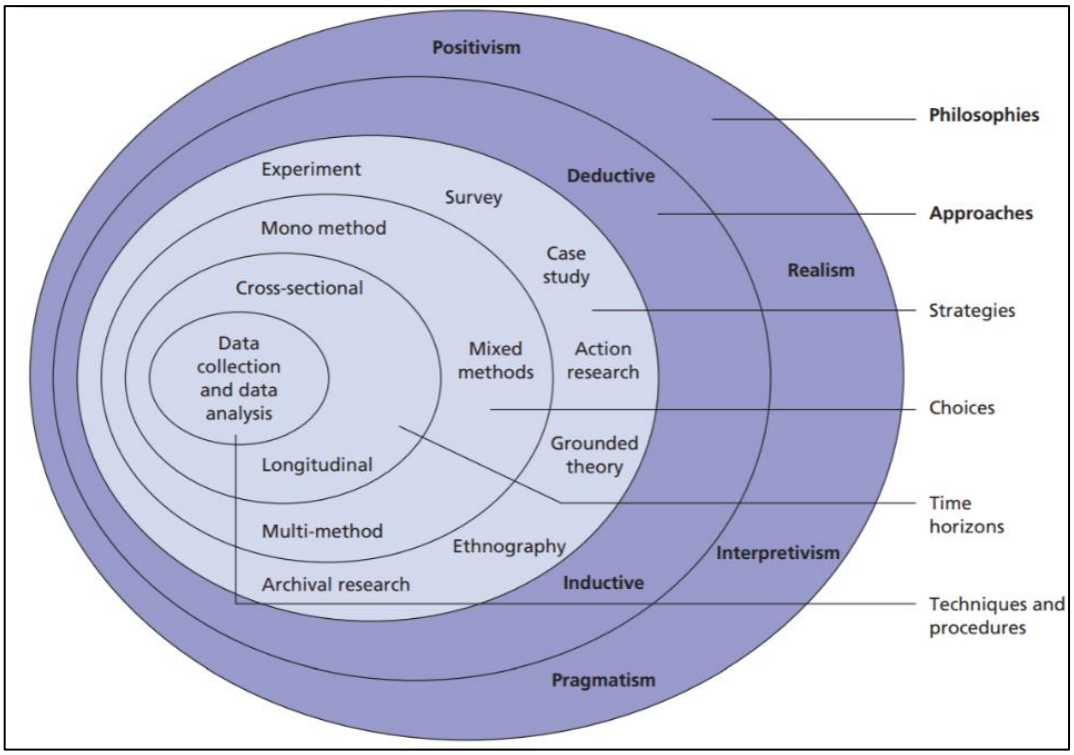

FIGURE 3-1 THE RESEARCH 'ONION'. SOURCE: SAUNDERS ET AL (2009)

This chapter discusses the research methodology within the context of this structure.

3.2 Philosophy

Saunders M. et al (2009) describe three branches of philosophy which inform the approach the researcher will take to designing the tools and methods of data collection in order to gain insight into the topic under review.

3.2.1 Ontology

Ontology is made up of two aspects of the nature of reality which Saunders M. et al (2009) describe as

> The first aspect of ontology we discuss is **objectivism.** This portrays the position that social entities exist in reality external to social actors concerned with their existence. The second aspect, **subjectivism,** holds that social phenomena are created from the perceptions and consequent actions of those social actors concerned with their existence.

3.2.2 Epistemology

Epistemology is the branch of the philosophy which describes what constitutes knowledge as it relates to the research topic under review. Three distinctive perspectives are discussed below which influence research design decisions.

Positivist philosophy

Positivism suggests a research approach that would include a strict measuring of quantifiable data from which a set of behavioural rules can be extrapolated. Positivism deals with measurable facts rather than subjective opinions gained from human interrogation. Positivist approaches are more likely to measure statistics and gather clear metrics to establish knowledge.

Realism Philosophy

Saunders M. et al (2009) maintain that 'Realism is a branch of epistemology which is similar to positivism in that it assumes a scientific approach to the development of knowledge.' They go on to describe two branches of realism. The first is **Direct Realism** which assumes that social phenomena exist independently of human senses. Data collected is not affected on how our senses interpret it, that it exists unchanged by the perception of our senses. **Critical Realism** suggest that there are two steps in establishing information about the world. The first is measuring data while the second is how our minds interpret this data. It is this interpretation that forms our understanding and is the reality which we exist within.

Interpretivist philosophy

Interpretivism emphasises the significance of the people as 'social actors'. As social actors people play a distinctive role in establishing, creating and reshaping the social entities in which they operate. Empathy plays a vital role in interpretivist research, which would include, for example, conducting surveys and interviews to establish meaning.

3.2.3 Axiology

Axiology is described by Saunders M. et al (2009) as the 'branch of philosophy that studies judgements about value'. This is significant in designing research as it must be decided how to measure value and what is of value to the research. For this research question 'How can Machine Learning be Leveraged for Software Testing within a Timeframe of Five Years?' we must establish what value machine learning can provide to software testing and how to measure this value. There are two separate value sets to consider when considering this question.

During this investigation the value of ML-Driven tools is assessed through semi-structured interviews of leaders in the field of software testing. These interviews establish value as it is perceived by the target users of automated software tools. The information gained from theses interviews are then used in an empirical analysis of selected ML-Driven tools available on the market. Through the comparison of the value sought and the value available from the tools the impact of machine learning on software testing practices can then be inferred.

Saunders M. et al (2009) suggest that **Pragmatism** argues that 'the most important determinant of the epistemology, ontology and axiology you adopt is the research question – one may be more appropriate than the other for answering particular questions.' For this research a pragmatic view was taken. Machine learning and software testing rely on social actors to design the frameworks and rules in which they operate. Due to the nature of machine learning it can be considered to have the potential to operate external to human intervention. However, it is the case that in order for machines to learn they must be prepared and taught by engineers beforehand. Machine learning can continue without the further need for human input but any advancement is an expression of the initial data set, input by humans. To understand the impact of machine learning on software testing the benefit of it to humans needs to be assessed. The extent

of this benefit ultimately will drive the design and evolution of the impact of machine learning on software testing.

To provide answers to each sub-question a mixed approach was deemed most suitable. Observable phenomena were assessed, by empirically analysing a sample of available ML-Driven software testing tools while semi-structured interviews also incorporated the subjective views of leaders in the field of software testing.

3.3 Research Paradigm

Saunders M. et al (2009) describes the research paradigm as 'a way of examining social phenomena from which particular understandings of these phenomena can be gained and explanations attempted.'

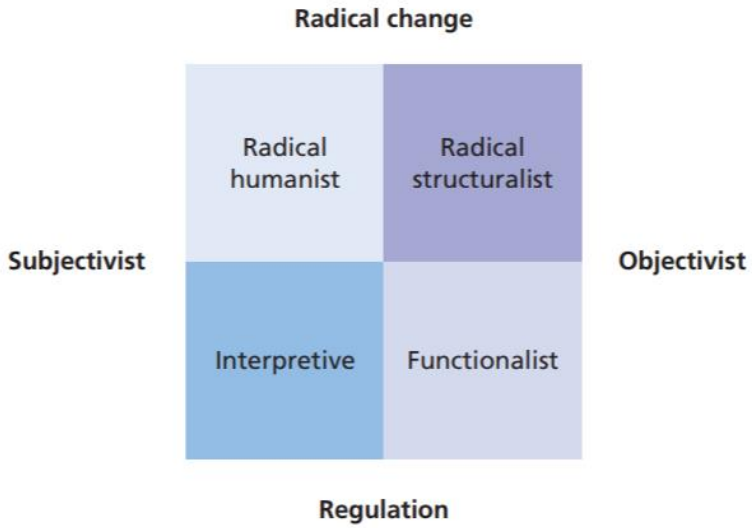

FIGURE 3-2 RESEARCH PARADIGMS. SOURCE: SAUNDERS ET AL (2009)

Functionalist paradigm:

Positioned within the Objectivist and Regulation dimensions the Functionalist paradigm aims to assess an issue, like machine learning and software testing by describing how current techniques and solutions are employed and researching what the impact of ML would be if an organisation were to adopt these emerging practices.

Interpretive paradigm:

Along the Subjectivist and Regulation dimensions is the Interpretive paradigm. Saunders M. et al (2009) identify the goal of this paradigm to 'understand the fundamental meanings attached to organisational life'. Adopting this paradigm for this research would require a study of human involvement, and shortcomings, when it comes to software automation and assess how machine learning within software testing would be accepted by the organisational culture in a software development department or company.

Radical humanist paradigm:

The radical paradigms are concerned with implementing significant change to the established practices in an organisation. Radical humanism resides on the Subjectivist and Radical dimensions. For this research to adopt this paradigm this could involve framing the failure to adopt machine learning into software testing practices as a symptom of the fear of change on behalf of software testers and managers. Overcoming this dysfunction would be a means to effect change within the organisation.

Radical structuralist paradigm:

The Radical structuralist paradigm is located within the Objectivist and Regulation dimensions. This paradigm is concerned with achieving radical change through identifying objective flaws in current business practice. From the perspective of this research this would attempt to research flaws in the organisational structure that underlie reasons for adopting current test strategies. The goal would be to create a case for adoption by describing solutions to these flaws through the use of machine learning.

The paradigm which this research is aligned with more closely is that of Functionalist. The research aims to establish how software tools and services that are machine-learning enabled compare to standard non-ML tools. As the research aims to establish the impact of ML on Software Testing it can be described as an investigation into the viability of the adoption of machine learning into the existing regulation of the software test function of an organisation.

The approach that was adopted for this research can be best described as relating to that of a Pragmatic philosophy. To obtain data to answer the three sub questions the design of the research was based on both semi-structured research Interviews and empirical analysis of secondary data. This multi-strategy is described by WILSON, J. (2010). as 'pragmatic research' in that the research does not attempt to 'fit' into any one paradigm,

but the researcher uses whichever methods he or she considers work best for their particular study'

3.4 Strategy and Choices

The semi-structured interview research was formed from the expert opinions of industry leaders in the field of software testing. Questions were designed to provide data which was then used to induce the impact of machine learning on current software testing practices. 'semi-structured – this is where you have a "theme" for your interview, and some carefully defined questions, but leave open the possibility of discussion of other cognate areas' – Emerald Publishing (c.2019)

A complementary empirical qualitative analysis was used on secondary data obtained through online literature and demonstrations of selected machine learning software testing tools. According to Saunders et al 'Qualitative data refers to all non-numeric data or data that have not been quantified and can be a product of all research strategies'. This was an Inductive approach, which is described by Dudovskiy, J. (c.2018) as

> 'beginning with a topic, a researcher tends to develop empirical generalisations and identify preliminary relationships as he progresses through his research. No hypotheses can be found at the initial stages of the research and the researcher is not sure about the type and nature of the research findings until the study is completed.'

As the research assumed limited knowledge of the tools under review an inductive approach was undertaken to gain a clearer understanding of the capability of the tools under review compared to a standard non-ML driven tool.

3.5 Approaches Overview

The two complementary approaches were used to answer the three sub-questions comprising the main investigation of this research. Criteria for the secondary data qualitative analysis of ML-Driven tools were aligned with questions asked of respondents in semi-structured interviews. This research was conducted over a cross-sectional time horizon. Both research approaches were conducted over a snapshot of three months.

Ethics approval was sought for the interviews carried out and informants were provided with an information pack which outlined a description of the research and included:

- A background of reasons for undertaking the research
- Criteria for selecting the participants
- The procedure for conducting the interviews
- A copy of the questions that would be asked
- A declaration of a potential conflict of interest, as the interviewer is also a leader in the field of software testing

Participants were asked to sign a consent form, which required them to understand that: *"I am participating in a personal capacity and not on behalf of my organisation."* This ensured that no sensitive information relating to the organisations in which they were employed was discussed.

3.5.1 Instrument Development and Analysis Approach: Semi-structured Review

The approach taken to semi-structured interviews involved the following steps:

1. The criteria for selecting candidates were that they had experience in the field of Software Test Leadership, were knowledgeable of software test automation and had a fundamental awareness of machine learning
2. Interviewees were known to the researcher and contact was initiated via LinkedIn
3. Interviews were conducted and recorded over Skype
4. Skype recordings were then transcribed by hand into word documents
5. The tool 'Dedoose' was then used to for qualitative analysis to identify patterns in the respondents' transcripts and create visualisations which aided in the interpretation of the results
6. Dedoose provides the user with the capability to breakdown each interview questions into categories and apply weightings to the responses to provide a metrics from which meaning can be derived from the responses.
7. Weightings of 1-10 were inferred for each response to quantitise the qualitative data, details of which are described in section 6 of this Chapter.

3.5.2 Instrument Development and Analysis Approach: Empirical Analysis

The approach taken to the empirical analysis involved the following steps:

- Five software ML-Driven software testing services were identified and selected from reviewing material from software product review websites and software tooling specialist sites online. These included:
    - Reviews for Software Test Automation. Gartner (c.2019).
    - 8 Innovative AI Test Automation Tools for the Future: The Third Wave Colantonio, J. (2017)
    - AI Driven Testing (AI-DT) open source tools. Stack Exchange (2018)
- An empirical analysis strategy was then created by using evaluation criteria and templates from the following sites for guidance:
    - Software Evaluation Checklist Tips. Strickland, A. (2019)
    - Reviews for Software Test Automation. Gartner (c.2019).
- The responses from the semi-structured interviews were also used to form the criteria for analysing the ML-Driven software testing services. Categories that respondents prioritised were added to the ML-tool analysis. This ensured both approaches were aligned for proper value analysis.
- Through rating the ML-Driven services in the areas that were a priority to the respondents, conclusions were then inferred as to the likely impact of the tools on the software testing industry.
- Data was gathered on each of the five tools through reviewing software demonstrations, whitepapers, training material and functional descriptions available online for each service.
- Selenium Web Driver, a standard (non-ML driven tool) was then used to compare both standard and machine-learning automated tools
- Dedoose was again used to quantitise the qualitative data which was then cross-checked with data from the semi-structured interviews

3.6 Detailed Approach

The following is a detailed description of the interview questions and empirical analysis approach. The rationale for each question and empirical evaluation is explained as it relates back to one of the three research sub questions.

3.6.1 Research Design: Machine Learning in Software Test Practice

The following are the interview questions and criteria under which were the services were assessed to answer the sub-question **'How is machine learning (ML) being used to impact software testing techniques?'**

Interview question 1		
Most organisations are still implementing a traditional software automation test strategy, if at all, where testers manually code and maintain automated scripts and use a tool like Selenium for organisation and execution. In your opinion, given the current maturity of Artificial Intelligence how can it be used in Software Automation?		
Rationale		
The capabilities of artificial intelligence and machine learning are relatively new to software testing. The rationale behind this question was to establish what the respondents considered as machine learning and their understanding of its maturity. Machine learning is a broad term so to initiate the interview it seemed pertinent to agree on an understanding of what machine learning meant to the respondent within the time horizon in which the interview was taking place. The question was designed to get an idea of how advanced the respondent thought machine learning was at the time. The respondents were selected because their roles required a high level of understanding of standard automation tools (i.e., automation tools that did not use machine learning), so they were well suited to provide an informed opinion as to the feasibility of aligning software test automation with machine learning.		
Response Groupings	**Description**	**Ratings**
1. Maturity of ML in Testing	The level of maturity which the respondents thought that machine learning had	0 = Low Maturity 10 = High Maturity

	researched in respect to its applicability as a test tool	
2. ML/ST Media Exposure	The extent to which machine learning tools had been discussed in industry related media and software test forums	0 = Low Coverage 10 = High Coverage

Interview question 2
There are now several AI-Driven Test Automation tools on offer in the market place. Has your organisation implemented any of them, or is it planning to? If not, why? If so, what was the business case for doing so and have you carried out a Cost-Benefit analysis? Do you think the service the tools provide is worth the cost?
Rationale
The aim of this question was to gain insight into if any of the respondents had any professional experience with AI-Driven test tools. The question was broken out into possible sub-questions which covered whether or not they had reviewed any tools and if they had what their opinion of them was, particularly from a cost-benefit perspective. When analysing these tools, it is important to factor in the cost of using them. While they might be powerful and offer a wealth of new features, they're impact on the software testing industry will be impeded if the cost of integrating them outweighs the benefit they offer. This lead on to the broader discussion of how the tools could reduce cost in other areas (such as the time and cost spend on coding test cases with standard automation tools) to offset the overall cost of the services on offer. During the literature review it was clear that machine learning could be used to improve software testing techniques on an experimental level, if the cost was too high then the machine learning wouldn't be used to improve software testing on an industry-wide scale.

Response Groupings	Description	Ratings
1. ML Adoption Rate by Respondents	The adoption rate of tools in the respondent's organisation, from not being aware of them at all to adopting them fully	0 – Not Aware of tools, has not looked at them 10 – fully adopted ML tool/service
2. Concern of Cost	The importance attached to ensuring the cost doesn't outweigh the benefits	0 – Does not mention cost at all 10 – Cost is of the highest priority
3. Reliability of ML Services	Concern of the reliability of the test cases that are generated from machine learning	0 – Does not mention reliability of generated test cases at all 10 – Reliability is of the highest priority
4. Security Concerns	Importance of Security when adopting ML Services which integrate into the system	0 – Does not mention security at all 10 – Security is of the highest priority
5. Ease of Integration	Concern expressed the rate of effort involved in Integrating a machine learning tool into existing infrastructure	0 – Does not mention integration at all 10 – Ease of Integration is of the highest priority

6. Cloud Integration	The Importance the respondents attached for the tools ability to integrate with cloud-based environments	0 – Does not mention cloud integration at all 10 – Cloud Integration is of the highest priority
7. Ease of Use	The importance attached to the ability of software testers and other end-users to learn the new tool	0 – Does not mention usability at all 10 – Usability is of the highest priority

Empirical Analysis:

Secondary Data Criteria Category 1		
Category: Machine Learning Maturity		
Criteria	**Data Capture**	**Ratings**
Company Maturity	How long has the Company which provides the service been in existence?	0 – less than a year 10 – over 10 years
Rationale		
The length of time that the company is in existence gives an indication of the current maturity of machine learning in software testing and provides an insight as to the likelihood that Software Test Leadership have been made aware of their services.		

Secondary Data Criteria Category 2		
Category: Machine Learning Adoption Considerations		
Criteria	**Data Capture**	**Ratings**
(Reliability) Quality of Generated Test Cases	Coverage and Readability of generated test cases	0 – no coverage or unreadable 10 – total coverage and easy to read
Cost of Tool	Cost of tool compared to Standard tool and compared across providers. Where cost information is not accessible a high cost is assumed	0 – Open Source 10 - $10,000+ per month
Ease for Testers to Learn Tool	Usability of tool, including the difficulty level in creating automated test cases. Where coding is still necessary a high difficulty is assumed	0 – High Difficulty. Testers will need to speed a lot of time on test creation 10 – Represents a vast reduction on time spent to create automated test cases compared to standard automation
Ease of Integration Set-Up	How easy it is to integrate with an organisations infrastructure (compared to standard tool), including Continuous Integration	0 – Easy to integrate 10 – Prohibitively difficult to integrate

	pipelines and Defect Tracking tools	
Ease of setting up Cloud Environment Integration	An area that was mentioned in the interviews was the ability of the services to integrate with an organisation's own Cloud Test Environments	0 – Easy to integrate with Cloud environments 10 – Prohibitively difficult to integrate
Quality of Analytical Tools	The quality of analytical tools, including Root Cause Analysis data and monitoring tools	0 - Tools are of no use 10 - Highest quality reports, etc.
Quality of Support and Training Material	The quality of training material available, including tutorials and customer support.	0 - Low quality 10 - High quality
Security Threat from Tool	Considering the network access that the service will need what threat do they pose if their system was compromised maliciously. Another consideration is what access they will have to sensitive Intellectual Property such as source code and business test cases	0 – No Security Risk 10 – Prohibitively high level of risk to the organisation
Rationale		

This category provides data on categories that software test leadership would consider when adopting machine learning services that are not directly related to the features they offer. This provides insight into areas such as cost, risk and complexity of integration compared to benefits

3.6.2 Research Design: Machine Learning & Commercial Availability

The following are the interview questions and criteria under which were the services were assessed to answer the sub-question '**How are commercial ML-Driven automated software testing tools different from traditional non-ML automated tools?'**

Interviews Questions:

Interview question 3
Given the potential of AI software test tools what specific features do you think could impact on your current Test Strategy and what benefit would you like to see. For example, do you think they will make it easier for software testers to automate test cases? Do you think that these tools will enable faster execution times, and therefore faster product development? Do you think these tools will be difficult to integrate with existing infrastructure? Are there any other impacts you can think of, both positive and negative?
Rationale
This question was designed to gain an understanding of what value the respondents would like to see from ML-Driven tools. Their experience of standard automation gave them a practical grasp of the limitations of standard tools so they were in a good position to give an informed opinion of what areas they would like to see improved by ML-Driven tools. This information was made valuable when compared to actual capabilities of the tools, which were assessed during the empirical analysis of the tools themselves. Criteria such as speed of execution, ease of implementation, ability to

generate test cases, were discussed. Negative aspects such were also discussed, such as the potential skill shortage that might arise from software testers a requiring working knowledge of underlying machine learning concepts, or the need to analyse large sets of results produced by increased test coverage.		
Response Groupings	**Description**	**Ratings**
1. Automatic Test Case Generation	Importance attached to Automatic Test Case Generation capability	0 – Does not mention automatic test case generation 10 – Automatic Test Case Generation is of the highest priority
2. Reduced Maintenance Effort	Importance attached to the reduction of maintaining automated scripts	0 – Does not mention Maintenance 10 – Maintenance is of the highest priority
3. Quick Feedback/Execution Times	Importance attached to the ability of the tool to execute large sets of test cases continuously, as part of a CI process	0 – Does not mention Test Execution/CI 10 – Test Execution/CL is of the highest priority
4. Automatic UI Testing	Importance attached to the ability of machine learning to automate UI Testing	0 – Does not mention UI Testing 10 – UI Testing is of the highest priority

Empirical Analysis:

Secondary Data Criteria Category 3		
Category: Machine Learning Features		
Criteria	**Data Capture**	**Ratings**
Automatic Test Case Generation Ability	Can the services generate test cases automatically without the need for human intervention?	0 – No ability 10 – Test cases are generated with no human intervention or design required
Reduced Maintenance Effort	Is there reduction in the effort needed to maintain automated costs once the underlying SUT has changed?	0 – Maintenance is as high as standard tools 10 – No Maintenance cost
Quick Feedback/Execution Times	The ability of Tool to execute large sets of test cases continuously. Can test case be run on the service's cloud environment, quickly and without the need for hardware to be dedicated from the organisation's network for execution	0 – No test execution ability 10 – High volumes of test cases executed at a significantly improved rate

Automatic UI Testing	Ability for automatic UI testing explicitly mentioned and described.	0 – UI testing not even implied 10 – Automatic UI testing fully available
Rationale		
What features are the tools offering and how can they be of benefit? Can they generate more efficient tests, reduce time in test creation, reduce cost of maintenance and expand code coverage?		

3.6.3 Research Design: Human Impact & Five-Year Forecast

The following are the interview questions to answer the sub-question **'How will the rate of adoption impact on professional software testing from both a human and business perspective within five years?'**

No empirical data was available for this question as its nature is subjective and requires a certain amount of speculation. However, conclusions can be inferred from the data collected in the above sections.

Interviews Questions:

Interview question 4
What impact do you think the introduction of AI software test tools will have on the Software Test profession (human-impact). Do you think it will lead to a reduction in the number of Software Testers employed by IT organisations? What positive impacts do you envisage for Software Testers? What negative impacts do you envisage for Software Testers?

Rationale		
This question was designed to discuss the impact of ML on the software profession from a human perspective. There is the possibility that ML will reduce the number of software testers required by a software organisation due to the capability of the new tools to generate test cases without the need for human design and intervention. For testers this could have a negative impact if this resulted in significant layoffs. There is also the possibility that ML-Driven tools will reduce the requirement for software testers to code automated test cases, freeing them up to adopt more creative tasks, such as higher-level planning and test design, or more in-depth test analysis. The role of the software tester might also be impacted by the need to grasp new skills such as data analysis, and machine learning understanding.		
Response Groupings	**Description**	**Ratings**
1. Probably of Reduction of Tester Roles	The likelihood of a significant reduction in Software Tester roles due to the introduction of ML	0 – Not likely tester roles will be reduced 10 – Extremely likely of test role reduction
2. Career Advancement	The likelihood of a positive impact on the Software Testers role through upskilling in ML-related tools	0 – Not likely that impact will be positive 10 – Extremely likely that test role will be improved through up-skilling
3. Improved Creative Freedom	The likelihood that ML will free Software Testers up for more creative tasks, providing better job satisfaction	0 – Not likely that testers will be freed up for more engaging tasks

		10 – Extremely likely that testers will be freed up for more engaging tasks

Interview question 5		
How do you see AI evolving within the Software Test profession in the next 5 years? Do you believe that AI software test tools will gain traction in the market? Do you believe that there will be an inflection point whereby organisations will adopt AI tools across the industry, replacing much of the current non- AI software test tools?		
Rationale		
This set of questions were designed to reveal how the respondents predicted the adoption rate of ML-Driven tools would look within a timeframe of 5 years. The adoption rate of standard automation is still usually reported to be 30% or less in software organisations. There are a number of factors to consider when discussing this question. Many organisations abandoned software automation due to drawbacks like high maintenance costs or the high-skill involved in automation. If this perception manifests as a mistrust of all automation then this will hamper the rollout of ML-Driven tools. However, if it can be proven that AI can address these impediments to adoption then the uptake may be rapid. As leaders in the software industry, the respondents were well-informed of developments in the industry through regular reports, seminars and other related material. This question discussed whether information on ML-Driven tools was being made available via these channels, and to what extent. This would indicate the market share and current adoption rates of the tools currently.		
Response Groupings	**Description**	**Ratings**

1. Probability of Widespread Adoption in 5 Years	Confidence that ML-Driven Tools will be used by a significant number of organisations in the next 5 years.	0 – ML Tools will not be used significantly in 5 years 10 – ML Tools will definitely be used in significantly in 5 years

3.7 Problems Encountered & Lessons Learned

3.7.1 Interviews

One interview candidate dropped out due to time constraints.

3.7.2 Empirical Analysis

For the empirical analysis of the machine learning tools it was assumed that trial versions of the services would be accessible and suitable. It became apparent that machine learning was invariably offered as a service rather than a standalone tool. Whereas standalone tools can be downloaded and assessed in isolation, services require third party access for trial runs.

The original plan was to trial these tools using the researcher's employer as a test subject. The tools would have been used to automate test cases and execute them on the employer's systems. Results would have then been compared to the standard tool used by the employer. Permission was granted to do this. However, access to these trial versions would have required introducing a security risk for each service under investigation. This five-fold security risk was deemed too high to progress with and as a consequence the research was altered from Quantitative Analysis to Qualitative Analysis, which relied on secondary material available from online sources.

3.8 Conclusion

The research was shaped by a pragmatist philosophy, which was determined by the nature of the investigation. The Functionalist paradigm was most appropriate to the investigation as it does not seek to interpret machine learning as a radical change to software testing but aims to assess it as a viable improvement to existing processes.

Research design encompassed a complementary two-dimensional approach that aimed to establish the value of machine learning as perceived by potential users compared to the observed value offered by machine learning services. These two qualitative methods provided the data required to establish the impact that machine learning will have on software testing within a timeframe of five years, as can be seen from the findings in Chapter 4.

4. Chapter 4: Findings and Analysis

4.1 Introduction

This chapter introduces the results of the data collection exercises from the semi-structured interviews and the empirical analysis of machine-learning tools. Relevant excerpts from the four semi-structured interviews are included in addition to information acquired from the empirical analysis. Data is discussed as it pertains to each sub question. Dedoose was used to analyse the data and present significant findings which inform the inquiry of the original question.

To analyse the data in Dedoose relevant excerpts were taken from each interview and service evaluation. The excerpts were then assigned a code and weighted according to their significance within the context of the interview question or evaluation criterium.

157 excerpts were used and 40 codes to quantitise the data returned from the qualitative analysis exercises. Figure 4-1 is display of the project summary from Dedoose.

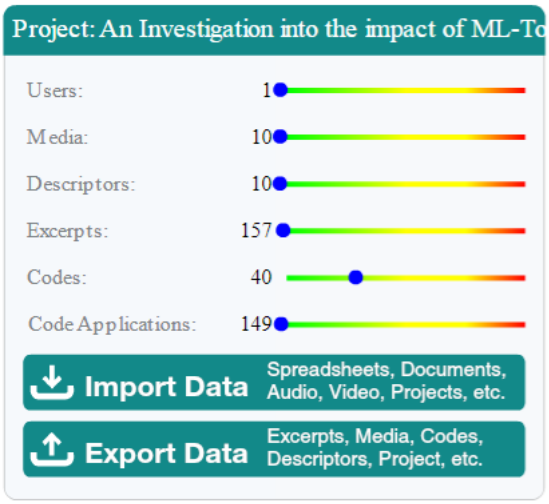

FIGURE 4-1 PROJECT SUMMARY, SOURCE: DEDOOSE (2019)

Interviews:

Four industry leaders in the field of software testing were interviewed. Each respondent had experience in creating and implementing a software testing strategy that incorporated software test automation. Question responses were analysed as outlined in the previous chapter. Interviews were recorded using Skype. Recordings were saved in mp4 format and transcribed into word documents, which were uploaded to Dedoose for analysis. Analysed data was then transferred to spreadsheets for further analysis and grouping.

Empirical Analysis:

Five commercial software testing services were investigated. The criteria for selection was that the tools had incorporated some form of machine learning into their solution and were identified as suitable samples through industry-related reports and evaluations, including:

- Reviews for Software Test Automation. Gartner (c.2019).
- 8 Innovative AI Test Automation Tools for the Future: The Third Wave Colantonio, J. (2017)
- AI Driven Testing (AI-DT) open source tools. Stack Exchange (2018)

The following is a description of each service selected for this investigation

Service	Description
Functionize	Functionize was founded in 2015 and describes their tools as: 'Legacy testing exposes your business to incalculable risk. Functionize's AI removes test churn, increasing the accuracy and efficiency of testers.' Functionize (c.2019)
TestSigma	TestSigma were founded in 2017. Their services is describe as 'AI-Driven Test Automation Ecosystem for Web, Mobile Web, Android & iOS Apps, and API automated testing. No coding skills required.' TestSigma (c.2019)
Appvance.AI	Appvance was founded in 2012, and describe the capability of their product as follows 'Appvance IQ delivers transformational productivity gains in both test creation and execution, the former through AI scripting and codeless test creation, the latter through unified functional, performance and security testing.' Appvance (c.2019)

TestIM	TestIM was founded in 2014. 'Testim uses artificial intelligence to speed-up the authoring, execution, and maintenance of automated tests.' TestIM.IO (c.2019)
Rainforest QA	Rainforest were an established service before introducing ML into their processes. 'We automate the QA process — not just tests — and include AI-based validation, on-demand testing, and automated test suite management, so you can align QA with the speed of your release goals and have confidence in every delivery.' Rainforest QA (c. 2019)

TABLE 4-1 MACHINE LEARNING TESTING SERVICES DESCRIPTION

For comparison purposes Selenium Webdriver was also assessed using the same criteria.

Service	Description
Selenium Web Driver	Selenium is an open source automation tools that can be used with several programming languages including Java. 'Selenium automates browsers. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) be automated as well.' seleniumhq (c.2015)

TABLE 4-2 SELENIUM WEB DRIVER DESCRIPTION

Tools were assessed using material available online, from their own websites usually which contained product descriptions, video tutorials, white papers and online demonstrations. This information was then saved into word documents, which were uploaded to Dedoose for analysis.

4.1.1 Findings: Machine Learning in Software Test Practice

The following section details the findings from collected data designed to answer **'**How is **Machine Learning (ML) being used to Impact Software Testing Techniques?'**

Interview question 1:

As discussed in Chapter 3 the rationale behind this question was to establish the respondents' perception of machine learning and if it was mature enough to begin making an impact on software testing. The second question was to assess the extent respondents had heard of machine learning being used in software testing via industry-related media, webinars, seminars, etc.

Response Groupings	Rsp.1	Rsp.2	Rsp.3	Rsp.4
1. Maturity of ML in Testing	2	1	2	1
2. ML/ST Media Exposure	1	2	1	2

TABLE 4-3 INTERVIEW QUESTION 1 RESPONSE FREQUENCY

The data set above represents how often the maturity of ML and the media exposure was discussed during the interviews. All four respondents discussed each topic at least once.

Response Groupings	Count	Min	Max	Mean	Median
1. Maturity of ML in Testing	6	1	3	2	2
2. ML/ST Media Exposure	6	1	3	1.8	2

TABLE 4-4 INTERVIEW QUESTION 1 RESPONSE WEIGHTINGS

The first row in Table 5-2 represents the how mature, or advanced each respondent thought machine learning was within the context of software testing. The minimum number is 1, representing that at least one respondent thought that machine learning had advanced to only a minimum level of impact on software testing. The mean number is 2, which suggests that across the respondents the opinion is that the maturity of machine learning is still extremely low, too low to be leveraged effectively in software testing.

As respondent 4 suggested: *"The impression I got is it that machine learning was more of an experiment than expecting real world value of it. I think there is potential there, but I think the testing domain is too diverse and unstructured to qualify for structured machine learning at this stage."*

The second row represents the extent to which the respondents have heard of machine learning in industry-related media. The mean figure is a low 1.8. Machine learning has been advertised and discussed seldom in software testing forums.

Empirical Analysis Category 1: Machine Learning Maturity Test

Response Groupings	Count	Min	Max	Mean	Median
Company Maturity	5	2	7	5	5

TABLE 4-5 EMPIRICAL ANALYSIS CATEGORY 1 WEIGHTINGS

The data in Table 5-3 represents the age of the companies which were assessed. The youngest was two years, with two being around since 2012. The mean was five years. When these figures are compared to the average age of small business published by jpmorganchase (2014), which found that '32 percent of small businesses are 5 years old or less' the mean age of the companies are within the age bracket of 32% of small start-ups. This would suggest that ML-Driven software testing services, while early in development, have been established as viable business models with companies such as Appvance having been around for 7 years. When compared to the respondent's data however, it can be argued that these companies are still largely unknown in the software testing industry.

Interview question 2

The next set of response groupings aim to uncover if the respondents have trialled any ML tools. Following on from that the aim was to understand what risks and benefits would be included in any assessment before adopting an ML-Driven tool into their test strategies.

Response Groupings	Rsp.1	Rsp.2	Rsp.3	Rsp.4
1. ML Adoption Rate by Respondents	1	1	1	1
2. Concern of Cost	0	0	1	2

3. Reliability of ML Services	0	1	2	2
4. Security Concerns	0	0	2	0
5. Ease of Integration	1	0	2	0
6. Cloud Integration	1	3	0	0
7. Ease of Use	3	4	3	2

TABLE 4-6 INTERVIEW QUESTION 2 RESPONSE FREQUENCY

Information to note from Table 5-4, which represents how often each topic was discussed by each respondent is that cost was only a concern for 2 of the respondents, and security was only mentioned by one respondent. Ease of use was discussed at great detail, which is understandable given the difficulty of using standard automation tools and the potential for ML-Driven tools to address this impediment.

Response Groupings	**Count**	**Min**	**Max**	**Mean**	**Median**
1. ML Adoption Rate by Respondents	1	1	1	1	1
2. Concern of Cost	3	6	7	6.3	6
3. Reliability of ML Services	5	6	8	7.2	7
4. Security Concerns	2	8	8	8	8
5. Ease of Integration	3	7	7	7	7
6. Cloud Integration	4	6	8	7	7
7. Ease of Use	12	5	8	6.7	7

TABLE 4-7 INTERVIEW QUESTION 2 RESPONSE WEIGHTINGS

The data represented in Table 5-5 shows that the actual adoption rate was non-existent, whereby only one respondent even considered ML tooling. Respondent 2 mentions: *"And now we're seeing the benefits or the cost of good testing coverage. So, I think we're now starting to look at tooling, such as machine learning, that would help us in that."* No other respondents had even considered ML-Driven tooling as an option.

Where cost was mentioned as a concern it was considered highly on the list of factors to consider. Respondent 3 drew the comparison of spending on machine learning with spending on more human testers: *"Am I getting an extra 2 testers worth of value out of machine learning? If I hire two new testers, will they be able to do the work that I'm paying for this subscription service and the AI?"*

Reliability of machine-learning services was discussed and it factored highly for respondents two, three and four. The main concern was that if a machine learning tool was automatically generating, and self-healing test cases then how could it be confirmed these test cases were adding value without the requirement for testers to verify the accuracy of the tests. This speaks to the general scepticism of the respondents in the ability of machine learning to replace human testers as the creative source of test cases for front-end applications.

The increased security threat was highlighted by Respondent 3, who considered uploading test cases to a third-party system as unacceptable, stating that *"Security is a huge thing. I just wouldn't upload test cases to a Third-Party company. I just wouldn't do it."*

Infrastructure integration was discussed, and two candidates were particularly curious as to the ability of machine learning to integrate with cloud environments. A mean score of 7 indicated that a high importance was given to the adaptability of the services to integrate with established CI and other support systems.

The potential for testers to easily create and analyse automated tests was of great interest to all respondents. Rapid test creation and execution is where the respondents could see a lot of the value residing in machine learning. Respondent 2 stated that *"if the language to define the test script is far more natural language, then it's going to make those business users far more productive"*

Empirical Analysis Category 2: Machine Learning Adoption Considerations:

Criteria	Count	Min	Max	Mean	Median
(Reliability) Quality of Generated Test Cases	5	0	4	1.4	0
Cost of Tool	5	5	8	7	7
Ease for Testers to Learn Tool	5	4	8	6.2	6
Ease of Integration Set-Up	5	3	7	5	5
Ease of setting up Cloud Environment Integration	5	3	3	3	3
Quality of Analytical Tools	5	5	8	5.8	5
Quality of Support and Training Material	5	3	7	5.6	7
Security Threat from Tool	5	3	8	4.6	4

TABLE 4-8 EMPIRICAL ANALYSIS CATEGORY 2 WEIGHTINGS

The data in Table 5-6 represents the weighted values for each of the criteria that were designed to align with the risks and benefits of adopting ML-Driven tools discussed during the interviews.

The first item '(Reliability) Quality of Generated Test Cases' was attributed a low score as only two of the tools offered the ability to generate test cases automatically. Test cases that were generated were difficult to read, which means deciphering results would involve a high level of effort and increase the need for human intervention.

Inevitability costs were higher than using an open source tools such as Selenium. $10,000 per month was suggested by one service for 30 testers using the tool, which represents a significant increase to spending on test services. However, this cost would be offset by the reduction in costs of running an inhouse Selenium grid solution and by the time saved from the testers being able to write scriptless automated tests (tests that don't require any coding).

All of the tools provided some type of scriptless automation, but two tools actively encouraged testers to use code to increase the quality of scripts. TestIM claimed that *'Using code in your tests is the best way to make sure they are reusable and easy to maintain.'* This would represent a return to the cost of scripting automated scripts incurred by standard tools. As this is one of the highest concerns expressed by the respondents this would discourage the adoption of machine learning tools.

Integration of the tools was straightforward for on-premise testing. Most offered solutions of integrating with the cloud system of the SaaS for creation and execution or provided a solution to run the system within the organisations own internal network, or a hybrid solution. The tools that scored highest in this category were capable of integrating with the most third-party services. Functionize offers integration with: *AWS CodePipeline, Bamboo CI, Circle CI, GitHub, Heroku, Jenkins, Jira, Pagerduty, Spinnaker.io, Travis CI, PagerDuty, ForBugz, Sentry.io, GitLab, Slack, Hipchat, Jira.* However, none of the services explicitly stated that they could easily integrate with a test system hosted in a Cloud. This suggests that this may be more complex, and as such the 'Ease of setting up Cloud Environment Integration' was weighted as low.

All services offered greater analytical capability and reports than the standard Selenium tool. Many of them offered monitoring tools which offered visibility of results as they were running in real-time while reports offered many levels to drill-down to identify where and when failures occurred, providing easier root cause analysis. Functionize states that it *'will visually learn your application-layout and identify any test failures or anomalies leveraging our Adaptive Event Analysis™ (AEA) Engine.'*

'Quality of Support and Training Material' varied, with one service providing only a selection of blogs to walkthrough their services. Others however had a comprehensive list of whitepapers, videos and 24-hour support made available to clients.

The security threat of for most of these tools would be similar to any service that was allowed access to an organisation's infrastructure. If the service itself was compromised then there is a comparable threat to the organisation, depending on the oversight that the organisation put in place. All services require the organisation to whitelist their systems in order to work. One interesting declaration was from Rainforest, which stated that *'All Rainforest testers are required to sign a Non-Disclosure Agreement with Rainforest in order to test with us.'* Whether or not this would make Respondent 3 more or less inclined to grant access to a Third-Party company is debatable.

4.1.2 Findings: Machine Learning & Commercial Availability

The following section details the findings from collected data designed to answer **'How are Commercial ML-Driven Automated Software Testing Tools different from Traditional non-ML Automated Tools?'**

Interview question 3:

The purpose of this question was to uncover what specific capabilities the respondents would like to see from machine learning, and the value of each one. The machine learning services were then assessed to review how close each was to providing these capabilities to expectations of the respondents.

Response Groupings	Rsp.1	Rsp.2	Rsp.3	Rsp.4
1. Automatic Test Case Generation	2	3	1	1
2. Reduced Maintenance Effort	2	1	0	1
3. Quick Feedback/Execution Times	0	0	2	0
4. Automatic UI Testing	0	0	1	0

TABLE 4-9 INTERVIEW QUESTION 3 RESPONSE FREQUENCY

Data in table 5-7 finds that all candidates were interested in the capability of machine learning to automatically generate test cases, without the need for human intervention. A reduction in the cost of maintaining scripts was also highly sought. Rapid feedback and UI testing was only discussed during Respondent 3's interview.

Response Groupings	Count	Min	Max	Mean	Median
1. Automatic Test Case Generation	7	3	10	7.1	8
2. Reduced Maintenance Effort	4	7	8	7.5	7.5
3. Quick Feedback/Execution Times	2	7	7	7	7
4. Automatic UI Testing	1	2	2	2	2

TABLE 4-10 INTERVIEW QUESTION 3 RESPONSE WEIGHTINGS

From Table 5-8 the mean values of automatic test case generation and reduced maintenance were of highest value. Respondent 4 gave his opinion on test generation as "*From a positive perspective – essentially, if you're fast-tracking a test case creation it is hugely valuable. And that goes without a shadow of a doubt".* All respondents saw the most value in these capabilities while rapid feedback was of high value to one respondent. UI testing was not rated highly as a desirable feature of machine learning.

Empirical Analysis Category 3: Machine Learning Features

Criteria	Count	Min	Max	Mean	Median
Ability to Automatically Generate Test Cases	5	0	8	3.2	0
Self-Healing of Broken Tests	5	3	7	4.8	5
CI Execution Capabilities and Feedback	5	8	9	8.4	8
Automatic UI Testing Capability	5	1	7	2.4	1

TABLE 4-11 EMPIRICAL ANALYSIS CATEGORY 3 WEIGHTINGS

This quality of each capability was assessed for each machine learning tool. When this data is correlated with the value expressed by the respondents a useful mechanism is provided which the current impact of machine learning can be evaluated.

The capability for automatic test case generating was only available in two of the services. All offer an easier method of creating automated tests. Where test cases are generated automatically human intervention is still required to review results and ensure that test coverage is adequate. Automatically generated test cases also learn from test cases that have already been uploaded to the system. The mean was assessed to be low of 3.2 for this category.

Self-healing test capability is offered by all services, however the level of human intervention required depends on the level of change of the underlying SUT. Where an SUT screen is altered machine learning can identify what changed and usually easily update the test case, through the use of multiple methods of locating an element, rather than relying on one method of locating an element, which is the case in Selenium. However, if functional paths through the system change significantly machine learning can provide information and offer suggestions but ultimately human intervention is still required to verify that changes to automated test cases are valid. From the interviews this capability was valued highly with a mean of 7.5 but the capability was allocated a relatively low mean of 4.8.

All services offered a high rate of execution, with many offering the capability of executing test cases on solutions that could simulate test runs on a range of client operating systems, browser types and versions. This execution capability provides a rapid feedback to clients but also requires that the client's test systems are available and capable of handling the increased levels of traffic that these test execution runs would generate. This is the highest rated capability with a mean of 8.4.

Automatic UI testing was only explicitly mentioned by only one service so the mean is low for this capability at only 2.4. The mean for UI testing as a desirable capability for respondents was only 2 so, this would not necessarily be an impediment to the adoption of ML.

4.1.3 Findings: Human Impact & Five-Year Forecast

The following section details the findings from collected data designed to answer **'How will the Rate of Adoption impact on Professional Software Testing from both a Human and Business Perspective within Five Years?'**

Interview question 4:

Question 4 was subjective; in that it required a certain amount of speculation as to what impact machine learning would have on the professional tester role.

Response Groupings	Rsp.1	Rsp.2	Rsp.3	Rsp.4
1. Probably of Reduction of Tester Roles	1	4	3	2
2. Career Advancement	3	0	2	1
3. Improved Creative Freedom	1	1	0	0

TABLE 4-12 INTERVIEW QUESTION 4 RESPONSE FREQUENCY

The potential reduction of the need for software testers was discussed with all respondents. Three of the respondents spoke of the potential of machine learning to improve the software testing role by introducing new skills and areas of interest. The potential for testers to be freed up for more creative and rewarding work was discussed with two of the respondents, where the potential reduction of low-level mundane tasks was discussed by all.

Response Groupings	Count	Min	Max	Mean	Median
1. Probably of Reduction of Tester Roles	10	2	8	3.2	3
2. Career Advancement	6	5	8	6.8	7.5
3. Improved Creative Freedom	2	7	7	7	7

TABLE 4-13 INTERVIEW QUESTION 4 RESPONSE WEIGHTINGS

The reduction of the tester role was role was discussed 10 times during the interview process and only one respondent, Respondent 4, felt that there would be a reduction in the number of software roles brought about by the advent of machine learning. The mean number was assessed at 3.2, providing valuable insight into how the respondents thought software testing roles would be impacted by machine learning. As Respondent 1 suggested: *"I don't see it leading to a reduction in the number of software testers because at the end of the day, there are certain things that still have to be manually done. I'm making the assumption that even with the AI tools, you have certain kind of test scenarios where we just test manually."*

Rather than test roles being reduced or diminished respondents thought that machine learning will bring new opportunities to software testing. Respondent 1 goes on to say *"On the plus side, which for me far outweighs the potential fear factor, is the fact that it's new technologies. We'd be learning and building up new knowledge around these new technologies, understanding a different way how AI can build out and automate tests."* This generally seemed to be the perception as the mean for this grouping was set as 6.8.

For those respondents that did discuss the reduction of mundane tasks and the increased time spent on more creative testing tasks, such as test design and analysis, the optimism was high and Improved Creative Freedom was scored highly as likelihood at 7.

Interview question 5

The goal of this question was to gain insight from the respondents, as leaders in software testing as to how machine learning will grow in the field over the next five years.

Response Groupings	Rsp.1	Rsp.2	Rsp.3	Rsp.4
1. Probability of Widespread Adoption in 5 Years	2	1	3	3

TABLE 4-14 INTERVIEW QUESTION 5 RESPONSE FREQUENCY

The topic was discussed with all respondents, and came up multiple times during three interviews.

Response Groupings	Count	Min	Max	Mean	Median
1. Probability of Widespread Adoption in 5 Years	9	2	6	3.3	3

TABLE 4-15 INTERVIEW QUESTION 5 RESPONSE WEIGHTINGS

A low score here represented the opinion that machine learning would not have a high adoption rate across the industry over the next five years, whereas a high score suggested that ML-Driven tools would move significantly towards being an industry-standard. Only Respondent 2 predicted a significant change, saying it *"will absolutely gain traction because AI offers such promise across the entire IT organisation, you've got to imagine that where there is an element of testing that is regression testing or evolution testing, that it's got to be a sweet spot for AI."*

The other three candidate were less sure that machine learning will significantly change the software testing landscape within a timeframe of five years and the mean came out as a low 3.3.

4.2 Conclusion

The analytical capabilities offered by Dedoose provided instrumental in correlating data from the interviews and ML-Driven testing services. Data collected from the interviews found that machine learning is not yet considered as alternative to manual and standard automation approaches in the software testing industry. When compared to the capabilities of the ML-Driven services, the areas in which respondents expressed value varied to the extent that that value could be achieved. These comparisons provided a framework from which conclusions could be inferred, which are discussed in Chapter 5.

4.3 Limitations

The comparison used for the machine learning tools was based on the standard automation tool used by the researcher's employer, Selenium Web Driver. This business model is a business-to-business web-based solution. This meant that the research was limited to assessing machine learning as an impact to this business model predominantly.

A further limitation which should be highlighted is that in order to quantitise the qualitative data weightings were allocated to data collected which are prone to subjective bias.

5. Chapter 5: Conclusions and Future Work

5.1 Introduction

This chapter will discuss findings from the previous chapters and draw conclusions from the information gathered during the literature review and research. The primary question is **How can Machine Learning be Leveraged for Software Testing within a Timeframe of Five Years?** This question is discussed within the context of the following main themes:

- The value that machine learning offers software testing in terms of improved capability
- The commercial availability and adoption rates of machine learning services in the software testing industry
- Disadvantages to machine learning services which would impede adoption
- Positive and negative implications for humans working in software testing
- The five-year forecast for machine learning in software testing

5.2 Answering the Research Question

5.2.1 Improved Capabilities

The integration of machine learning into software testing has produced promising results as is evidenced by experiment results such as the one conducted by Rosenfeld, A. et al which found that machine learning could *'automatically cover a large portion of the human testers' work suggesting a significant potential relief in the manual testing efforts.'* Machine learning can be used to learn and improve to provide better fault detection systems, as reported by Bowes, D. et al which found *'that mutation-aware fault prediction can outperform traditional fault prediction, for a range of predictive modelling machine learning algorithms.'* A range of experiments described more innovative applications in the areas of improved branch coverage, model-based testing and stabilising pre-existing tests.

Machine learning has been adapted and commercialised with services offering improved capabilities in four general categories:

Improved test case creation is facilitated by machine learning through 'scriptless' automation, which removes the requirement to code automated tests, allowing for faster and cheaper test creation.

Services also provide the ability to automatically generate new test cases, however this capability was limited, and was not offered by all of the services included in the study. As respondents considered this high on their priorities for machine leaning and services scoring low for actually delivering on this capability the implication is that the services are still limited in their ability to meet customer expectations in this area.

The services make it easier for testers to **maintain automated scripts**, an area identified by Throvagunta, S. et al (2018) during their review of the 'Main challenges in achieving desired level of test automation' where 61% of respondents found 'it difficult to automate because our applications change too much with every release'. This was echoed by respondents from this research where reduced maintenance effort scored highly on their list of concerns. Services offer 'Self-Healing' tests. This capability was limited however, and only straightforward changes can be made directly, with human intervention required for more complex functional changes brought about changes to the SUT. A mean of 4.8 out of 10 was calculated for this capability which again is relatively low compared to the 7.5 priority score that respondents provided. This represents another limitation of machine learning that would discourage potential clients.

Test cases can be executed through the services' systems. All the services included in the research offered the ability to execute large sets of test cases using their systems to host execution jobs. This provides the ability to execute test cases in parallel, on an array of different client operating systems and browser types, providing increased coverage and eliminating the requirement for clients to maintain their own internal grid structures. The advantage is **quicker feedback and improved analytical ability** as the services also offer improved root cause analysis and defect reporting through machine learning application. This scored highly on respondents' priorities and the mean was calculated highly on the empirical evaluation of the services. Currently, this is one of the stronger selling points of machine learning services.

Automated UI tested has also been reported as an advantage of machine learning. 'By emulating the human eye and brain, our algorithms only report differences that are perceptible to your users' according to Applitools (c.2018). However, for the services included in the research the mean for Automatic UI testing was only 2.4. The mean for UI

testing as a desirable capability for respondents was only 2 so, this would not necessarily be an impediment to the adoption of ML.

5.2.2 Commercial Availability and Adoption

The availability of ML-Driven services is a factor which indicates the extent to which machine learning will impact software testing practices. Awareness of ML-Driven tools was very low amongst respondents which implied that machine learning currently has a very low impact on software testing currently. It was found that respondents were not exposed to any machine learning related material in any of the industry forums they were involved in and they had no exposure to services via advertising or sales.

A review of the companies providing the services found that they were relatively young, with the mean being just five years. While the companies were relatively young their ability to remain in business for a period of five years suggests that machine learning in software is a viable business model, and will grow further as awareness grows.

5.2.3 Potential Disadvantages and

The concerns of software testing leadership about adopting ML-Driven services also provides informative metrics which can be applied to the question under investigation. Concerns discussed and raised by respondents included

- Concern of Cost
- Reliability of ML Services
- Security Concerns
- Ease of Integration
- Cloud Integration
- Ease of Use

While standard automation tools like Selenium are open source, machine learning currently is available only as a subscription service. This featured highly on respondents' priorities with a mean of 6.3. The costs of the services were rated at a high 7. When considering the cost however, the savings implied by the services should be taken into account, such as reduced time on creating test cases, providing faster feedback and reducing manual test effort. Costs can be offset also by reduced maintenance of scripts, and a reduction of costs on maintaining an internal grid of client environments to execute

the test scripts. Organisations considering adopting one of the services would need to perform a full cost-benefit analysis before making a decision. While one service was costed at $10,000 per month for 30 users, this is well below the average salary for one tester. If test creation could at least partially be covered by machine learning then arguably the headcount of testers could be reduced to cover this cost. If savings can outweigh the costs then obviously adoption rates would be positively impacted.

Reliability of automatically generated test cases was another concern. How can the test cases that machine learning algorithms generate be validated and trusted to effectively test a system? Only two service providers offered this capability, and those services that did produced large volumes of test cases that still needed to be analysed manually. The suggested progression of automated testing as illustrated in Figure 2-4 suggests that the ability for machine learning to automatically create test cases is still an aspiration rather than a reality. When ML-Driven tools can reliably generate test cases with minimal human intervention then software test leaders will start taking a genuine interest.

There are increased security concerns as ML-Driven tools will require access to internal test environments and intellectual property in the form of test cases and functional design. Organisations which are more security conscious might baulk at introducing a new level of vulnerability where one did not exist beforehand. As service providers grow in maturity and reputation, they will need to establish trust to allay these fears. Organisations will need to perform a comprehensive risk-analysis before integrating ML-Driven services into their operations.

ML-Driven tools are integrated into organisations continuous integration systems to execute tests regularly. Organisations might also require integration with other support systems, like defect tracking tools. This scored a high rate of 7 for respondents and scored a 5 for ease of integration. Organisations might find they need to modify internal structures to accommodate integration, which might incur extra cost which would negatively impact on adoption rates of machine learning services.

A significant drawback of standard automation tools is the skill level involved in successfully implementing them. This is highlighted in Figure 2-5 where 48% respondents cited lack of skills as a challenge when adopting automation. Respondents interviewed in this research were concerned about the usability of machine learning tools, with a score of 6.7 calculated. ML-Driven services scored highly at 6.2 for ease of use. ML-Driven tools make it easier to create automated tests, with the necessity for coding varying across

tools. The ability to automate without coding is a strong selling point as it reduces the delay in producing automated scripts, which provides faster feedback and reduces tester effort and training requirements. Evidence suggests that this could be a strong selling point, particularly for organisations that have not adopted automation at all and would prefer not to train testers how to code.

5.2.4 Human Impact

Investigating the impact of machine learning on the software test profession entailed assessing the probability of a reduction of software testing roles, while also researching possible changes to the nature of work required of the role.

Respondents mostly returned a low score for this question on the likelihood of a reduction in tester roles, with the mean calculated at 3.2. The overriding impression was that software test roles would not be reduced. The respondents argued that software testers would still need to be involved in design of test cases and interpretation of results. From studying the test generation and creation features of ML-Driven tools this argument is verified to a certain extent as testers are still required to create test cases.

Positive implications for the test roles were that there would be added job satisfaction due to the introduction of new skills related to machine learning, such as becoming familiar with ML-Driven tools, Business Intelligence, Data Analysis skills, etc. Respondents returned a 6.8 score for the likelihood of a positive impact on the software tester role through upskilling in ML-related practices. A mean of 7 was calculated as the probability that there would be improved levels of creative freedom.

However, the introduction of more complex tasks to the test role could be a poison chalice as Van De Ven, T., et al suggest that the *'Traditional tester is no longer adequate, as working with AI requires professionals with a diverse range of competencies such as testing, mathematical optimization, neuro-linguistic programming, AI, business intelligence skills and algorithmic knowledge'* . The potential change to the skill set required by software testing roles might be so radical that current testers will find they are lacking in the ability to adapt. Results from this investigation however, suggest that radical changes such as these will not happen soon, and will probably not take place within five years.

5.2.5 Five Year Forecast

Respondents were sceptical that ML-Driven tools would experience wide-spread adoption within a timeframe of five years. The mean score for this grouping was calculated at a low 3.3 out of 10. Only one candidate thought that adoption rates would occur significantly. Respondent 1 suggested that '*There'll always be some companies who like to be leading edge – it's their modus operandi to always be leading edge, and be the first to come up with a new set of technology and champion it. But I think it'll take time. Companies adopt change or embrace change at different speeds and paces. I don't see any time in the near future where non-AI software tools will be redundant.'* For ML-Driven tools to become an industry standard there will need to be more case studies which showcase their benefits. It has been established that the respondents have not been made aware of any the services from software testing forums and industry media and none have any plans to introduce machine learning into their practices in the near future. The information collected from this investigation suggests that ML-Driven Services are still in a very early stage of development. While it is difficult to predict the future accurately it would seem that widespread adoption will not occur within five years.

5.3 Conclusion

There are many beneficial applications available from machine learning which can be adopted by software testing practices. These benefits can be costly however, and those that are most sought by leaders are not yet fully realised. Fully automated test case generation and test case maintenance are still not available and human intervention is still required. However, the ML-Driven services should not be dismissed as they do offer value. This value, arguably, would be most apparent to organisations who have yet to successfully integrate a standard automation strategy. If software testers are provided the ability to produce scriptless automated scripts using a tool that also enables them to quickly identify where and how an automated test is broken by changes to the SUT then the adoption rate of ML-Driven tools could find traction quickly. As pressure to deliver stable software grows, the imperative for rapid feedback on the quality of the software will grow proportionally. It is perhaps this imperative that will drive the adoption rates of ML-Driven software test tools. However, given the current very low adoption rate of ML-Driven services and lack of awareness of machine learning capabilities the rate of adoption does not look set to increase significantly within a five-year time frame.

5.4 Future Work

To extend on the research conducted during this investigation a quantitative research method could be used to fully assess the cost benefit analysis of using a Machine Learning tool. This would entail designing a set of test cases beforehand. The same set of test cases could then be automated using a standard automation tool like Selenium and an ML-Driven automated tool.

Information could be gathered by:

- Comparing the time taken to automate the script using each tool
- Comparing the time taken to fix broken tests after a major upgrade
- Comparing the cost of executing the scripts on an inhouse Selenium grid compared to executing the test cases on a hosted cloud service
- Assessing the security risk of using the ML-Driven service and the cost of mitigating the risk should the tool be adopted
- Assessing the time saved on UI Testing, assuming it was available

There would be other implied costs associated with the change process in adopting an ML-Driven tool which would need to be factored in.

Machine learning is coming to software testing, probably not in the next five years, but it looks certain to make an impact at some stage over the coming decades. As the CEO of Appvance, Surace, K. (2018) advices 'While small companies and start-ups may try anything new, even a new recorder labelled "AI", high quality enterprises don't change easily or quickly and are not easily fooled with marketing hype. They need to see real impact across hundreds of applications. And true enterprise level support to roll out AI generated tests over time, with care.'

6. References

2014. A large share of small businesses are young businesses [Online]. Available: https://www.jpmorganchase.com/corporate/institute/small-business-longevity.htm [Accessed 28 April 2019].

2017. Sizing the prize. What's the real value of AI for your business and how can you capitalise?: Price Waterhouse Cooper.

2018. AI Driven Testing (AI-DT) open source tools. Stack Exchange [Online]. Available from: https://sqa.stackexchange.com/questions/31336/ai-driven-testing-ai-dt-open-source-tools/31423 [Accessed 28 April 2019].

2018. AI predictions. 8 insights to shape business strategy. Price Water House Cooper.

Applitools [Online]. Available: https://applitools.com/ [Accessed 28 April 2019].

Appvance IQ [Online]. Available: https://www.appvance.ai/ [Accessed 28 April 2019].

BOWES, D., HALL, T., HARMAN, M., JIA, Y., SARRO, F. & WU, F. 2016. Mutation-Aware Fault Prediction. University College London.

COLANTONIO, J. 2017. 8 Innovative AI Test Automation Tools for the Future: The Third Wave [Online]. [Accessed 29/04/2019 2019].

Dedoose [Online]. Available: https://www.dedoose.com/ [Accessed 28 April 2019].

DUDOVSKIY, J. Inductive Approach (Inductive Reasoning) [Online]. Available: https://research-methodology.net/research-methodology/research-approach/inductive-approach-2/ [Accessed 28 April 2019].

FANPING, Z., LING, L., JUAN, L. & XUFA, W. 2009. Vulnerability Testing of Software Using Extended EAI Model. World Congress on Software Engineering.

Functionize [Online]. Available: https://www.functionize.com/ [Accessed 28 April 2019].

GHAHRAI, A. 2018. Test Automation Advantages and Disadvantages [Online]. Available: https://www.testingexcellence.com/test-automation-advantages-and-disadvantages/ [Accessed 28 April 2019].

GRANO, G., TITOV, T. V., PANICHELLA, S. & GALL, H. C. 2018. How High Will It Be? Using Machine Learning Models to Predict Branch Coverage in Automated Testing. Department of Informatics, Switzerland.

GROZ, R., SIMAO, A., BREMOND, N. & ORIAT, C. 2018. Revisiting AI and Testing Methods to Infer FSM Models of Black-Box Systems. 2018 ACM/IEEE 13th International Workshop on Automation of Software Test.

GÜLDALI, B., MLYNARSKI, M. & SANCA, Y. 2010. Effort Comparison for Model-based Testing Scenarios. Third International Conference on Software Testing, Verification, and Validation Workshops.

How to... conduct empirical research [Online]. Emerald Publishing. Available: http://www.emeraldgrouppublishing.com/research/guides/methods/empirical.htm?part=6 [Accessed 28 April 2019].

ISTQB Glossary [Online]. Available: https://glossary.istqb.org/search/test [Accessed 28 April 2019].

KHOSROWJERDI, H., MEINKE, K. & RASMUSSON, A. 2018. Virtualized-Fault Injection Testing: a Machine Learning Approach. 2018 IEEE 11th International Conference on Software Testing, Verification and Validation.

KING, T. M., SANTIAGO, D., PHILLIPS, J. & CLARKE, P. J. 2018. Towards a Bayesian Network Model for Predicting Flaky Automated Tests. 2018 IEEE International Conference on Software Quality, Reliability and Security Companion.

LACHMANN, R., NIEKE, M., SEIDL, C., SCHAEFER, I. & SCHULZE, S. 2016. System-Level Test Case Prioritization Using Machine Learning. 2016 15th IEEE International Conference on Machine Learning and Applications.

MARR, B. 2016. What Is the Difference Between Artificial Intelligence And Machine Learning? [Online]. Available: https://www.forbes.com/sites/bernardmarr/2016/12/06/what-

is-the-difference-between-artificial-intelligence-and-machine-learning/#1e2a6a602742 [Accessed 28 April 2019].

NARKHEDE, P. 2017. INTRODUCTION TO SELENIUM GRID – WHAT IS SELENIUM GRID? [Online]. Available: https://automationtalks.com/2017/07/22/introduction-to-selenium-grid-what-is/ [Accessed 28 April 2019].

PLATZ, W. 2017. What's beyond continuous testing? AI [Online]. Available: https://sdtimes.com/ai/whats-beyond-continuous-testing-ai/ [Accessed April 28 2019].

RADIGAN, D. Continuous integration, explained [Online]. Available: https://www.atlassian.com/continuous-delivery/continuous-integration [Accessed 28 April 2019].

Rainforest QA [Online]. Available: https://www.rainforestqa.com/ [Accessed 28 April 2019].

Reviews for Software Test Automation [Online]. Gartner. Available: https://www.gartner.com/reviews/market/software-test-automation [Accessed 28 April 2019].

ROSENFELD, A., KARDASHOV, O. & ZANG, O. 2018. Automation of Android Applications Functional Testing Using Machine Learning Activities Classification. 2018 ACM/IEEE 5th International Conference on Mobile Software Engineering and Systems.

SAUNDERS, M., LEWIS, P. & THORNHILL, A. 2009. Research methods for business students, Harlow, Financial Times Prentice Hall.

Selenium HQ [Online]. Available: https://www.seleniumhq.org/ [Accessed 28 April 2019].

SHARMA, L. 2017. Difference between Verification and Validation [Online]. Available: https://www.toolsqa.com/software-testing/difference-between-verification-and-validation/ [Accessed 28 April 2019].

SHARMA, P. 2017. Advantage and Disadvantage of Automation Software Testing [Online]. Available: http://paasis.com/advantage-disadvantage-automation-software-testing/[Accessed 28 April 2019].

STRICKLAND, A. 2019. Software Evaluation Checklist [Online]. Available: https://checklist.com/software-evaluation-checklist [Accessed 28 April 2019].

SUBRAMANIAN, R. 2018. How AI is transforming Software Testing [Online]. Available: https://confengine.com/selenium-conf-2018/proposal/5964/how-ai-is-transforming-software-testing [Accessed 28 April 2019].

SURACE, K. 2018. The AI Testing Hype Machine [Online]. Available: https://www.appvance.ai/ai-testing-hype-machine [Accessed 28 April 2019].

TestIM [Online]. Available: https://www.testim.io/ [Accessed 28 April 2019].

TestSigma [Online]. Available: https://testsigma.com/ [Accessed 28 April 2019].

THROVAGUNTA, S., OLSEN, B. & AYMER, A. 2018. World Quality Report 2018.

TORNHILL, A. 2018. Assessing Technical Debt in Automated Tests with Codescene. 2018 IEEE International Conference on Software Quality, Verification and Validation Workshops.

VAN DE VEN, T., DUPJAN, G. & MAMNANI, D. World Quality Report 2018.

WILSON, J. 2010. Essentials of business research: a guide to doing your research project, London, SAGE.

7. Bibliography

BERRAL, J. L., POGGI, N., CARRERA, D., CALL, A., REINAUER, R. & GREEN, D. 2017. ALOJA: A Framework for Benchmarking and Predictive Analytics in Hadoop Deployments. *EMERGING TOPICS IN COMPUTING*.

BUDNIK, C., GARIO, M., MARKOV, G. & WANG, Z. 2018. Guided Test Case Generation Through AI Enabled Output Space Exploration. *2018 ACM/IEEE 13th International Workshop on Automation of Software Test*.

COLANTONIO , J. 2018. *How AI is changing test automation: 5 examples* [Online]. Available: https://techbeacon.com/app-dev-testing/how-ai-changing-test-automation-5-examples [Accessed 28/04/2019 2019].

GEBIZLI, C. S., METIN, D. & SOZER, H. 2015. Combining Model-Based and Risk-Based Testing for Effective Test Case Generation. *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*.

GUPTA, R. 2018. *Can Artificial Intelligence replace human in Software Testing?* [Online]. Available: https://www.webomates.com/blog/artificial-intelligence-and-software-testing/ [Accessed 28 April 2019].

LABS, M. T. *WHAT ARE THE ADVANTAGES OF ARTIFICIAL INTELLIGENCE IN TESTING?* [Online]. Available: https://www.marutitech.com/artificial-intelligence-in-testing/[Accessed 28 April 2019].

LE, J. *The 10 Algorithms Machine Learning Engineers Need to Know* [Online]. Available: https://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html [Accessed 28 April 2019].

MAKADIA, M. 2018. *T op 4 Advantages of Integrating AI in Software Testing* [Online]. Available: https://www.business2community.com/business-innovation/top-4-advantages-of-integrating-ai-in-software-testing-02143577 [Accessed 28 April 2019].

PLASTINO, E. 2018. Game changing value from Artificial Intelligence: eight strategies. *Strategy & Leadership. 2018,* Vol. 46 16-22.

PURAM, A. D. 2018. *Your Guide to Leading Software Testing Tool Ranking and Review Sites* [Online]. Available: https://smartbear.com/blog/test-and-monitor/top-software-testing-tool-reviews/ [Accessed 28 April 2019].

SARRO, F. 2018. Predictive Analytics for Software Testing. *2018 ACM/IEEE 11th International Workshop on Search-Based Software Testing.*

STRAATEN, A. 2018. *Functionize Root Cause Analysis & Self-Heal* [Online]. Available: https://www.functionize.com/blog/functionize-root-cause-analysis-self-heal/ [Accessed 28 April 2019].

SYPOLT, G. 2017. *AI Test Automation: The AI Test Bots Are Coming* [Online]. Available: https://saucelabs.com/blog/ai-test-automation-the-ai-test-bots-are-coming [Accessed 28/04/2019 2019].

XINGHAN, Z. & XIANGFEI, G. 2018. An AI Software Test Method Based on Scene Deductive Approach. *2018 IEEE International Conference on Software Quality, Reliability and Security Companion*.

8. Appendix: Ethics Proposal

School of Computer Science & Statistics Research Ethics Application

Part A

Project Title: An Investigation into the impact of AI-Tools on the Software Testing Profession

Name of Lead Researcher: Karl Morrissey

Name of Supervisor: Paula Roberts

TCD E-mail: morrisk4@tcd.ie Contact Tel No.: 0861574603

Course Name and Code (if applicable): Management of Information Systems

Estimated start date of survey/research: 25/03/2019

I confirm that I will (where relevant):

- Familiarize myself with the General Data Protection Regulation Act and the College Good Research Practice guidelines http://www.tcd.ie/info_compliance/dp/legislation.php;
- Tell participants that any recordings, e.g. audio/video/photographs, will not be identifiable unless prior written permission has been given. I will obtain permission for specific reuse (in papers, talks, etc.)
- Provide participants with an information sheet (or web-page for web-based experiments) that describes the main procedures (a copy of the information sheet must be included with this application)
- Obtain informed consent for participation (a copy of the informed consent form must be included with this application)
- Should the research be observational, ask participants for their consent to be observed
- Tell participants that their participation is voluntary
- Tell participants that they may withdraw at any time and for any reason without penalty
- Give participants the option of omitting questions they do not wish to answer if a questionnaire is used
- Tell participants that their data will be treated with full confidentiality and that, if published, it will not be identified as theirs
- On request, debrief participants at the end of their participation (i.e. give them a brief explanation of the study)
- Verify that participants are 18 years or older and competent to supply consent.
- If the study involves participants viewing video displays then I will verify that they understand that if they or anyone in their family has a history of epilepsy then the participant is proceeding at their own risk
- Declare any potential conflict of interest to participants.
- Inform participants that in the extremely unlikely event that illicit activity is reported to me during the study I will be obliged to report it to appropriate authorities.

- Act in accordance with the information provided (i.e. if I tell participants I will not do something, then I will not do it).

Signed:  Date: 21/03/2019

Lead Researcher/student in case of project work

Part B

Please answer the following questions.		***Yes/No***
Has this research application or any application of a similar nature connected to this research project been refused ethical approval by another review committee of the College (or at the institutions of any collaborators)?		No
Will your project involve photographing participants or electronic audio or video recordings?		Yes
Will your project deliberately involve misleading participants in any way?		No
Does this study contain commercially sensitive material?		No
Is there a risk of participants experiencing either physical or psychological distress or discomfort? If yes, give details on a separate sheet and state what you will tell them to do if they should experience any such problems (e.g. who they can contact for help).		No
Does your study involve any of the following?	Children (under 18 years of age)	No
	People with intellectual or communication difficulties	No
	Patients	No

School of Computer Science and Statistics Research Ethical Application Form

Details of the Research Project Proposal must be submitted as a separate document to include the following information:

1. Title of project
2. Purpose of project including academic rationale
3. Brief description of methods and measurements to be used
4. Participants - recruitment methods, number, age, gender, exclusion/inclusion criteria, including statistical justification for numbers of participants
5. Debriefing arrangements
6. A clear concise statement of the ethical considerations raised by the project and how you intend to deal with them
7. Act etc.

Part C

I confirm that the materials I have submitted provided a complete and accurate account of the research I propose to conduct in this context, including my assessment of the ethical ramifications.

Signed:  Date: 21/03/2019

Lead Researcher/student in case of project work

There is an obligation on the lead researcher to bring to the attention of the SCSS Research Ethics Committee any issues with ethical implications not clearly covered above.

Part D

If external or other TCD Ethics Committee approval has been received, please complete below.

External/TCD ethical approval has been received and no further ethical approval is required from the School's Research Ethical Committee. I have attached a copy of the external ethical approval for the School's Research Unit.

Signed: …… Date: ……. ……

Lead Researcher/student in case of project work

Part E

If the research is proposed by an undergraduate or postgraduate student, please have the below section completed.

I confirm, as an academic supervisor of this proposed research that the documents at hand are complete (i.e. each item on the submission checklist is accounted for) and are in a form that is suitable for review by the SCSS Research Ethics Committee

Signed: Date: ....26/3/19.……

Supervisor

Completed application forms together with supporting documentation should be submitted electronically to the online ethics system - https://webhost.tchpc.tcd.ie/research_ethics/ When your application has been reviewed and approved by the Ethics committee, hardcopies with original signatures should be submitted to the School of Computer Science & Statistics, Room 104, Lloyd Building, Trinity College, Dublin 2.

CHECKLIST

Please ensure that you have submitted the following documents with your application:

1.	• SCSS Ethical **Application Form**	Y
2.	• **Participant's Information Sheet** must include the following: a) Declarations from Part A of the application form; b) Details provided to participants about how they were selected to participate; c) Declaration of all conflicts of interest.	Y
3.	• **Participant's Consent Form** must include the following: a) Declarations from Part A of the application form; b) Researchers contact details provided for counter-signature (your participant will keep one copy of the signed consent form and return a copy to you).	Y
4.	• **Research Project Proposal** must include the following: a) You must inform the Ethics Committee **who** your intended participants are i.e. are they your work colleagues, class mates etc. b) How will you recruit the participants i.e. **how** do you intend asking people to take part in your research? For example, will you stand on Pearse Street asking passers-by? c) If your participants are under the age of 18, you must seek both parental/guardian AND child consent.	Y
5.	• Intended **questionnaire**/survey/interview protocol/screen shots/representative materials (as appropriate)	Y
6.	• **URL** to intended on-line survey (as appropriate)	

Notes on Conflict of Interest

1. If your intended participants are work colleagues, you must declare a potential conflict of interest: you are taking advantage of your existing relationships in order to make progress in your research. It is best to acknowledge this in your invitation to participants.
2. If your research is also intended to direct commercial or other exploitation, this must be declared. For example, *"Please be advised that this research is being conducted by an employee of the company that supplies the product or service which form an object of study within the research."*

Notes for questionnaires and interviews

1. If your questionnaire is **paper based**, you must have the following **opt-out** clause on the top of each page of the questionnaire: *"Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions are responded to."*
2. If you questionnaire is **on-line**, the first page of your questionnaire must repeat the content of the information sheet. This must be followed by the consent form. If the participant does not agree to the consent, they must automatically be exited from the questionnaire.
3. Each question must be **optional**.
4. The participant must have the option to '**not submit, exit without submitting'** at the final submission point on your questionnaire.
5. If you have open-ended questions on your questionnaire you must warn the participant against naming **third parties**: "*Please do not name third parties in any open text field of the questionnaire. Any such replies will be anonymised."*
6. You must inform your participants regarding **illicit activity**: "*In the extremely unlikely event that illicit activity is reported I will be obliged to report it to appropriate authorities."*

TRINITY COLLEGE DUBLIN

INFORMATION SHEET FOR PROSPECTIVE PARTICIPANTS

Background:

You are invited to participate in this study on Artificial Intelligence use in Software Testing which aims to establish how AI is impacting on the Software Testing industry and gain information on how its use will grow across the profession in the next 5 years.

To do this I am gathering information by interviewing managers and leaders in the field of Software Testing to gain expert opinions on the topic. To that end I would appreciate your participation in my study.

Additionally, I am reviewing what functionality is available by AI software test tools and how this functionality differs from established non-AI software test tools currently in use. Through researching specialist Software Testing websites such as saucelabs.com, sqa.stackexchange.com, marutitech.com and gartner.com I have shortlisted 5 market leaders in AI software test tools to review.

Interviews and Study Conducted by:

This study is being carried out by myself, Karl Morrissey in the School of Computer Science and Statistics at Trinity College Dublin. This study is being carried out as part of the requirements for a M.Sc. in Information Systems Management at Trinity College Dublin.

Participant Criteria:

I am seeking participants who are Leaders in the field of Software Testing and who have an active role in assessing tools and methodologies that are used in the Software Test Strategy of the ICT function in their respective companies.

Procedure:

I will be asking specific questions on the technical and human impact of AI in Software Testing, in addition to questions around the adoption of AI tools in IT. Your responses will be used to answer the overall

question as to how AI is being used in Software Testing and how its use will grow in the industry over the next 5 years.

Expected Duration:

I expect the interview to last 30-45 minutes

Risks

I do not expect there to be any risks to the participant and a transcript of the interview will be made available on request.

Withdrawal:

Your participation is entirely voluntary, and you can withdraw at any time without penalty. In order to withdraw, simply email me @ morrisk4@tcd.ie and I will remove your data from my study.

Questions:

Each question is optional. Feel free to omit a response to any question during the interview; however, I would be grateful if all questions are responded to.

Debriefing:

A transcript of the interview will be made available on request by emailing me at morris4@tcd.ie.

Preservation of participant and third-party anonymity in analysis, publication and presentation of resulting data and findings:

The data will be analysed in order to establish expert opinions on the use of AI in Software Testing. The interviews will be audio recorded for transcribing at a later time. No names will be published in my final study. I plan to publish the results of the research in academic journals and conference proceedings. I will do this in a way which does not identify you, or any other individual participant. The research results will be published in a M.Sc. dissertation at Trinity College Dublin. No audio or video recordings will be made available to anyone other than myself, nor will any such recordings be replayed in any public forum

or presentation of the research. Recordings will be saved for a year after which they will be destroyed by ensuring all recording files are deleted, including any back-ups.

Cautions about inadvertent discovery of illicit activities

While it is unlikely that illicit activities would be disclosed, if you do so, we would be obliged report them to the appropriate authorities.

Provision for verifying direct quotations and their contextual appropriateness

Recordings and transcripts will be made available on request by emailing me at morris4@tcd.ie

Declarations of conflicts of interest

As I myself am a Manager within the Software Test industry there is an inherent conflict of interest. I will also be interviewing at least one colleague, which presents a conflict of interest which you should be aware of.

Contact Details of researcher in case of queries

If you have any queries, feel free to contact me, Karl Morrissey at morrisk4@tcd.ie and I will be happy to answer questions about the interview.

TRINITY COLLEGE DUBLIN

INFORMED CONSENT FORM

LEAD RESEARCHER: Karl Morrissey (morrisk4@tcd.ie)

BACKGROUND OF RESEARCH: This research is on Artificial Intelligence use in Software Testing. This research aims to establish how AI is being used in the Software Testing industry and gain information on how its use will grow across the profession in the next 5 years.

To do this I am gathering information by interviewing managers and leaders in the field of Software Testing to gain expert opinions on the topic. To that end I would appreciate your participation in my study.

Additionally, I am reviewing what functionality is available by AI software test tools and how this functionality differs from established non-AI software test tools currently in use. Through researching specialist Software Testing websites such as saucelabs.com, sqa.stackexchange.com, marutitech.com and gartner.com I have shortlisted 5 market leaders in AI software test tools to review.

PROCEDURES OF THIS STUDY: For this study I am interviewing Leaders in the field of Software Testing who have responsibility in the design and implementation of the overall Test Strategy for their company. I will be asking specific questions on the technical and human impact of AI in Software Testing, in addition to questions around the adoption of AI software test tools in IT. There should be no risk to yourself. I will make the audio recordings available to you and I will make a transcript available to you should you wish. No names or identifying data will be included in the published dissertation.

PUBLICATION: The research results will be published in a M.Sc. dissertation at Trinity College Dublin. No audio or video recordings will be made available to anyone other than myself, nor will any such recordings be replayed in any public forum or presentation of the research.

CONFLICTS OF INTEREST: Participants should will be aware that I am a Manager in the field of Software Testing myself. Participants who are colleagues will be aware of my role in the company in which we work.

Individual results may be aggregated anonymously and research reported on aggregate results.

DECLARATION:

- I am 18 years or older and am competent to provide consent.
- I have read, or had read to me, a document providing information about this research and this consent form. I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
- I understand that if I make illicit activities known, these will be reported to appropriate authorities.
- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty.
- I understand that if the results of the research have been published, or my data has been fully anonymised so that it can no longer be attributed to me, then it will no longer be possible to withdraw
- I understand that I may stop electronic recordings at any time, and that I may at any time, even subsequent to my participation [request to] have such recordings destroyed.
- I understand that, subject to the constraints above, no recordings will be replayed in any public forum or made available to any audience other than the current researchers/research team.
- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
- I have received a copy of this agreement.
- I am participating in a personal capacity and not on behalf of my organisation.

By signing this document, I consent to participate in this study, and consent to the data processing necessary to enable my participation and to achieve the research goals of this study.

PARTICIPANT'S NAME:

PARTICIPANT'S SIGNATURE:

Date:

Statement of investigator's responsibility: I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

RESEARCHERS CONTACT DETAILS: email: morrisk4@tcd.ie

RESEARCHER'S SIGNATURE:

Date:

TRINITY COLLEGE DUBLIN

Research Project Proposal

Title of project:

An Investigation into the impact of AI-Tools on the Software Testing Profession

Purpose of project including academic rationale

The purpose of this study is to determine how Artificial Intelligence can be used in Software Testing. By reviewing the literature that is available on the topic, reviewing AI software test tools that are currently available and interviewing leaders in the field of Software Testing I will gather information to synthesis my dissertation.

I am reviewing 5 AI software test tools that are currently on the market and have been identified as leaders in the field by online reviews from specialist Software Testing sites, and the research and advisory company, Gartner.

I plan to review the tools along the following categories and rationale. To structure this set of criteria I referred to marutitech.com, gartner.com and checklist.com/software-evaluation-checklist/ for related evaluations.

Category	Rationale
Cost Benefit	Rationale: The cost of implementing the tools might outweigh the benefit or any tool, which would influence the current impact of the AI tools in the industry
Ease of implementation	Rationale: The complexity of implementing the tools might outweigh the benefit or any tool, which would influence the current impact of the AI tools in the industry

Ease of Use	Rationale: Current tools take effort, time and skills to implement. Much of a tester's time is taken with ensuring scripts run smoothly and are maintained. If these tools are easy to use and reduce this effort then the implications could be: • Testers are freed up to design better test cases, or carry out other tasks • Organisations might be presented with a business imperative to reduce or eliminate the number of testers they employ
Available Features, such as test **case automation and generation**	What features are the tools offering and how can they be of benefit? Can they generate more efficient tests, reduce time in test creation, reduce cost of maintenance and expand code coverage?

I will be asking specific questions on the technical and human impact of AI in Software Testing, in addition to questions on the adoption of AI software test tools in IT. Participants should be aware that each question is optional.

Participants should feel free to omit a response to any question; however, I would be grateful if all questions were responded to.

The questions are as follows:

1. Most organisations are still implementing a traditional software automation test strategy, where testers manually code and maintain automated scripts and use a tool like Selenium for organisation and execution. In your opinion, given the current maturity of Artificial Intelligence how can it be used in Software Automation?
2. There are now several AI-Driven Test Automation tools on offer in the market place. Has your organisation implemented any of them, or is it planning to? If not, why? If so, what was the business case for doing so and have you carried out a Cost-Benefit analysis? Do you think the service the tools provide is worth the cost?
3. Given the potential of AI software test tools what specific features do you think could impact on your current Test Strategy and what benefit would you like to see. For example, do you think they will make it easier for software testers to automate test cases? Do you think that these tools will enable faster execution times, and therefore faster product development? Do you think these tools

will be difficult to integrate with existing infrastructure? Are there any other impacts you can think of, both positive and negative?

4. What impact do you think the introduction of AI software test tools will have on the Software Test profession (human-impact). Do you think it will lead to a reduction in the number of Software Testers employed by IT organisations? What positive impacts do you envisage for Software Testers? What negative impacts do you envisage for Software Testers?
5. How do you see AI evolving within the Software Test profession in the next 5 years? Do you believe that AI software test tools will gain traction in the market? Do you believe that there will be an inflection point whereby organisations will adopt AI tools across the industry, replacing much of the current non- AI software test tools?

Participants will be Leaders in Software Testing who are responsible for Tooling and Software Test Strategies in their organisations. Selections will be made based on contacts that are known to me already in the IT industry. At least one person will be from my own company, others will be individuals I have worked with in the past. I intend to contact the participants directly by email and phone.

Debriefing arrangements

- Interviews will take place in person if possible, or over the phone and will be audio recorded
- Recordings and transcripts will be made available to the interviewees on request by contacting me at morrsk4@tcd.ie.
- I envisage interviews to last 30-45 minutes.

From an ethics perspective, as I myself am a Manager within the Software Test industry there is an inherent conflict of interest. I will also be interviewing at least one colleague, which presents a conflict of interest which candidates should be aware of. All identifying information will be removed from the final publication of the dissertation and interviewees can withdraw their participation as outlined in the Informed Consent Form.

This study complies with the General Data Protection Regulation Act 2018