Expertise
and insight
for the future

Mikael Mäkelä

Utilizing Artificial Intelligence in Software Testing

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

24 November 2019

PREFACE

By selecting this topic, *Utilizing Artificial Intelligence in Software Testing* for my Master's thesis, I am looking to the future. The subject was relatively new for me from the AI point of view, as it might be for most of the software testing specialists. Anyhow I believe this is the direction to which the software testing is going and testers should be aware of it.

Software complexity has increased tremendously with the technology updates, but from my perspective software testing has not been able to follow with it. Software testing is still performed pretty much with the same methods as it has been done a decade ago, even though the requirement of testing more and faster has grown. Different AI driven software testing approaches have been researched and AI has taken more role in the testing tools developed during the last few years. It is not clear in what scale the AI will be used and how autonomous the AI test systems will be in the future, but my personal interest is staying on the drivers seat on this development.

Thanks for Sami Kaltala and Pekka Vainiomäki for the presentation giving an example of using AI in software testing in FiSTB Testing Assembly 2018. This caught my interest in the topic and gave me a focus in my studies. Secondly, I want to thank Metropolia and all the teachers for the courses assisting on this path and the instructor Auvo Häkkinen for guiding in this writing process. Finally, I want to thank my wife and family for making this year off from work for studies possible and finding this new spark in the software testing and systems development.

Espoo, 24.11.2019
Mikael Mäkelä

Author Title Number of Pages Date	Mikael Mäkelä Utilizing AI in Software Testing 60 pages + 4 appendices 24.11.2019
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Auvo Häkkinen, Principal Lecturer

Artificial Intelligence has become more and more important part of computer science and IT business. Anyhow, it has not been used in the software testing widely.

At the same time the demand for using the test automation and testing more efficiently has increased. Using the AI and machine learning could be one approach to reduce the manual work in the software testing and also in the traditional test automation which still requires tremendous amunt of manual work. However, Artificial Intelligence and machine learning technologies still are pretty unknown for the major part of the software testing community.

The primary objective of the study was to form recommendations how the artificial intelligence could be utilized in the software testing. The secondary objective was to present the methods as part of standardized test process.

The research was conducted by creating the research keywords based on the testing standards and best practices. The different machine learning solutions related to software testing and its sub categories were searched both from the academic publications and online materials of the test tool providers, conference recordings and trainings provided by the AI testing specialists.

The actual search results are collected and categorized in four sections. In the first section as a theoretical background the AI and machine learning high level approaches have been presented and also challenges on testing the AI and with an AI.

In the second section the AI Testing model by AI for Software Testing Association is presented and reflected to a Paul Gerrard´s New Model for Testing.

In the third section the fundamental testing process has been used as a frame in which the different machine learning approaches from the other researches have been collected.

In the last section the different testing tools are presented and how the machine learning could be used to improve the efficiency using them. The section also presents some existing commercial and open source tools that are having macine learning features in their implementations.

Keywords	artificial intelligence, AI, machine learning, software testing, test automation

Tekijä Otsikko Sivumäärä Aika	Mikael Mäkelä Utilizing AI in Software Testing 60 sivua + 4 liitettä 24.11.2019
Tutkinto	Master of Engineering
Koulutusohjelma	Information Technology
Ohjaaja	Auvo Häkkinen, yliopettaja

Tekoäly on tullut päivä päivältä tärkeämmäksi osaksi tietojenkäsittelytiedettä ja ohjelmistoalaa. Kuitenkaan sitä ei ole hyödynnetty laajasti ohjelmistotestauksessa.

Samaan aikaan testiautomaation käyttöä on pyritty lisäämään ja ohjelmistotestauksen tehokkuutta parantamaan. Tekoäly ja koneoppiminen voisivat olla keino saada testauksen manuaalista työtä vähennettyä ja myös testiautomaatiota tehostettua, joka vaatii tänä päivänä huomattavan määrän manuaalista työtä. Kuitenkin tekoäly ja sen hyödyntäminen on vierasta suurimmalle osalle ohjelmistotestausyhteisöä.

Tämän opinnäytetyön päätavoite on esittää suosituksia, miten tekoälyä voidaan hyödyntää ohjelmistotestauksessa. Toinen päätavoite on esittää menetelmät osana standardoitua testausprosessia.

Tutkimus on toteutettu keräämällä avainsanoja ohjelmistotestauksen standardeista ja parhaista käytänteistä. Näiden avulla haettiin tekoälyratkaisuja akateemisista julkaisuista, verkkomateriaaleista esimerkiksi ohjelmistoyritysten verkkosivuilta, konferenssitallenteista sekä tekoälytestaukseen erikoistuneiden henkilöiden pitämistä koulutuksista.

Tutkimus on kategorisoitu neljään osaan. Ensimmäisessä osassa kuvataan teoriataustatietona tekoälyn ja koneoppimisen perustietoja sekä kuvataan haasteita, joita tekoälyn testaaminen ja sen avulla testaaminen asettavat.

Toinen osa esittelee AIST yhteisön luoman tekoälytestauksen mallin, jota peilataan Paul Gerrardin luomaan malliin ohjelmistotestauksesta.

Kolmannessa osassa ohjelmistotestauksen prosessia käytetään pohjana esittelemään ja yhdistämään eri tutkimuksissa ja lähteissä kehitettyjä koneoppimismenetelmiä yhdeksi kokonaisuudeksi.

Viimeisessä osiossa on esitelty eri testauksen työkaluja, ja analysoitu miten eri koneoppimismalleilla niiden tehokkuutta voisi parantaa. Kappaleissa myös esitellään eri kaupallisia ja avoimia työkaluja ja miten tekoälyä on hyödynnetty niissä.

Avainsanat	tekoäly, ohjelmistotestaus, koneoppiminen, automaatiotestaus

Contents

List of Abbreviations

AI	Artificial Intelligence. There is no official description available. The characteristics of the AI are autonomy and adaptivity. With AI the system can act independently and adapt to the changes.
AIST	Artificial Intelligence for Software Testing. “An emerging field aimed at the development of AI systems to test software, methods to test AI systems, and ultimately designing software that is capable of self-testing and self-healing.” (AISTA, 2019)
AISTA	Artificial Intelligence for Software Testing Association. An association founded by Tariq King and Jason Arbon to support the research for AI driven testing, testing AI systems, and systems self-testing.
AI Testing	AI driven testing, the testing activity in which Artificial Intelligence has been used
ANN	Artificial Neural Network. Simulated neural networks in a computer modelled from the biological neural networks in our brains.
DevOps	DevOps is a culture, principle, and process that automate and streamline the software development lifecycle from code development to production.
FiSTB	Finnish Software Testing Board. A local organization of the ISTQB to support testing professionalism in Finland by arranging certification exams, seminars and co-operating with the schools.
GPU	Graphical Processing Unit. A microprocessor primarily used to process the computer graphics but can also be used with training the machine learning systems.
ISTQB	International Software Testing Qualifications Board. A certification organization for the software testing professionals.
MBT	Model Based Testing. A software testing approach which is based or involves using models describing the application or system behaviour.
ML	Machine Learning. A method where data is fed to a system that is using it to learn how to perform a specific task.
NLP	Natural Language Processing. A machine learning approach to process natural language data.
QA	Quality Assurance. A quality management activity to gain confidence in filling the quality requirements.
TA	Test Automation. An approach of the software testing where the test process is executed completely or partly automatically. Traditionally test automation has been used in test execution and result validation activities.

1 Introduction

Discussions about artificial intelligence (AI) has taken more and more space during the last years. Artificial Intelligence has become more and more important part of computer science. The words such as artificial intelligence and algorithm have become more common in everyday discussion due to social media, AI ethics etc. Anyhow the Influence of the AI in the field of software testing has been low.

Test automation (TA) is considered useful and, many times even mandatory especially in following DevOps and agile product development models. This is based on the principle of delivering high quality working software in short development cycles and minimizing the work that does not bring value. The benefits of the test automation have been the reduction of repetitive manual work and in that way saving time in the testing. The tests are executed consistently and are repeated exactly the same way each time. The test automation is more objective assessment and gives easier access to the information that the testing generates in the forms of graphs about progress, defects and performance. (ISTQB, 2018, p. 81) Even though there are a lot of benefits in the test automation, maintaining the traditional test automation requires a lot of manual work. (King, et al., 2019) At the time of starting the study there were only a few tool sets available using machine learning (ML), targeting to minimize the manual work in the TA. Machine learning and its methods have been available for a long time already, but still only a few people have put effort into how it could support the testing.

“AI will fundamentally transform software testing. Testers make quick judgments based on data. AI makes even quicker judgments based on far more data. No area of software testing will be left unchanged by AI” (Arbon, 2018), claims Jason Arbon as he describes how AI influences in the Software testing in the future. This study is a collection of models, methods and tools to support this vision to become reality.

“Software Testing – Dynamic program verification using finite tests from an infinite execution domain.” In his definition of software testing, Tariq King highlights the problem in dynamic testing. In his opinion a test specialist has an impossible task to test everything. The software complexity increases exponentially when new functions, applications and their integrations are added in the system. Conventional test coverage

in the normal manual testing can increase only linearly when the amount of test resources remain the same. Therefore the test coverage gap increases. This is where the AI and intelligent test agents could come to support the testing. (King, 2019)

Even though there is a need to increase the automation in the testing activities, there are major risks in it. When the artificial intelligence is brought to the test automation, these risks need to be taken into consideration more closely. ISTQB has in its certification material a good collection of risks related to the test automation. These are reflected to the findings in this research in the chapter 4.6.

There are many questions needed to be answered in this research. What is artificial intelligence? Which forms of AI can be integrated to software testing? How could AI implementations serve the testing process, test planning, test execution on different test levels and testing different kind of quality characteristics that our application or system has? Can an AI system be intelligent and reliable enough to validate the test results? What kind of activities could be feasible to be automated with artificial intelligence?

One of the main questions is: what kind of requirements does the utilizing of the AI set for the system development process, testing process in specific and for the development team? As mentioned before, the DevOps and Agile methodologies drive the test automation as much as possible or feasible. Therefore, the study and its results are targeted for the persons interested and utilizing these methods.

The primary objective of the study is to form recommendations for how the artificial intelligence could be utilized in the software testing. The secondary objective is to present the methods as part of standardized test process and link the different approaches and terms as one common language for the testing and for the whole system development community.

In my work as a test engineer and specialist I have found it useful to use a common language and terminology within the software testing community. It was probably after a couple of months after starting to work as a test engineer as I was directed to a course arranged by ISTQB International Software testing board targeting the foundation level certificate. I learned the basics of the testing process and beneficial tools that in my opinion every tester should use. I have noticed many testers using these processes and tools, but the problem comes when discussing about the testing. What does a tester

mean about functional testing, system testing, integration testing, and performance testing. If there is no common agreed language it might take time for sharing the understanding what is done, how and when. Therefore, I try to link the AI tools and methods for different areas of the testing process and methods thought by the ISTQB in its certification training material.

The scope of the study consists of studies on AI and Software testing. It is also analysis on the literature studies to present the suggestions on utilizing the AI in the software testing as a methods of implementing the fundamental testing process and its phases.

1.1 Justification for the Study

World Quality Report 2018/2019 highlights the importance of AI and its influence in the testing in the near future. The report also highlights the challenge of implementing the AI in the testing. According to Van De Ven, et al. “AI is an emerging technology and the knowledge, expertise and maturity to apply it to quality assurance (QA) is lacking in many organizations”. (Van De Ven, et al., 2018)

Artificial Intelligence for Software Testing Association (AISTA) also sent a survey in 2017 to its members. At that time, only 22% of the respondents were acquainted with AI and machine learning technologies, but 76% of the respondents believed that “AI will have significant impact on manual testing within 3 years”, that is until the end of year 2020 (King, et al., 2019). As the survey was sent to the people interested in how to apply AI in software testing and/or working with AI and its implementations in testing, the responses do not represent the whole testing community and actual tester’s interest or knowhow on AI or machine learning technologies.

Both surveys provide evidence that I was not the only software test engineer without AI expertise in the field of software testing when starting this study. Also, they justify the research and spreading the information for the whole software testing community. This research provides ideas on how the AI could be utilized in different projects or system development processes in different organizations in practice and offer a basis for further analysis and studies in this area.

1.2 Research Process and Methods

In the beginning of starting the research there was quite small amount of information available about the software testing and its relation to artificial intelligence. It was fragmented and not systematically described or categorized. The first task for the research was to create a testing and artificial intelligence keyword structure in forms of processes, techniques, tools and use targets on which the research and its suggestions could be built, and results presented.

The keyword structure for the software testing in this research is based on ISTQB certification syllabi, and the glossary (ISTQB, 2018). The ISTQB material is based on the testing standards. These have been added as a reference in the appendices. The ISTQB allows copying the content in the documents if the source is acknowledged. Therefore, the actual terms and definitions are used to prevent misunderstandings and errors in the interpretations. These definitions have been thought thoroughly and tested by multiple test specialists around the world.

ISTQB has not taken part in the discussion on AI driven software testing and its terminology. In these cases, the referred terms and abbreviations in the source material will be used and reflected to the ISTQB terminology if appropriate.

There has been criticism towards the testing standards. Some say that they represent an old-fashioned view and do not fit in the agile product development models, for instance due to requirements for too much documentation. According to the convenor of ISO Software Testing Working Group (WG26), Dr Stuart Reid, “the standard is fully aligned with agile development approaches and users taking a lean approach to documentation can be compliant with the standard”. The development team has the freedom to use whatever document naming and format that best fits the organization. The standard is a guideline of best practices unless the organization demands following it strictly. (Reid, 2014)

The research process as its initial plan is presented in the diagram below (Figure 1)

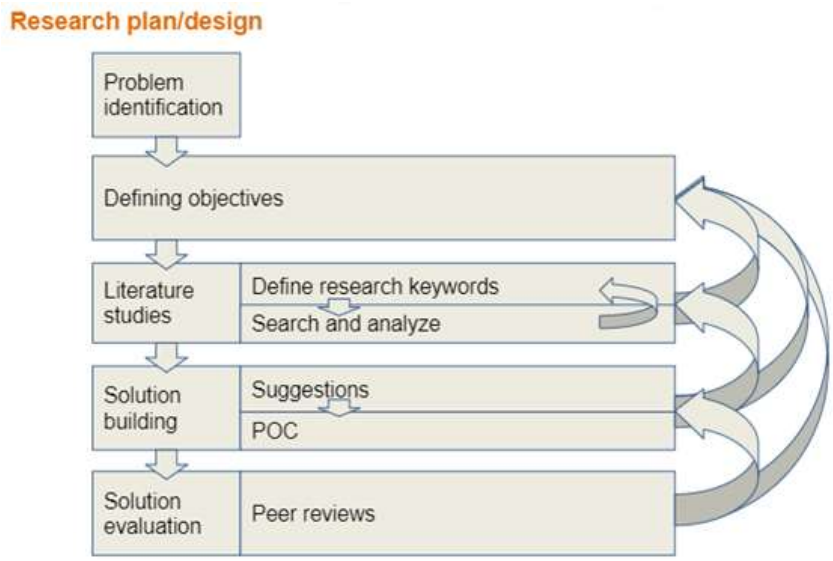

Figure 1 – Initial research plan/design

The research process is divided into four main phases. The objectives were initially defined in the approval of the research plan and remained the same during the research.

In the literature studies first the testing and AI specific keywords were gathered for the actual search of the different approaches using machine learning in the software testing. The literature studies formed the basis for the research and created the end structure for this document.

This research is based on qualitative methods. The research material in the literature studies is based on the scientific research, papers written on the conference publications and online material. The online material consists for example material from the test tool providers, different videos of the presentations in the conferences, video recordings of the trainings arranged, and presentations of the solutions built. As the development on using AI in different fields of systems development (including testing) has been rapid during the last few years, the online material (e.g. YouTube videos, blog posts) has led to the newest information. This information requires critical thinking and should be cross-referenced and reflected to the scientific research on the subject.

The collected keywords are presented in the appendices of the research.

In the solution building phase, the findings of the literature studies are formalized and presented. First the AI driven testing as a model or architecture was presented. Secondly the suggestions on how artificial intelligence could be used in different testing activities, were presented based on the fundamental test process. In this phase also the traditional testing tools are presented and reflected to the AI testing tools. Also some commercial and open source testing tools are presented that implement the machine learning in the solutions.

The solution evaluation for the reliability and validity was initially planned as basing on the peer reviews conducted with the other test specialists and TA engineers. During the research and writing the section of existing knowledge it was clarified that the testing community is lacking on the skills on AI and machine learning. Therefore, it would have been good to have a review from a person having experience on AI and QA Strategies, data sciences and AI testing.

2 Existing Knowledge and Current State Analysis

This section is an overview and a high-level approach for the AI driven testing. It contains the survey analysis of the world quality report, interpretations on the results and comparison to the survey sent for the AI testing community. In the end it presents the Artificial Intelligence for Software Testing (AIST) Association behind the survey and the active persons who have been acting towards the AI testing.

2.1 Status and Trends on the Software Testing on 2019

Three companies MicroFocus, CapGemini and Sogeti that are acting on software test tooling and test services publish annually a paper called World Quality Report. The paper describes the current status and trends driving the software testing and product quality. It is probably one of the most remarkable publications on the field, but also collect criticism in the public like Lee Hawkins in his blog text. (Hawkins, 2018)

The three key findings in the year World quality report 2018-19 are:

1. End-user-satisfaction is the primary objective of the quality assurance and software testing strategy
2. The software testing efficiency is held back due to challenges with the level of test automation, test data and test environments
3. The skills needed for the quality assurance and software testing have changed

For two of these there is a direct relation with the AI, both with its benefits and the challenges it influences into the organizations and their testing activities. Mark Buenen and Ajay Walgude present in the report's executive summary the importance of the AI (Buenen & Walgude, 2018), whereas Van De Ven et. al, broaden the reasoning in the recommendations (Van De Ven, et al., 2018).

There are huge expectations for AI to improve the software testing efficiency (Key finding 2). AI is seen as a method for transforming the testing into self-generating, self-running and self-adapting activity. This is in line with the findings of this research in forms of solutions to support the testing activities.

The challenges in efficiency of the QA and testing have been identified in low level of automation, challenges with the test data and test environments and too slow testing process. These have specifically challenged the companies using agile product development models holding back QA and test efficiency. AI is seen as a solution to derive full benefits from the Agile and DevOps models.

Survey highlights that the utilizing AI is in its initial state, but still the companies have quite impressive plans for using the AI. 57% of the survey respondents said that they had “projects involving the use of AI for QA and testing, already in place or planned for the next 12 months.” The planned methods include intelligent test automation (45% of the responses), predictive analytics (36%) and descriptive analytics in testing (35%).

The initial status of AI in testing leads to the third key finding of the survey, which leads to updating the testing strategy and skills needed in the testing with AI.

“AI developments require a new specific approach to validation and verification” and “AI is going to be one of the biggest trends in QA and testing for the next two to three years and organizations will need to develop a strategy around it.” – (Buenen & Walgude, 2018)

This new testing strategy should include aspects on gaining a certain level of maturity in the automation, implementing analytics to support the test planning and improve the efficiency, and a strategy for working towards creation of self-learning and self-aware systems that can be applied to testing.

For creating the strategy and implement it in use the new skills and roles should be brought as part of the development and testing teams:

AI QA Strategist, who understands the implications of AI for the business processes. This person uses his business knowledge and understanding in data management and data processing techniques as mathematical optimization, natural language processing and robotics to derive the testing strategy for the applications.

Data scientists, who sift through data and use predictive analytics, mathematics and statistics to build models.

AI test experts, who own the traditional technical testing skills and can build machine learning algorithms, mathematical models and NLP models to plan and execute the tests based on the test strategy.

2.1.1 Criticism Against the Survey and its Results

The survey can be criticized as most of the replies are coming from the very largest corporations and don't represent the small companies or start-ups, in which the testing methods and processes might differ greatly.
When taken a look at the responder's role or job title:

- CIO (27%)
- IT director (22%)
- QA/Testing Manager (20%)
- VP Applications (18%)
- CMO/CDO (7%)
- CTO/Product Head (6%)

It is possible that other than QA/Testing Managers don't work with the actual testing activities in the organizations. (Hawkins, 2018)

2.2 Status and Trends in the AI Driven Testing Today

When comparing the results of the survey produced by AISTA to the world quality report the analysis highlights the same results on the competence of using AI in the software testing and need for the AI and data specialists in the development and testing teams.

As stated in the introduction in 2017 only 22% of the members of AIST Association were familiar with AI and machine learning technologies, but 76% believed that AI has significant impact on manual testing within the next 3 years (King, et al., 2019).

It was also asked in the survey if the AI will replace all manual testing and when would this happen. The responses were quite polarized. 23% of the respondents said this will happen already on 2025, but 35% said it will never happen. Half of the 23% were manual testers whereas almost all of 35% were test automation engineers with most likely more technical background compare to manual testers. Are the manual testers hopeful to give

up their jobs to AI and TA engineers pessimistic based on their experience on challenges with current test automation? (King, et al., 2019).

Regardless the opinions, there has been a significant increase in the companies providing AI supported testing tools in the market. The first companies offering AI-driven testing services came into market on 2014. Most of these companies are start-up companies offering system level test tools for the mobile applications. (King, et al., 2019)

The most of the current AI approaches in the tools are aiming to validate the differential changes of the application over the builds, visual testing on the look and feel of the applications and self-maintaining the test cases when the UI changes. These are mainly used for the regression testing purposes, but there also are applications featuring declarative testing, where machine learning has been used to “specify the intent of the test in natural or domain-specific languages”, which can be further taught to the test agents to execute the flow. (King, et al., 2019) These approaches are further discussed and analysed in the research.

2.3 Associations to Support AI Driven Software Testing

There are multiple associations to support AI and machine learning development or the software testing, but in the research unfortunately only one association combining these two practices was found; Artificial Intelligence for Software Testing Association (AISTA). Fortunately, there is also other open source communities and universities making tools and researches around the topic. Also, the companies developing the AI testing utilities support the open research for the AI driven testing. For example, Test.ai and Ultimate Software have open-sourced some of their solutions. Also, many other tool vendors and their employees publish blog posts and articles in the company web sites.

The Artificial Intellicence for Software Testing Association (AISTA) is founded by Jason Arbon (CEO at Test.AI) and Tariq King (Head of Quality at Ultimate Software). According to the blog posts by the founders the association is founded on 2017. The founding members publish articles and blog posts and give trainings and speeches at the conferences on the AI Testing. Even though the association seems quiet according to the frequency of published posts in the web site, the founders publish other material in the internet, present in the conferences and send newsletters concerning the AI testing.

Test.ai is a company founded by Jason Arbon et al. According to Jason Arbon “the goal of the Test.AI is to test all the worlds apps” (Arbon, 2018). Test.ai publishes monthly newsletter related to software testing and AI. Test.ai also has open sourced some of its machine learning classification training data, training process and has created an API integration with Appium, application testing tool (Arbon, 2019). Test.AI can somewhat be considered as an association to support the AI driven testing even though it is a company responsible for its stake holders and investors.

3 Testing AI and Testing with an AI - Theoretical Background

To answer the question what is an AI for Software testing, the AISTA has written a good definition: “An emerging field aimed at the development of AI systems to test software, methods to test AI systems, and ultimately designing software that is capable of self-testing and self-healing.” (AISTA, 2019)

Testing on AI system requires run time testing as the systems learn and adapt on the changes. Self-testing is runtime testing in adaptive systems. Tariq Kings idea of self-testing consists of self-configurable, self-healing, self-optimizing and self-protecting autonomous test bots that are built as part of the system. (King, 2019)

The following chapters give an overview of the Artificial Intelligence and different approaches for the Machine Learning.

3.1 Characteristics of Artificial Intelligence

Artificial intelligence can mean different things to different people. There is no exact definition for AI and so it could be better to describe AI with its key characteristics, autonomy and adaptivity. In a scientific manner AI is a subfield of computer science. Its related fields are machine learning, which is a subfield of AI, deep learning which is a subfield of machine learning, data science and robotics, which requires aspects from all the fields of AI. (University of Helsinki & Reaktor, 2019)

Russell and Norwig have taken the concept of rational intelligent agent as an approach for describing artificial intelligence. The agent is an autonomous entity can perceive its environment by using sensors and acts towards goals specific in its environment. How the goal is defined depends on the agent architecture. For example, the goal of the simple reflex agent is to act based on condition-action rules. The most advanced learning agent specifies its goals based on learning from its environment. (Russell & Norwig, 2016)

Machine learning can be said to be a subfield of AI but to be more precise it is a subfield of computer science and some parts of machine learning are subfield of statistics. Machine learning systems enable AI systems adaptivity by creating models from data.

Machine learning solutions are heavily dependent on available data which improve the learning performance. Machine learning approaches form the basis for the robotics related AI problems (University of Helsinki & Reaktor, 2019)

According to Meinke and Bennaceur machine learning has been successfully used in many areas of software engineering as presented in the diagram below. (Figure 2)

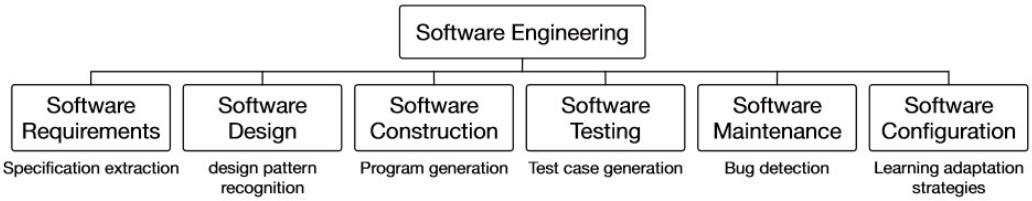

Figure 2 - Applications of machine learning for software engineering (Meinke & Bennaceur, 2018)

This diagram shows the areas where machine learning has been successfully used. It was presented to audience in International Conference of Software engineering 2018 (Meinke & Bennaceur, 2018). It is reasonable to say that the number of different machine learning applications in software testing has increased from the test case generation to test planning, prioritization, execution etc.

Supervised learning methods are used with classification and regression problems. In the classification problems the task is to predict the correct output or label for the input. The model is created by taking number of samples, labeling them and supervisor trains the model based on the sample and label pairs and after this training the model can independently produce label for the inputs. For example, algorithm could give an answer to a question is an image of an animal a cat. In the regression problems the predicted output is a number and used for example predicting number of occurring traffic accidents based on the conditions on road and weather.

A good practice to train the model is to split the dataset into training data and test data. The training data is used to train the algorithm to create a model or rules that can predict the outputs based on the input variables. To make sure that the model can be generalized test data can be used by running the model against it and by comparing the result of predictions to actual outputs. This gives the rate of confidence to the algorithm.

Overfitting can occur when the rules come too complex and restricted. The model works perfectly with the training data but does not work with the test data. Avoiding overfitting

is one of the most essential skills of the data scientist. (University of Helsinki & Reaktor, 2019)

In the unsupervised learning the correct answers are not available. In this case evaluating the model performance is more difficult. The method is used for example in clustering problems where the similarities in the input data are identified and forming clusters separated from the data in the other clusters.

Another example of unsupervised learning is generative modeling. This is based on generative adversarial networks and can be used for example creating artificial images of people faces.

Semi-supervised learning is a combination of the two previous methods. In the training both labeled and unlabeled data is used. In Pseudo-Labeling technique the first training of the network is done with manually labeled data set. This part is still regular supervised learning practise. After this the unlabeled data set is run through the prediction process and outputs are added as labels of the input. This step is called pseudo labeling. After this both data sets are combined, and the training is done as if the complete data set is manually labeled in the means of supervised learning.

3.2 AI Testing Basics

The AI builds the layer of abstraction on top of the software being tested the same way as human brain creates an abstraction of the software and its expected behaviour. Human brain has a notion and understanding that for example a web shop application has shopping items with pictures and descriptions, search fields, shopping basket, login feature, and other features common between each. We want the machine think and act like a human being and in this task the AI is a good asset. (Arbon, 2018)

According to King et al. the “key advantages of AI-driven testing are that it is general purpose, reusable, robust, adaptive and scalable”. Machine learning approaches can be used on testing different test levels, quality attributes, and applications on different domains. The same tests can be used on multiple applications within a domain or between domains due to the independence. Use case examples of the previous are adding items to a shopping cart, or a login form functionality. With for example reinforcement learning methods new tests can be created continuously. This prevents

so called pesticide paradox which can be a problem with traditional test automation. Pesticide paradox occurs when the same tests are executed repeatedly, and test automation becomes blind for new bugs. By combining AI driven test generation and running multiple test agents from the cloud the test coverage can be significantly improved. (King, et al., 2019)

3.3 Benefits of AI Testing for the Agile Development

Agile methodology is a software development model based on the Agile Manifesto which has been written against documentation driven, heavy weight software development process. Whereas agile concerns the development of a software the DevOps acts as a bridge between the development and operations.

The Agile software development method is derived by twelve principles from which the customer satisfaction and continuous delivery of value is on the highest priority. (Authors of the Agile Manifesto, 2001)

The principles that can benefit from use of AI are:

- “customer satisfaction and continuous delivery”
    - Need for continuous testing, which is supported by the autonomy of an AI system
- “Welcome changing requirements”
    - Adaptivity of the AI enables changing the tests when the requirements and therefore application features are changed
- “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
    - Automated testing shortens the required time for testing. Adaptivity of an AI on the other hand reduces the time on maintaining the automated tests.
- “Working software is the primary measure of progress”
    - This highlights the importance of the QA and testing activities.
- “Continuous attention to technical excellence and good design enhances agility.”
- “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly”
    - The previous two principles support the development team finding better ways of doing the work and reducing the work that does not bring value to the customer. This leads to principle that whatever can be automated,

should be so that human expertise can be focused where the automation cannot be used.

3.4 Testing the AI - Characteristics and Challenges

The following chapters describe the challenges the nature of AI sets for the testing the AI and in using it within the software testing.

When creating AI tools to support the testing one should consider testing the test system thoroughly and understanding the testing of AI is highly important. The AI systems usually are, according to King et al. “complex, non-deterministic and self-adaptive”. They also create challenges for the testing with the “feature computation, sampling quality, outlier tolerance, label quality, quality drift and traceability”. (King, et al., 2019)

In the presentation on STAREAST 2019 Jason Arbon quotes Michael Bolton as following,” The key thing in testing AI based Systems is critical thinking.” This applies in the testing the data, testing the output, and even testing the impact on humanity. Now that AI based testing systems are being built even more critical thinking is needed for letting the AI test the AI. (Arbon, 2019)

There can be multiple reasons for the machine learning algorithm to produce incorrect output. In most of the cases the reason is due of insufficient amount of training data for example biased data and this could lead to fairness issues. The other reasons are unsuitable machine learning approach or architecture, the machine learning architecture has learnt a wrong function or due to an implementation bug. As said the primary reason is the insufficient data and the solution is to increase the diversity in it. (Dwarakanath, et al., 2018)

According to Marijan et al. in their research the discussion on testing machine learning systems has been divided in two scientific communities, the machine learning community and the software testing community. The researchers highlight the issue that these two communities interpret the term testing differently. For the machine learning community the testing focuses on machine learning model to estimate its prediction accuracy and improve its prediction performance. The testing happens in the machine learning model training process according by community best practices of splitting the dataset in training, testing and in some cases validation data sets. For the software testing community, the

testing focuses on any given inputs even when the expected result is unknown or not specified. In the integration or system level testing the machine learning component is tested in interaction with other system components or as a complete system. In this all the software quality attributes should be taken in consideration. Marijan et a. highlight in their research correctness, robustness and reliability. (Marijan, et al., 2019)

How the two communities interpret the testing, this support standardizing the testing terminology. The testing team in the future consists of data specialists, machine learning architects, software testers, business specialists. It is a benefit that all these communities can have a common language for the testing activities.

In the research Marijan et al. highlight three fundamental challenges for the test specialists. These are presented in the following chapters (Marijan, et al., 2019)

Probably the biggest challenge for testing AI and using the AI in the software testing is the absence of the test oracles. Machine learning systems are non-deterministic and defining expected values is difficult. The correctness of the output cannot be easily determined. There are some testing techniques used for testing this kind on non-testable systems; differential testing technique and metamorphic testing.

The challenge in the large input space is how to select the appropriate set of inputs and combinations that can reveal the defects in the solution. This also complicates the test oracle problem. There are researched solution examples that apply coverage metrics on neural networks, but these quite easily lead to combinatorial explosion. There is a research done also on combinatorial testing techniques which could lead to promising results on assisting in this challenge.

White-box testing of an AI system requires high test effort. Due to this reason the researchers have proposed on black box techniques instead. One recommended technique is adversarial testing technique. This uses “perturbations as inputs to generate adversarial examples for testing the robustness of machine learning models”. Even though the method has been used widely in image classification model testing it can be expanded in other formats as well like text and audio.

One quality attribute still not recognized by the software quality standards is fairness and the ethics of the system. This is a unique characteristic for the AI system simulating a

human decision making. This can occur when using biased training dataset and often the cause is found in cognitive biases of humans. Also, it is possible that some biases are coming to the training data from other AI algorithms. (Johnson, 2018)

Johnson highlights in his article the “need of methodologies that provide greater trust, transparency, and control of AI's fairness - fairness being the positive goal attained by eliminating the negative effects of bias and other inequities.” (Johnson, 2018)

There are open source communities providing program suites to help gaining the fairness in AI systems for example IBM open sourced AI Fairness 360. Some of the tools in addition to aiding fair results provide information on their decision and explanation how they came in their conclusion to giving transparency in the decision-making process. (Johnson, 2018)

3.5 AI Testing Impact on the Traditional Testing

In the following sections the different aspects of software testing having influence by AI driven testing and research on the subject are presented. The traditional testing with its terminology has been presented as ISTQB presents it in its certification material. (ISTQB, 2018)

3.5.1 The Test Approach

The test approach is “the implementation of the test strategy for a specific project” The ISTQB classifies test approaches in three different categories, Design-Based Testing, Risk-Based Testing and Model-Based testing.

The Design-Based testing approach is probably the most commonly used approach in the field of software testing. The test cases are “designed based on the architecture and/or detailed design of a component or system (e.g., tests of interfaces between components or systems).” One proposed method and enabler for AI understanding the test basis is by specifying the requirements in machine readable and understandable manners.

In **the Risk-Based Testing** “all the testing activities including management, selection, prioritization and resources are based on corresponding risk types and risk levels”. One example of a ML approach is the Predictive Test Prioritization (PTP) technique which is used in selecting the most optimal prioritization technique for the application in test (Chen, et al., 2018).

The third approach, **Model-Based Testing** (MBT), is “based or involving models”. This has been the approach of all the AI testing solutions in this research. In the solutions the test cases are generated based on the AI generated models of the application.

Based on the ideas represented by Paul Gerrard and presented in the following chapters in this research, all the testing is involving models (Gerrard, 2017). This is true on individual tester level and on decisions individual testers make. The model should be shared between all the testers and model should be testable by other testers to call the testing project model based.

3.5.2 Test Techniques

The test technique is “a procedure used to derive and/or select test cases”. The test technique is selected in the design phase and the test cases are written or generated by following the technique.

Probably the most commonly used test, or test design techniques are Specification based techniques (Black-box techniques) and Structure based techniques (White-box techniques). The test technique selection in some cases could be automatically derived. For example from specification based techniques boundary value analysis should be quite easily trained for the AI if in general the test cases are derived from the specifications.

Karl Meinke et al. introduced a Learning Based Technique on 2012, which is an enhancement to the specification based technique by adding a machine learning training loop in the test execution for the SUT model training and automatic test generation based on the model (Meinke, et al., 2012). The technique is further described in the chapter 4.5.1

The other test techniques are Defect based techniques and Experience based techniques, from which the most used variation is Exploratory testing**.** The approach of AI testing can be considered as one form of exploratory testing based on agent learning to use the application in the same means as a manual tester would explore the application and find out executable elements and test what are the outcomes of the actions.

3.5.3 Software Test Automation

The test automation (TA) is usually referred as creating and executing a computer executable scripts out of predefined input actions in an application and their outcome verification steps (Mosley & Posey, 2002). The execution results are logged, and reports generated per the run. In this process the only automated steps are execution, logging, and report generation. Human testers input is needed in various phases of the TA process including the test goal definition, gaining the understanding on how to test the software, planning the specific test scenarios, writing or recording the TA scripts, manually running the tests which cannot be automated by the tool and analysing the execution results. (King, et al., 2019)

Using the AI in the software testing has biggest impact on the ways the traditional test automation is executed and maintained. In the traditional test automation the scripting techniques such as linear scripting, process-driven scripting, keyword- or action word driven testing etc. have been used (ISTQB, 2012). However, when thinking about the AI driven testing it depends on the model and approach of the solution how the tests are created.

Most probably the keyword or action-driven testing technique is the closest one to general AI testing. Keyword-driven testing technique is defined as “a scripting technique that uses data files to contain not only test data and expected results, but also keywords related to the application being tested. The keywords are interpreted by special supporting scripts that are called by the control script for the test.” (ISTQB, 2018)

The keywords are based on the high level meta language and describes the executed functions. Each word in this language represents a full or partial use case or function in the application under test. For example Login or SignOut present high-level business process and ClickButton represents a single low level interaction in the system.

Memón et al. have presented the Goal driven testing technique in 1999. The term is not recognized by ISTQB, but it has also been used by Jason Arbon and Tariq King in their presentations. Goal-driven testing is an approach to automatically generate test cases based on the given initial state and the goal state. The technique is used in AI driven testing in the test planning. The AI test agent a.k.a hierarchical planner produces the set of test cases to cover all the state transitions from the initial state to the goal. (Memon, et al., 1999) The test planner can also use other test techniques as part of its state transitions.

The technique when using the ISTQB classification is model-based keyword driven testing, where the test planner recognizes the use cases from the application models and the initial state and goal state are passed as keywords to the test planner.

3.5.4 Regression Testing

The regression testing is “testing of a previously tested component or system following modification to ensure that defects have not been introduced or have been uncovered in unchanged areas of the software, as a result of the changes made.” (ISTQB, 2018)

This is where the efficient TA is probably most crucial. Due to limitation of resources and complex software system the full testing of software is not feasible. There are different regression testing techniques to limit the number of test cases to be executed. The techniques can be categorized in prioritization, selection, and minimization approaches (Lachmann, 2018). One machine learning driven approach for the prioritization is called Predictive Test Prioritization (PTP), which is further discussed in the chapter 5.1.2. (Chen, et al., 2018).

3.6 “New Model for Software Testing” – The Model for Building the AI Testing

Paul Gerrard has been presenting an idea and demand for the “new model for the software testing” in different seminars and presentations, including in the FiSTB Testing Assembly on September 2019. His opinion is that the old ways of performing the testing does not work in the future due to the agility to setting the pressure to the testing,

digitalization bringing complexity on the systems and the testing responsibility is being redistributed to the whole development team.

In the high level the model is as described in the following picture. (Figure 3)

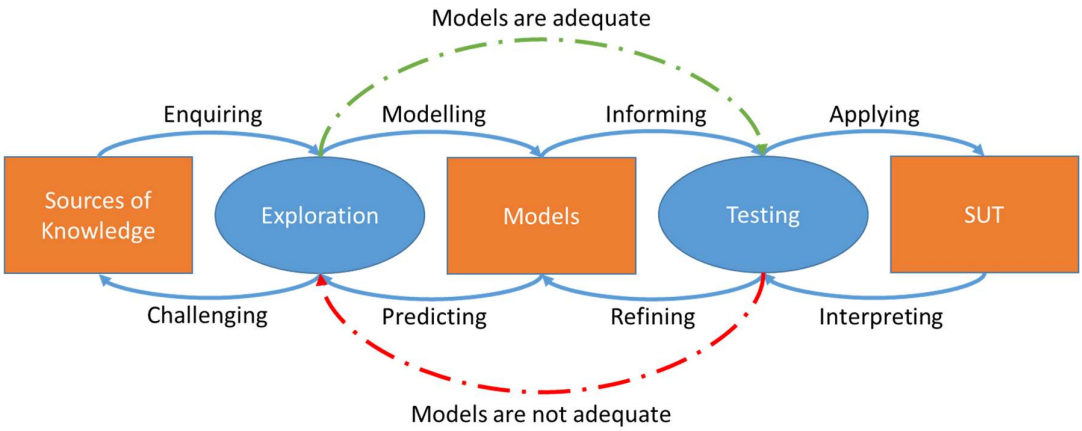

Figure 3 - New Model for Testing (Gerrard, 2017)

This model represents a workflow in the tester's optimal mind-set. It can be completely generalized to be independent from the logistics of the testing. It does not matter which tools or processes are used or followed. (Gerrard, 2017)

The model is completely in line with the testing processes and best practices in the testing. It represents brilliantly the way of how the tester should be thinking when executing the test process. Therefore, it suites perfectly as a reference model for the AI testing.

How this model of the testing fits in the AI testing is discussed more in the following chapters.

4 AI Testing – The AI Test System

The fundamentals of AI testing are based on the idea of “automatic abstraction of application and test logic” (Arbon, 2018). This can be achieved with intelligent learning agents, that can autonomously perceive and act in its environment. They can plan and create the test cases on the target system by exploring the functionality and learning how the application is used. In the end they can execute the tests and do evaluation on the test results. The agents are organized with the other agents and can act on different hierarchical levels. (King, 2019)

AI test frameworks can be used as generalized for testing cross-domain applications and testing multiple applications within the domain. Designing the AI test framework to work with the cross-domain applications and multiple applications in the domain as a general framework sets a requirement for the test “library of the common user flows”. In this library the elements and the actions are connected. When the AI agent observes an element in an application it can interact with, it can make a query to the database for the test cases for that specific type of element. For example, if an AI agent has observed a search box for a city, it can request for a test cases of that element and execute the tests on any application having the similar kind of search box. (Arbon, 2018)

Benefits of the generalized tests is that no tests are needed to be written twice and the test lab can be shared between different teams in the organization. This is supported by using different cloud solutions which can be easily scalable and accessible. (Arbon, 2018)

The architecture presented below in the following chapters is based on the Tariq Kings AI Testing “Masterclass: AI and Machine Learning Skills for the Testing World 11.04.2019” presentation (King, 2019) and the presentation slide sets prepared by the King and Arbon. It also has some author’s improvement ideas and comments based on the Paul Gerrards “New model for the testing” (Gerrard, 2017).

4.1 AI Agents

The recommended approach for the AI testing is to utilize one or more testing agents capable on exploration and testing the target application. The agents can be used in two purposes:

- Coordinator agent
- Test agent

The agents are based on executing the MAPE-K control loop originally presented by IBM in its white paper “An Architectural blueprint on autonomic computing”. MAPE comes from words Monitoring, Analyzing, Plan and Execute. All this is based on the Knowledge. Knowledge can be seeded from outside, acquired with machine learning, maintained (model updates) and sharing the knowledge. The functional environment of the AI test system is described in the following picture. (Figure 4)

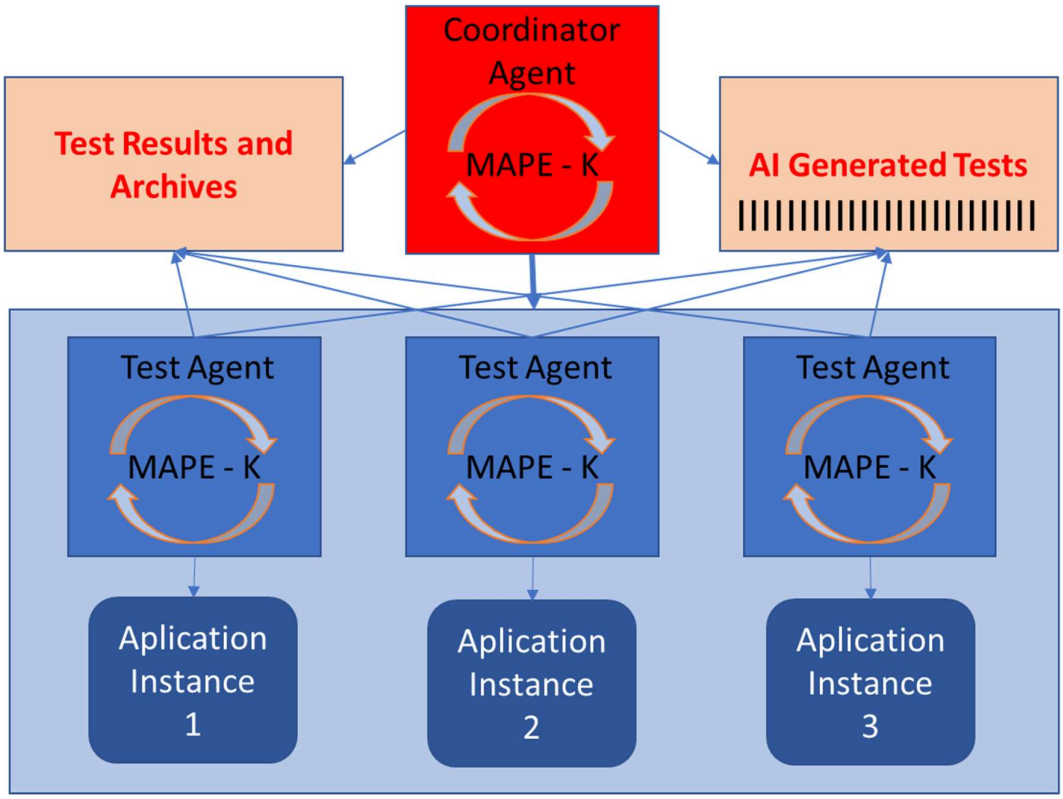

Figure 4 – Functional environment of AI test agents (King, 2019)

The test agents are autonomous. Ideally, they should act on a target application independently without interfering each other. This can be done by directing the agents to test the different features of the application or to utilize container technology for example docker for the target applications so that each agent works on its own SUT instance.

Cloud computing, virtual machines and virtual services help scaling the system. From the cloud it is easy and efficient to increase the capacity for the test executions, data generation, and provision the environments with using container technologies (e.g. docker).

In the analysis and design phase of the testing process the test agents explore the application to gain understanding of the system under test, environment and differences between the builds. They are responsible for building the model of the application by examining and exploring the target system. As part of the implementation phase the agents create the test cases in the test sets based on the models.

In the test execution phase, the agents execute the tests and save the outcome to the test results and history archives. They interact with the application under test by inputs and verifying the outcomes. At the same time, they learn from the outcomes of the interactions and continuously update the models of the applications.

The agents, also called as bots can be utilized for different testing activities and instructed with the testing attitude to concentrate on validating the outcome as positive testing or trying to find errors in the unexpected functionality by performing negative testing.

The coordinator agent act as a test manager for the test agents. It administers the test sets by removing duplicate tests, scheduling and distributing the tests to the test agents.

The process how to carry out the testing can be split in four different activities, exploring and perceiving, modelling, interacting and learning from the interactions.

4.2 Explore and Perceive

The initial purpose of exploring is to perceive the application. This can be done by classifying for example the application UI elements behaviour. This answers the question what can be interacted with in the application. This does not mean that the testing is restricted only on system level testing and on the user interfaces. It can be applied on other test levels for example unit testing or testing applications with no UI like services. The test level or input interface only effects on the built discovery engine. In addition, later after multiple iterations of the executions comparison can be done on the differences between the builds.

When perceiving the application, the agents identify the candidate for the control labels. The elements may be identified based on the generalized Classification. This can be done by combining the information from the document object model (DOM), computed render trees and combined with image recognition.

Even though Tariq King in his presentation concentrates mainly on the exploring the application the exploration process include also the other test basis. Paul Gerrard presents the exploration process in his model as following. (Figure 5).

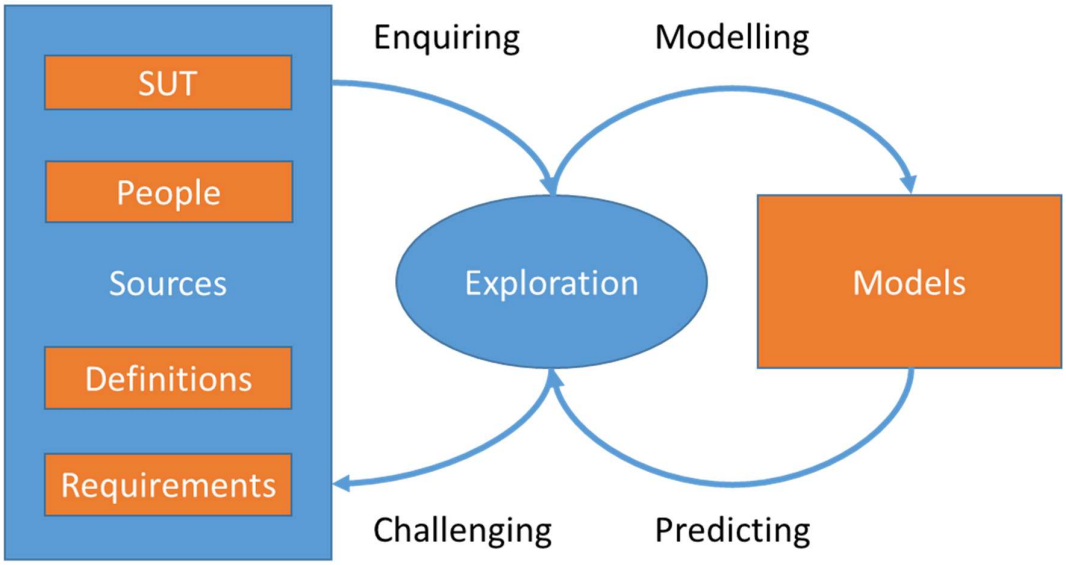

Figure 5 –New Model for Software Testing - Exploration Process (Gerrard, 2017)

In practice the human tester explores all the sources of information including the SUT. Application features can be discussed with the other people. The definitions can be red

and interpreted, specifications, user stories and the requirements can be explored. The challenge for the AI is how to explore the requirements and definitions, but there are tools presented for this as well.

4.3 Modelling

The approach for the AI driven testing implements model-based testing where the AI bots create a model of the target application. As presented in the picture (Figure 5) we as humans create the models of the application in our mind. These models can be presented as test plans and test cases, or testing can be executed in the means of exploratory testing methods if there is no wish to use time on writing the models down.

Anyhow the AI agents as well as human based test team applying model-based testing technique, require these models in the testing. Some example of the methods for presenting the models are state machine and Markov chain. The selected model should represent the target application or system and fit in the purpose of testing it.

Building models and the input space explosion can be handled with the probabilistic modelling. When the bots figure out what it can interact with, it builds a state model of the actions and their outcomes. When the observations of the items that can be interacted with increases the models are merged and grow all the time bigger. This leads to input space explosion. As an example of a web page where are 100 actions and use case of 30 steps. This creates 100^30 different paths to be modelled. To handle this input space explosion, King presents probabilistic modelling which is based on the idea that some elements and actions are more interesting than others.

4.4 Interaction

The actions are generated from the models for the agents to navigate through the application UI, input data and verify the outcomes of the actions. In the interactions for example goal-based testing can be used in which the test agents use the application model graphs and initial status and end goal instructed on high level test plans. The traditional test drivers like Selenium WebDriver can be used in the interactions.

In the Gerrard's model the interaction occurs in the testing phase as described in the following picture. (Figure 6)

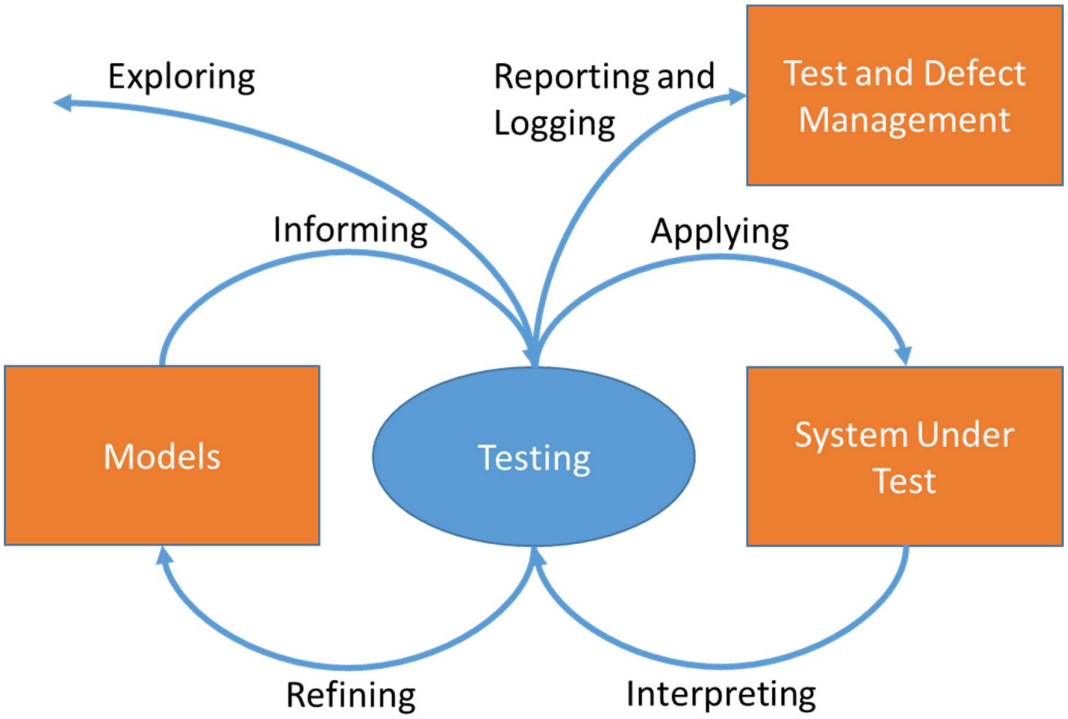

Figure 6 - New Model for Software Testing - Testing Process (Gerrard, 2017)

Something to be highlighted at this point is that even though there are some approaches taken on the test oracle problem; the result interpretation is still highly human driven activity. The interpretations can be done, and the models may be updated based on the results, but there must be some reliable mechanism for that. For example, being able to build the models based on the requirements and the specification or using some of the presented mechanisms to build the test oracle as presented in the chapter 5.3.6.

The model refining is a pure learning and exploring activity which will be discussed in the next section.

4.5 Learning

Based on the outcomes of the interactions the agents learn to improve the quality in the future interactions in the system. In the initial state the interactions can be silly, but as the number of iterations increase the agent learn on using the system. It should be

highlighted that even though the initial interactions are not leading to a desired outcome according to the set goal of the execution, the random interactions could lead to outcomes that would not have been tested in the manual human driven testing.

4.5.1 Learning Based Technique

One of the early adaptations of the AI testing was Learning Based Technique which was presented by Meinke et al. on 2012. (Figure 7)

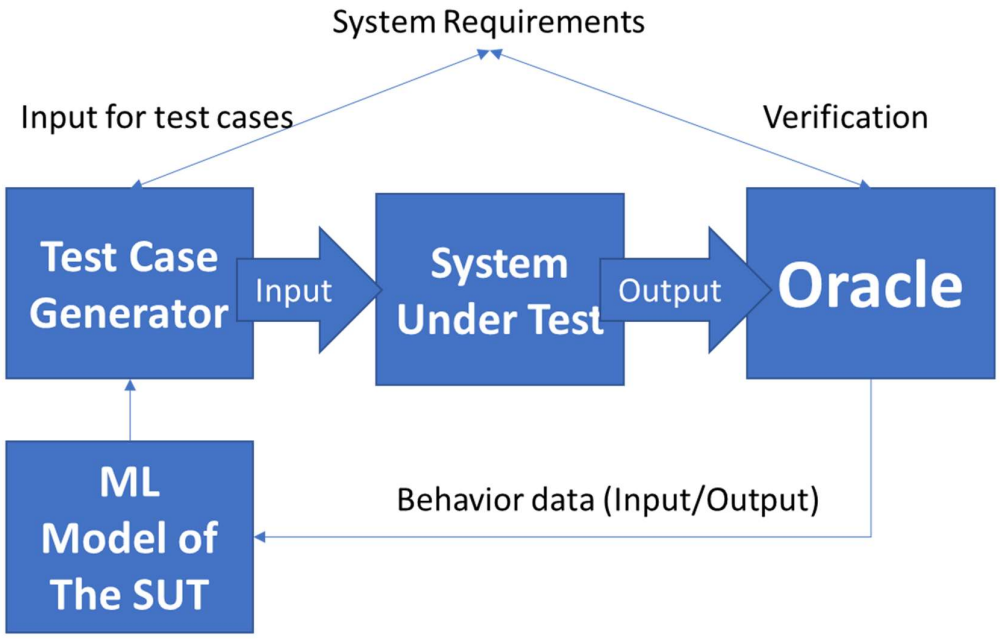

Figure 7 - Learning based test technique (Meinke, et al., 2012)

Learning based test technique is a specification-based technique which target is to improve the traditional specification-based techniques with machine learning enabled models and automatic creation of the test cases with the models. (Meinke, et al., 2012) It is advisable to use more than one learning technique to gain confidence in the training process.

From Paul Gerrard's model point of view the learning occurs at the exploration phase by enquiring the sources of knowledge. This includes perceiving the SUT by applying the models, interpreting the results and refining the models if the outcome of the test is unknown and the validation has been correct.

The challenge for the AI is still how to interpret the definitions and requirements into the model. One approach is to use structured manners in the feature specification that by using Natural Language Processing (NLP) techniques the AI could be able derive the models. One example technique could be using Behavior-Driven Development (BDD) in which the system behavior is represented in the format of Given-When-Then by using Gherkin language. The BDD is used in the agile development teams to enhance the communication between all the parties, developers, testers and business participants by sharing the common language.

This kind of machine learning approaches have been used with the goal-driven testing where the specification was given to the test agent in the Gherkin style-language in the format of initial state and goal state for the agent. In the interaction the agent uses the application model graphs and initial status and end goal instructed on high level test plans.

Also, the test flow designer in the chapter 5.2.1 is using the similar approach.

The machine learning approaches in the learning context are:

- Deep Learning and Artificial Neural Networks (ANN)
- Decision Tree Learning
- Reinforcement learning
- Bayesian Networks

Deep learning is a subfield of machine learning. What makes deep learning deep comes from the complexity of mathematical models. Artificial Neural Networks (ANN) are simulated neural networks in a computer modelled from the biological neural networks in our brains. Neural networks consist of neurons that can process information independently. One neuron separated from other neurons is simple and cannot do much but connected to other neurons they can form a really powerful system. Using these artificial neural networks has come reasonable after the gaming industry and development of more and more powerful graphical processing units (GPU) that are built for processing small tasks parallel and which suits perfectly for neural processing.

For classifying the UI elements with decision tree learning the steps are:

- Data preparation in which the data is gathered and pre-processed

- Feature engineering in which the attributes that will be modelled are specified
- Model training in which decision trees and random forests is used

Decision tree learning is a supervised learning tool for two problem types, classification and regression. Tree like shaped model is visual and simple to understand. The tree consists of one decision node, one or more change nodes and multiple outcome nodes.

With random forests technique the efficiency of decision trees can be improved by minimizing the risk of overfitting. This technique is based on building multiple trees from randomly selected parts on the training set. (Ho, 1995)

For exploring and interacting with the application the reinforcement learning, or Q-learning and goal-driven testing technique is a preferred approach. Reinforcement learning is a machine learning technique in which the algorithms or agents train the model by interacting with the environment and receiving rewards from the actions performing correctly and penalties resulting incorrectly. It can be compared to a child learning new tasks with trial and error. (Sutton & Barto, 2018) After multiple training rounds the machine will know how to reach the target without a person explicitly telling how to do it.

Q-learning is a model free method of reinforcement learning. In this AI can define its optimal action based on its previous interactions without creating a model beforehand. The action is taken based on the Q-function which is used to predict the best action for a state to maximize the cumulative reward in the end. This Q function is updated iteratively with Bellman Equation. This works for simple policies, as the system counts the easiest way of earning the rewards. Therefore, there should be a random exploration ability for program to take some random action instead of optimal action. (Raval, 2017) Q-learning method has been used for example automating different game agents.

Based on the rewards and penalties the agent gain from the interactions, the actions can be directed to wanted outcome. The agents can also be directed to learn different kind of attitude towards the testing for example verifying the expected functionality, positive testing, or trying to break the system, negative testing.

The benefits of using the Q-learning technique include the following aspects. For its adaptiveness it suites general purposes meaning the same framework can be used for

different testing types and for any application or domain. The tests are reusable and pesticide free since new tests are generated each time.

Bayesian networks are based on the Bayes' theory of conditional probability. This presents the probability of an event happening, given that it has some relation to one or more other events.

Daniel Rodriquez, Javier Dolado and Javier Tuya have researched the topic of Bayesian Concepts in Software Testing in 2015. They used systematic literature review protocol on the main digital libraries to search and select 40 different references applying Bayesian approaches in the field of software testing.

Their initial review on the subject has addressed example solutions on six different areas of software testing having influence on all the phases in the testing process. Anyhow most of the researches were related in the test execution or predicting the execution results as 60% of the publications were related to fault and defect prediction methods. About 10% of the references were related for both software testing effort estimates and test generation. They have been used as probabilistic graphical models for the prediction and graphical models for optimization and data generation. Different approaches were used like Bayesian Networks, Markov models with its extensions Dynamic BNs and Influence Diagrams, Naïve Bayes and Tree augmented Naïve Bayes (TAN). (Rodriguez, et al., 2015) Naïve Bayes classifier is probably most commonly used and most powerful implementation of Bayes´ Theorem in the machine learning (Mahendru, 2019).

4.6 Risks of the AI Testing

Even without influence of the AI use in the system, the complexity of the solution makes it highly error prone. The system should be designed for testability and all the challenges described in the chapter 3.4 should be considered. There needs to be a mechanism to manage the complexity. (King, 2019)

In the following sections the risks of utilizing the traditional test automation in the testing projects presented by the ISTQB (ISTQB, 2018, p. 81) have been reflected to the AI Testing.

“Expectations for the tool may be unrealistic (including functionality and ease of use)”. This is a risk for the AI Testing especially at the time when there is a lot of AI hype in the air. There might be tools offering AI features which might not suite in their purpose or at least in the way that is desired.

“The time, cost and effort for the initial introduction of a tool may be under-estimated (including training and external expertise)”. The external expertise to current testing or agile development teams is needed in forms of AI QA Strategists, Data Scientists and AI Test Experts to initially enable the AI Testing development.

“The time and effort needed to achieve significant and continuing benefits from the tool may be under-estimated (including the need for changes in the test process and continuous improvement in the way the tool is used).” The authors opinion based also the survey answers of the TA specialists is that there is understanding in the risks related to TA and somewhat sceptical thinking against the implementing the AI testing and efficiency of the AI testing. There most probably are benefits in using the AI Testing, but the implementation is not easy in any means. As the analysis of the World Quality Report highlights, the new skills are needed in implementing the AI testing.

“The effort required to maintain the test assets generated by the tool may be under-estimated.” The AI testing and its currently available commercial tools are targeting to minimize this risk from the traditional TA side.

“The tool may be relied on too much (seen as a replacement for test design or execution, or the use of automated testing where manual testing would be better).”
To mitigate this risk the testing the AI requires critical thinking. The architecture needs to be planned with care, with maintainability and testability.

“Relationships and interoperability issues between critical tools may be neglected, such as requirements management tools, configuration management tools, defect management tools and tools from multiple vendors.” All these factors should be thought when starting to implement the AI test framework. How the AI testing could benefit from the data all these tools are providing and what are the means to let the AI understand the available data.

“The tool vendor may go out of business, retire the tool, or sell the tool to a different vendor. The vendor may provide a poor response for support, upgrades, and defect fixes.” These are risks related to the commercial solutions of the tools and should be thought when making the decisions on buying them. To mitigate this risk, an approach would be to build a framework basing to the open source tools, having solid development and user base. Most of the general tools used in the machine learning development are open source tools, so the risk of “suspending an open source project” should not be high. Also, there is lots of support available from the machine learning community.

As a general note for all the previous risks, implementing the AI Testing tools with the Agile or DevOps methodologies as a side product of the actual product development would be a good approach mitigating the risks. The costs can be kept low and new approaches in the AI testing development can be taken rapidly.

5 Machine Learning Approaches Within the Test Process

The following chapters describe the fundamental testing process and what kind of AI driven approaches there are to support the individual phases of the testing process. The fundamental testing process was selected for the reader to understand the phasis in the testing and help categorizing the different ML approaches. It also structures the document section in hierargical manner. The process descriptions are defined by the ISTQB (ISTQB, 2018).

The ISTQB defines the testing process as “the set of interrelated activities comprising of test planning, test monitoring and control, test analysis, test design, test implementation, test execution, and test completion.”

The following picture and chapters describe the phases in the process (Figure 8)

Figure 8 – Test process

5.1 Planning, Monitoring and Control

The test planning phase identifies and plans on high level all activities and resources that are needed to meet the testing objectives based on the selected test strategy. Monitoring enables the team to follow the testing activities and test efficiency and control acts with changes if needed.

The approaches for the test planning are learning from the production use, perform test case prioritization using predictive prioritization technique and evaluating the test suite quality by using predictive mutation testing.

5.1.1 Learning from the Production Use

It is possible to instrument the production and test environments in detailed level. This enable learning from the actual real user behavior and failures in the system. The logs are not only generated from the error conditions, but the activities can be logged, and models crated even from individual the user sessions. There is enough capacity and tools available to process and analyze that kind of data. An example for this kind of solution is Splunk (Splunk, 2019). The production statistics and analysing the real user behaviour is also considered important area for future research according to the Mabl product engineers (Tatar, 2019).

The analysis on the product use and root cause analysis for the errors could be used to drive the test planning and prioritization, but also as instructions on automated test case creation. This does not have to be limited to functional testing but also non-functional attributes like performance, security, accessibility and so on. (Crispin, 2019)

There also are studies using Bayesian networks on the system and product quality monitoring, defect predictions and identifying software faulty modules (Rodriguez, et al., 2015).

5.1.2 Test Case Prioritization

The purpose of the test case prioritization is to focus the testing capacity and effort in the activities that are most important. One benefit of the prioritization over test case selection or minimization approaches is that the prioritization allows continuous testing until all the test cases are executed and there is no need of leaving anything untested. (Lachmann, 2018) This approach suites also well for multiple concurrent test agents to be executed.

The existing prioritization techniques can be categorized in the groups of cost-unaware techniques, cost-aware techniques and baseline technique. Based on the study of Chen et al. there is never one optimal prioritization technique that can outperform for every application. The team built Predictive Test Prioritization (PTP) method can predict by using the machine learning the optimal test prioritization method from the selection of the prioritization techniques for the specific application. This outperforms other randomly selected prioritization methods significantly. The method is based on building the

predictive model by collecting distribution of test coverage, testing time and coverage per unit time. (Chen, et al., 2018)

Another example of using AI in test case prioritization is using reinforcement learning techniques. This was used by Spieker, et al. in their research *Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration* (Spieker, et al., 2017)

5.1.3 Predictive Mutation Testing

Mutation testing is said to be an effective method for testing the test suite quality. In the mutation testing the SUT is altered with the “mutants” and test suite executed while monitoring the number of killed mutants with the tests. Anyhow the mutation testing is computationally expensive as each mutant is executed independently against the whole test suite. The research team has developed a Predictive Mutation Testing model, which predicts the mutation test results without actual mutant execution. The results of the research were promising. (Zhang, et al., 2016)

5.1.4 Quality Models

Bayesian approaches have been used for building the quality models like activity-based quality model. They have been used to monitor the system reliability and for estimating the software testing effort and productivity and test plan generation. (Rodriguez, et al., 2015)

5.2 Analysis and Design

Test analysis is “the activity that identifies test conditions by analyzing the test basis.” The aim of the analysis is to gain understanding on the types of tests that are needed to accomplish the objectives of the testing. In the presented AI testing approaches the exploration of the target application or system for example classifying the UI elements is one approach to analyze the test basis.

The test design is “the activity of deriving and specifying test cases from test conditions.” In the test design phase, the test case design techniques are determined that the test

coverage can be achieved. In the end the test cases are created to meet the identified test conditions. The test cases can be designed in low/concreate level or high/logical level. Each have their own benefits and are depending on the level of the specified design to the application or the feature.

5.2.1 Test Flow Design

Declarative testing can be used to design and implement the test execution flow. It is a method for the AI testing agent to specify the intent of a test and how to execute the test. The test is described in natural or domain-specific language. (King, et al., 2019) There is a community driven example of “Abstract Intent Test syntax” available (Arbon, et al., 2018) on which anyone can comment or contribute.

Another approach for designing the test execution flows is to use Natural Language Processing (NLP) with Long Short-Term Memory Recurrent Neural Networks (RNN). This approach has been used by the Ultimate Software with its AGENT –prototype. The test cases are generated in co-operation with the human testers based on the human input from abstract test flows. (King, 2019)

The agent predicts the sequences of the next steps what a manual tester would take and execute the tests. For example, in case of testing the form including a text field named as “First Name”. If the human defined input for the network is “Observe TextBox FirstName” the predicted next word is “Try” for acting with the text field. The second predicted word could be “VALID” or “INVALID” defining the input the system feeds for the action and what the test focus is. After the action taken, there would be predicted “Observe ErrorMessage” or NotObserve ErrorMessage” to validate the test result. (King, 2019) (Santiago, et al., 2018)

5.3 Implementation and Execution

The test implementation “prepares the testware needed for test execution based on test analysis and design.” This include creating the automated tests, organizing the tests in the order of execution, finalizing the test data and environments, forming execution schedule and do all the preparations to enable the test execution to begin.

In the implementation side the auto-healing and self-maintaining tests is probably one of the most used form of using AI in the testing. Based on the research of the tool features from the marketing material almost all the commercial tools using AI has some level of automatic correction and test script maintenance support based on the UI changes.

The test execution is “the process of running a test on the component or system under test, producing actual result(s).” The tests are executed according the plan determined in the implementation phase. Also, additional tests can be executed based on the findings done during the execution and that can be executed without additional preparations or implementations. The test execution ends in the test oracle problem which is as an issue of identifying the test output correctness.

5.3.1 Service Virtualization

When developing applications using service-oriented architectures the developers and testers may face challenges with the services like with unfinished and malfunctioning responses, capacity limitations and service unavailability, access issues, and restrictions on using different kind of configurations with the tested applications.

The services can be categorized as stateless and stateful services. Example of stateless service can be like weather broadcast service that returns a forecast for a specific city whereas a shopping cart service requires to know the content added to the cart in order to serve with the checkout functions, making it as a stateful service. The stateful services are more difficult to simulate and machine learning can be a good asset in it.

Eniser and Sen have researched stateful service virtualization that can be used to create a virtual duplicate of the dependent service for the development and testing purposes. (Enişer & Sen, 2018)

The process of creating virtual services with the assistance of machine learning the steps are capturing, modelling and simulating. The requests and responses between actual service is recorded (when the service is available and trustworthy) or written manually in the interaction repository for the machine learning algorithms to model the service. This virtual service is deployed in the required environments and return synthetic responses for the given requests instead of using real service.

In the evaluation they created two approaches to create the models which performance in terms of accuracy and training time were compared. Classification Based Virtualization (CBV) using Repeated Incremental Pruning to Produce Error Reduction (RIPPER) algorithm was recommended based on its training time compare to the Sequence-to-sequence Based Virtualization (SSBV) which was based on Recurrent Neural Networks, to be more specific Long Short-Term Networks. Anyhow the accuracy of the SSBV was better than with CBV, so if there is time and GPU units available to train the model, SSBV would be more appropriate method to select.

5.3.2 Test Data Creation

Mable has some plans on implementing machine learning algorithm on test data creation. Based on Lisa Crispin´s blog post on the Mable web site, one considered potential use case for Mable is to train an AI to identify a comprehensive sample of the production data, mask it and create a set of test data for the testing purposes. This is to tackle the issue of not being able on using the actual production data in the tests due to the complexity of the possible data variations. (Crispin, 2019) Also Bayesian approaches have been used in the test data generation amongst other implementations (Rodriguez, et al., 2015).

5.3.3 Differential Testing

In the differential testing the application version differences are classified and the models are updated with the learning from the feedback. (King, et al., 2019) This is not necessarily only considering functionality, but also the efficiency of actions taken or use case duration can be measured and compared between the builds.

5.3.4 Visual Testing

The visual testing uses “image-based learning and screen comparisons to test the look and feel of an application” (King, et al., 2019) For example Mabl creates a visual model of the pages visited when the user journeys are recorded. It uses at minimum ten screenshots per page depending on how visually dynamic the pages are. By comparing the screenshots mabl labels the elements that changes frequently as dynamic regions. For other regions the tool notifies on visual changes highlighting the area on the

screenshot. (Tatar, 2019) This kind of testing can be used on functionality and usability testing.

5.3.5 Bayesian Approaches for the Test Implementation and Execution

There are multiple areas in which the Bayesian approaches have been used in the test implementation and execution. In the implementation these have been used for test generation and test data generation, test case selection by prioritization of the test cases and optimizing the regression testing (Rodriguez, et al., 2015).

In the test execution the approcaches include classification of test automation failures in the graphical user interface testing (Martinez, et al., 2014), fault and defect prediction as defect prediction from static measures (Rodriguez, et al., 2015), identifying software faulty modules (Rodriguez, et al., 2015) and predicting test flakiness (King, 2019)

5.3.6 Test Oracle

Probably the biggest challenge for the AI testing is the test oracle problem. Currently testing AI based systems requires critical thinking from the human testers and approaches are limited mainly in human-oracle based validation. Anyhow there are some researches done for tackling this issue.

Braga et al. have studied the automation of the test oracle mechanism for the software. Their approach is based on historical data captured from the SUT by specific collection data collection component. The test oracle is trained with the data gathered by a specific knowledge discovery mechanism from the full history data. The team proposes a method of using a combination of usage logs and two machine learning techniques: AdaBoostM1 and IREPalgorithms. Other machine learning based methods have also been researched. For example Support Vector Machine, using a single ANN, multiple ANN´s, a k-nearest oracle-based dynamic ensemble selection method. (Braga, et al., 2018)

Marc Roper has created a test strategy that uses combined unsupervised learning method (clustering) and semi-supervised learning techniques to identify the failing test cases. In the strategy first phase the test cases are clustered in the initial groups. In the smallest clusters are the test cases which outcomes contain biggest proportion of the

failures. These tests are checked first manually as they are more likely containing errors. In the second phase the rest of the test cases are classified as failing or passing by using semi supervised learning technique. (Roper, 2019)

Dwarakanath et al. have done a research on metamorphic testing. The traditional testing validates that the input is processed in a system under test to an expected output. These may be called as input and output pairs. Input 1 is processed as output 1 and input 2 as output 2. The inputs can be selected in a way that there should be a relation between inputs and outputs. This relation between outputs is called as Metamorphic Relation. If this relation is not maintained, it can expect that there is a defect in the implementation. In this research Dwarakanath et al. gained on average 71% error catch rate for the bugs seeded in the system. (Dwarakanath, et al., 2018)

6 AI Tools for Software Testing

The following chapters present the traditional test tool classification and how the artificial intelligence approaches affect in the tools and their use. Also some currently existing testing tools are presented that use machine learning in their implementations. The classification of the tools is based on ISTQB syllabi. (ISTQB, 2018)

6.1 Tools Supporting the Management of Testing and Testware

The classification of the test management tools is challenging, as many test management tools may cover all the phases in the testing process or may have close integration to other tools covering other areas of the softwre testing. Test management tools can be used in the requirements and specifications management, designing the test cases and linking the test suites, scenarios, cases in the requirements, controlling the test executions and also in the defect management. Also how the AI features and AI testing will be managed in the future is unknown area. This is depending on the level of AI used.

In general the ISTQB classifies the tools in the following sub categories:

- Test management tools and application lifecycle management tools
- Requirements management tools
- Defect management tools
- Configuration management tools
- Continuous integration tools

In the future the AI most probably will give suggestions on the requirements for the system for example based on the application usage behaviour and efficiency of the usage on the production. Also, the business processes may be maintained by the suggestions made by the AI that would lead to the requirements management process. For example, in the research of Meinke and Bennaceur machine learning has been used in Specification extraction (Meinke & Bennaceur, 2018).

The defect management tools are used manually monitor and control the defects and their fixes. If in the future linking the AI testing and AI development for example in the

means auto-correcting the code these may be part of the self-testing and self-fixing systems.

6.2 Tools Supporting the Static Testing

AI has taken hold also on the static code analysis tools. On August 2019 a Swiss start-up company DeepCode, using machine learning for automating coder reviews, got four million US dollars from the investors. The AI bot is free for use for the small companies less than 30 developers, for open source projects and educational purposes. (Lomas, 2019)

6.3 Tools Supporting Test Design, Implementation, Execution and Logging

“Test design tools aid in the creation of maintainable work products in test design and implementation, including test cases, test procedures and test data.”

- Test design tools
- Model-Based testing tools
- Test data preparation tools
- Acceptance test driven development (ATDD) and behavior driven development (BDD) tools
- Test driven development (TDD) tools

Test execution and logging tools include following:

- Test execution tools
- Coverage tools (e.g., requirements coverage, code coverage)
- Test harnesses
- Unit test framework tools

Many of the AI driven test design and implementation tools also support test execution and logging so therefore this chapter combines these two tool types in the testing process.

6.3.1 Commercial Tools

The following chapters present tool examples implementing AI and Machine Leargning features in their automated testing solutions. Some of the tools use record and replay method where the ML is used in automatically self-healing the test cases when the system under test changes. Some of the tools use ML in automated exploring the system and automated test creation. Also visual validation on the changes is a common approach.

All the following descriptions are based on the marketing material of the companies and therefore are not scientifically reliable. Anyhow some small scale evaluation on the features was performed with the Mabl, Applitools, and SOFY solutions.

Mabl is a record and replay TA tool with auto-healing features enabled by machine learning algorithms. Machine learning enables evolving the test cases as the application evolves giving consistent and accurate feedback about the application without minimal manual update on the test cases.

Based on the marketing material of the company the tool is designed to support static testing as a peer review support tool, support system testing with end to end testing features and exploratory testing functions. Quality intelligence tool enable the monitoring to identify trends, patterns and anomalies in the application behavior based on the data received from the continuous testing and actual production use of the end users.

Machine learning is used by creating model of the application with the recording the functionality with the user stories or journeys. The model is constantly updated with the test runs. The results of the test runs are compared with the learned model of the application. (Hristov, 2019)

Mabl is hosted on Google Cloud and machine learning tools have been selected from the Google solution portfolio; Cloud ML Engine, Dataflow and Lighthouse. (Tatar, 2019)

Functionize uses machine learning in test case creation. It can create test cases autonomously from written English language instructions by using NLP technology called Adaptive Language Processing. It can also automatically create new test cases based on actions performed by live users.

Functionize has self-healing features that minimize manual changes on the test cases affected by the code and application changes and giving so called smart suggestions on changes to be added to the test cases. (Functionize, 2019)

With a quick glance at the product web pages the features seem to be quite similar to what Mabl has got to offer.

Applitools is a visual confirmation tool that can be integrated with many existing test automaton frameworks and programming languages. It also supports multiple browsers, screen resolutions and mobile application on both portrait and landscape modes. (Applitools, 2019)

Applitools uses AI and machine learning in classifying the UI elements in the screen captures and comparing the elements in the set baseline recording or screen shot. Dynamic content can be excluded from the comparison so frequently changing content does not cause false positive alarms in the execution results. The tests are easier to write as no validation steps are needed in the UI and the defects are found based on any change in the visual appearance. Also test maintenance is easier as no locators or elements are used that would need regular changes when the applications change even without visual change. The new user interface changes are approved in the screen captures by marking the change as a new baseline. Defects can be managed in the same screen captures by highlighting the faulty functionality. These defects can be integrated to version control or task management systems.

Eggplant is an automatic test generation and test execution optimization tool. The key elements of the Eggplant are the model of the SUT, Test Case Creation, Coverage analysis and Bug Hunting. The model, called in the eggplant as the Identical Twin is a state and action model of the application. Based on the model and the previous test executions the ML algorithms design the test cases. In this model the test coverage can be visully presented based on the number of executed tests in the each module or application in the system. The engine designs new tests for the modules not touched by the tests before. The bug hunting feature when identifying a defect in the application the actions taken are saved and the system would automatically try to find different methods of triggering the same issue to provide additional information for the development. The means in bug hunting for example use the different state transitions if available in the model or using different data. (Eggplant, 2019)

TestRigor is an AI powered exploratory testing tool with features as automatical page object detection, automatical recognition and attempt to understand how the system under testing works and automatical exploratory test creation based on the gained understanding. No recording is required. It does test prioritization based on the understanding of the most important features on the application. (Testrigor, 2019)

With **Testim** user can record or code automated test cases. The tool has auto correction features that reduce the amount of manual work needed when the target application changes. The tool does visual validation on the pages tested and user can use different data sets, conditions and browsers and so on. The AI features concentrate mostly on ease of use of the recorded steps and that the steps can be reused between different test cases. (Testim.io, 2019) This is enabled by the AI locator model in which hundreds of attributes of the page or application elements are saved and few changes in the attributes don´t necessarily break the test. This is called as dynamic location strategy. As the number of tests executed increases more reliable results are while changes to application are learnt (Subramanian, 2018).

Apptest is an AI Bot powered exploratory testing solution for mobile applications. (Apptest.ai, 2019)

Sofy.ai features include exploratory testing engine that can browse through every webpage and screen on the application. It can automatically suggest and create new test screnarios based on the changes in the application. (Sofy, 2019)

TestCraft is a visual TA tool to create selenium-based tests by using drag and drop interface. It uses AI to automatically create a dynamic test model that is updated whenever there are changes in the AUT and therefore reduces time on test maintenance. The tests can be executed parallel in different environments and against different browsers. (TestCraft, 2019)

Diffblue Cover is an AI powered testing tool for java applications featuring automatical creation of unit tests. (Diffblue, 2019)

The company **Test.ai** is marketing itself as the world's first company to build AI-powered TA platform. The platform is told to scale to test thousands of applications without need

of programming and maintaining the tests. (test.ai, 2019) The company CEO and Co-Founder Jason Arbon has founded the AISTA Artificial Intelligence for Software Testing Association together with Tariq King. Jason Arbon has presented the ideas of the AI Testing and the test.ai solution in speeches in different conferences that have been refercenced in this reseach. The company and its products might become noticeable in the future even though at the moment company does not sell anything based on the information on its web sites.

6.3.2 Open Source Tools

AGENT is an machine learning-based multi-agent tool for testing web applications. It is created by Ultimate Software as a prototype of an AI testing solution. The AGENT uses training data from AGENT-X which is a label classifier extension tool for Chrome to help classifying the web elements. (Santiago, et al., 2018)

The AGENT utilizes multiple intelligent agents that explore the AUT and create concrete test flows based on the Long Short-Term memory (TSTM) generated abstract flows trained by human testers. In the test generation the agents predict the next sequence of actions that a human tester might take. Executed tests detect if the AUT operates in the way the of learned test flows behaviors. The testing process is coordinated by the coordination agent which from the message queue dispatches the testing assignments. This supports using multiple concurrent test agents and parallel execution of testing, exploring and test generation functions. (AGENT, 2019)

The AGENT technology stack includes:

- Python – Programming language
    - Pandas – Data analysis library for Python
    - NumPy – Library for scientific computing with Python
    - Scikit Learn – Library for data mining and data analysis with Python
    - Keras – The Python Deep Learning library
    - TensorFlow – Platform for machine learning
    - Lark – Natural language processing
- Celery – Distributed Task Queue / Task Manager
    - Flower – Visualizes the tasks running
- RabbitMQ – Open source message broker

- JSON – lightweight data-interchange format
- Selenium WebDriver– Browser automation tool

These run as micro services under docker. Default configuration of AGENT executes one coordinator agent and one test agent, but it can be configured for multiple test agents. (King, 2019)

Repository:
https://github.com/UltimateSoftware/AGENT
https://github.com/UltimateSoftware/AGENT-X

Appium is an open source TA framework for mobile applications. It has integrated web element locator classifier using Test.ai open sourced algorithms.This easies the test writing and makes tests less brittle. It is also possible for the testing community to add new features to the classifier by improving the training process by adding more data, using different training methods, or adding new labels for the elements. The existing training data can be found in Kaggle. (King, et al., 2019)

Repository for the element classifier:
https://github.com/testdotai/appium-classifier-plugin

DrQA is a Q&A system fetching the answers from the documents in the Wikipedia. The repository includes code, data and pre-trained models for the Wikipedia queries. The tool is not specifically designed for the testing, but as it is not dependent on the Wikipedia graph structure the tool can be applied to use any set of documents. Therefore, DrQA could be considered as a one approach for creating test oracle for the AI test system by creating the models from the specifications and requirements. (King, et al., 2019) (DrQA, 2019)

Repository:
https://github.com/facebookresearch/DrQA

AppFlow is a cross domain test tool for Android applications. It utilizes machine learning in screen and UI element classification. (Hu, et al., 2018)

Repository:

http://github.com/columbia/appflow
http://github.com/columbia/appflow-dataset.

SikuliX is a tool that uses OpenCV image recognition libraries to identify the elements on the desktop computer screen. It cannot be described as AI testing tool, as all the testing is directed by human. (RaiMan, 2019)

6.4 Tools Supporting Performance Measurement and Dynamic Analysis

This categorization include the following type of tools even though quite often specially commercial performance testing tools are including or integrated with the monitoring and dynamic analysis tools:

- Performance testing tools
- Monitoring tools
- Dynamic analysis tools

The currently available testing tools using AI measure some performance characteristics. For example Mabl measures the test run time, creates a model of the execution time updating the model with every execution, and as the test result notifies if the execution time has increased or decreased compared to the models prediction range. (Tatar, 2019)

The AI test system described in chapter 4 configured executing multiple agents parallel in the same environment the effect can also be considered as load testing. This anyhow cannot be said to be systematic testing without specific planning and control of the coordinator agent based on non-functional requirements that the test execution can follow. In more advanced methods the requirements could be automatically adjusted with the statistics based on the production use. The AI test system could be instructed to monitor the system resources during the testing and set the use cases and number of concurrent transactions based on written requirements. The same testing tools could be used both in the functional testing and performance testing and this makes using the traditional performance testing tools somewhat obsolete. However, it should be noticed that running for example multiple browser processes simultaneously generates heavy

load on the load generator and therefore all the functional testing approaches cannot be used in the load or stress testing.

One example of the monitoring and dynamic analysis tools utilizing AI and ML is splunk which uses ML algorithms in anomaly detection, predictive analytics and clustering. (Splunk, 2019)

The AI solution in the performance testing could include aspects in the research by Luo et al. The team has created a tool FOREPOST, which utilizes Feedback-Driven Learning methods to identify the performance problems. In the traditional performance testing the input parameters often are fed in a random order for the test execution tool to produce the requests for the target application to be processed. This can be due to lack of resources in analyzing the data and estimating the effect on the target system. In this approach the performance testing cycles produce data of the computational load generated by the single input in the dataset. Based on this data the tool selects based on ML algorithms, the inputs in the datasets for the following execution cycles. In this dataset are inputs which impact on the system performance is the highest. The bottlenecks in the system can be more easily identified as the noice from the inputs having smaller capacity impact is eliminated. (Luo, et al., 2016)

The FOREPOST tool enhances the commonly used open source tool Apache Jmeter with AUT profiler, Weka ML tools and its JRip class to implement propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER). With its current setup the system works only on Java applications, due to the selected profiling tool TPTP. (Luo, et al., 2016)

The similar kind work has been done by J. Wildstrom et al., but in their research the machine learning techniques were used in learning more efficient hardware configurations for serving the same capacity of inputs. (Luo, et al., 2016)

The tool is available here:
http://www.cs.wm.edu/semeru/data/ICSE16-FOREPOST/

6.5 Tools Supporting Other Specialized Testing Needs

In this categorizaation the ISTQB lists the following tools:

- Data quality assessment
- Data conversion and migration
- Usability testing
- Accessibility testing
- Localization testing
- Security testing
- Portability testing

According to Crisping there have been experiments on how the machine learning, supervised learning, image recognition and natural language processing were used to improve the accessibility testing effectiveness (Crispin, 2019). This can be considered one type of usability testing.

The visual testing approaches could be used for example testing the localization aspects, whis was one of the approaches for this research in its initial planning.

6.5.1 Security Testing

There are some books explaining how to use machine learning for penetration testing, as well as security testing tools using machine learning are available. One example for the book is Mastering Machine Learning for Penetration Testing: Develop an extensive skill set to break self-learning systems using Python, written by Chiheb Chebbi. Also, there are some penetration testing tools available for example Pentoma by SEWorks and Wallarm Security Testing by Wallarm. (Dsouza, 2018)

As an example, Pentoma according to the tool developers can scan any web-based application running on most popular cloud servers and identifies security risks like:

- SQL Injection
- Local File Inclusion
- Remote File Inclusion
- Unvalidated Redirects and Forwards
- XSS
- Unknown Security Risks

(SEWORKS Team, 2018)

6.6 General Purpose AI Tools and Frameworks

According to the research made by the company Figure Eight the most commonly used machine learning frameworks by the data scientists are:

- Pandas
- Numpy
- Scikit-learn
- Matplotlib
- TensorFlow
- Keras
- Seaborn
- Pytorch & Torch
- AWS Deep Learning AMI
- Google Cloud ML Engine
- Theano
- Microsoft Azure Machine Learning
- IBM Watson Machine Learning
- Amazon SageMaker

This research shows the community supporting the open source tools over the commercial or so-called White Label solutions. (Figure Eight, 2018) Most of these open source tools have been used for example in the AGENT prototype development.

7 Discussions and Conclusions

The objectives for the research have been met. There is a bunch of example approaches using machine learning presented from the different aspects of the software testing. These researches answer the question how AI implementations could serve the testing. In the beginning of starting this research it was unclear what would the findings be and what kind of document would this master's thesis be in the end. The scope and data in the document prove that the field of AI testing has grown rapidly during the last few years and there actually was much more information available that was assumed in the beginning. Also, the structure of the document is arranged to support both the standardized software testing and evolving AI testing linking the terminology of the processes, tools and methods together. The current understanding of the software testing is reflected to the recent research field of the AI testing and different approaches that have been taken to solve the testing problems with the support of machine learning.

7.1 Reliability and Validity

In the introduction there were a bunch of research questions that have been thought in this research. Many of them have been answered in some level. Unfortunately, the research was not able to give any evidence on applying the solution suggestions in practice. It would have been interesting to demonstrate for example the AGENT prototype and enhance it with some additional approach presented in the research chapter 5.

This research has been written by the test engineer/manager having non-working experience on machine learning and AI in practical testing activities. Like it is stated in the current state analysis, the testing and application development teams require specific skills in applying the machine learning methods in the AI development and testing. Unfortunately, completing one university course "Artificial Intelligence and Machine Learning" did not yet qualify the author call himself as a Data Scientist or AI test expert. Anyhow, after finalizing the thesis he might be on a solid path for targeting the role of AI Test Expert whenever that role becomes available. That remains for the AI experts and data scientists to evaluate.

Anyhow even this research provides promising results for implementing the AI testing, there also is criticism against the AI testing. Jeff Nyman for example in his blog post asks any AI testing enthusiasts to show evidence in AI testing having improved efficiency against the traditional test automation in running the tests or alerting the problems. He had no replies in his original challenge nor for the post he presented (Nyman, 2019). There are no answers in this kind of questions. Jason Arbon, in the AIST newsletter on august 2019, says that it would require eight years for his AI agents to perform this kind of arbitrary testing Nyman presented in the challenge. AI testing is not complete in any means. There are only approaches, many of them proven efficiency in the areas they have been presented. For example, by evaluating some of the tools presented in this research, improvement or efficiency in maintaining the tests could be seen when changes are done in the system being tested. But this is not AI testing in its full scale.

7.2 Future of the AI Testing

This brings us in the possible future of the AI testing which is discussed by Tariq King in the AIST “masterclass” presentation (King, 2019). The first step towards full scale AI testing is enhancing the existing testing tools with machine learning features. This is happening already today with the example tools presented in this research by for example the image classification integration with Appium.

The second step is to replace the current and traditional tools with more advanced AI testing tools and frameworks. These frameworks still require human intelligence to validate and train the system on the areas where machine learning approaches have not found solutions. The test oracle problem would most likely be the last one to be implemented.

The third step is for the AI takin over the human activity. Tariq’s opinion is this will happen but cannot say when. He throws the responsibility for us test specialists, architects, system developers, data scientists and anyone contributing in the development work.

8 References

AGENT, 2019. *AGENT - AI Generation and Exploration in Test.* [Online]
Available at: https://github.com/UltimateSoftware/AGENT

AISTA, 2019. *Artificial Intelligence for Software Testing Association.* [Online]
Available at: https://www.aitesting.org/

Applitools, 2019. *AI Powered Visual UI Testing and Monitoring.* [Online]
Available at: https://applitools.com/

Apptest.ai, 2019. *Apptest.ai | Mobile App Test.* [Online]
Available at: https://apptest.ai/#/main/home

Arbon, J., 2018. *AI for Element Selection - SeleniumConf Chicago.* [Online]
Available at: https://youtu.be/3YLxZfZHnPg

Arbon, J., 2018. *AI for Software Testing - AICamp.* [Online]
Available at: https://youtu.be/eit9DYa_DMg

Arbon, J., 2019. *The AI Testing Singularity | STAREAST 2019.* [Online]
Available at: https://www.youtube.com/watch?v=9xLnHcMIhhk

Arbon, J. et al., 2018. *Abstract Intent Test (AIT) Syntax.* [Online]
Available at: https://goo.gl/XbwgkL

Authors of the Agile Manifesto, 2001. *Manifesto for Agile Software Development.* [Online]
Available at: http://agilemanifesto.org/

Braga, R. et al., 2018. *A machine learning approach to generate test oracles.* Sao Carlos, Brazil, ACM.

Buenen, M. & Walgude, A., 2018. *World Quality Report 2018-19 - Executive summary.* [Online]
Available at: https://www.sogeti.com/globalassets/global/wqr-201819/wqr-2018-19_secured.pdf

Chen, J. et al., 2018. *Optimizing test prioritization via test distribution analysis.* Lake Buena Vista, FL, USA, ACM.

Crispin, L., 2019. *Blog - Practical use cases for machine learning.* s.l.:Mabl.

Diffblue, 2019. *Diffblue - AI for Code.* [Online]
Available at: https://www.diffblue.com/about-us

DrQA, 2019. *Reading Wikipedia to Answer Open-Domain Questions.* [Online]
Available at: https://github.com/facebookresearch/DrQA

Dsouza, M., 2018. *How artificial intelligence can improve pentesting.* [Online]
Available at: https://hub.packtpub.com/how-artificial-intelligence-can-improve-pentesting/

Dwarakanath, A. et al., 2018. *Identifying Implementation Bugs in Machine Learning Based Image Classifiers using Metamorphic Testing.* Amsterdam, Netherlands, ACM.

Eggplant, 2019. *Eggplant AI.* [Online]
Available at: https://eggplant.io/products/dai/eggplant-ai

Enişer, H. F. & Sen, A., 2018. *Testing Service Oriented Architectures Using Stateful Service.* Gothenburg, Sweden, ACN.

Figure Eight, 2018. *Data Scientist Report 2018,* s.l.: Figure Eight.

Functionize, 2019. *Automation Testing with Machine Learning - Functionize.* [Online]
Available at: https://www.functionize.com/

Gerrard, P., 2017. *The New Model for Testing.* [Online]
Available at: https://youtu.be/1Ra1192OpqY

Hawkins, L., 2018. *Blog | The World Quality Report 2018/19 – applying critical thinking.* [Online]
Available at: https://therockertester.wordpress.com/2018/09/25/the-world-quality-report-2018-19-applying-critical-thinking

Ho, T. K., 1995. *Random Decision Forests.* Montreal, IEEE.

Hristov, A., 2019. *Mabl | End-to-End testing: REVOLUTIONIZED.* [Online]
Available at: https://mabl.wistia.com/medias/vtzm83dg35

Hu, G., Zhu, L. & Yang, J., 2018. *AppFlow: Using Machine Learning to Synthesize Robust, Reusable UI Tests.* Lake Buena Vista, FL, USA, ACM.

ISTQB, 2012. *Advanced Level Syllabus - Technical Test Analyst,* s.l.: s.n.,

ISTQB, 2018. *Foundation Level Syllabus,* s.l.: s.n.

ISTQB, 2018. *Standard Glossary of Terms used in Software Testing.* Version 3.2 ed. s.l.:ISTQB.

Johnson, R. C., 2018. *Overcoming AI Bias with AI Fairness.* [Online]
Available at: https://cacm.acm.org/news/233224-overcoming-ai-bias-with-ai-fairness/fulltext

King, T., 2019. *Masterclass: AI and Machine Learning Skills for the Testing World.* [Online]
Available at: https://youtu.be/gvHnbexnn-4

King, T. M. et al., 2019. *AI for Testing Today and Tomorrow: Industry Perspectives.* Newark, CA, USA, IEEE.

Lachmann, R., 2018. *Machine Learning-Driven Test Case Prioritization Approaches for Black-Box Software Testing.* Nuremberg, Germany, European Test and Telemetry Conference ettc2018.

Lomas, N., 2019. *DeepCode gets $4M to feed its AI-powered code review tool.* [Online] Available at: https://techcrunch.com/2019/08/06/deepcode-gets-4m-to-feed-its-ai-powered-code-review-tool/

Luo, Q., Poshyvanyk, D., Nair, A. & Grechanik, M., 2016. *FOREPOST: a tool for detecting performance problems with feedback-driven learning software testing.* Austin, Texas, ACM.

Mahendru, K., 2019. *An Introduction to the Powerful Bayes' Theorem for Data Science Professionals.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2019/06/introduction-powerful-bayes-theorem-data-science/

Marijan, D., Gotlieb, A. & Ahuja, M. K., 2019. *Challenges of Testing Machine Learning Based Systems.* Newark, CA, USA, USA, IEEE.

Martinez, J., Thomas, T. & King, T. M., 2014. *Echo: A Middleware Architecture for Domain Specific UI Test Automation.* s.l., ACM.

Meinke, K. & Bennaceur, A., 2018. *Machine Learning for Software Engineering: Models, Methods, and Applications.* Gothenburg, Sweden, IEEE.

Meinke, K., F.Niu & Sindhu, M., 2012. *Learning-Based Software Testing: A Tutorial.* s.l., Springer.

Memon, A. M., Pollack, M. E. & Soffa, M. L., 1999. *Using a goal-driven approach to generate test cases for GUIs.* Los Angeles, USA, IEEE.

Mosley, D. J. & Posey, B. A., 2002. *Just Enough Software Test Automation.* s.l.:Prentice Hall.

Nyman, J., 2019. *AI Test Challenge Follow Up.* [Online]
Available at: http://testerstories.com/2019/05/ai-test-challenge-follow-up/

RaiMan, 2019. *RaiMan´s SikuliX.* [Online]
Available at: http://sikulix.com/

Raval, S., 2017. *Q Learning Explained (tutorial).* [Online]
Available at: https://youtu.be/aCEvtRtNO-M

Reid, S., 2014. *Response to Stop 29119 Petition.* [Online]
Available at: http://www.softwaretestingstandard.org/29119petitionresponse.php

Rodriguez, D., Dolado, J. & Tuya, J., 2015. *Bayesian concepts in software testing: an initial review.* Bergamo, Italy, ESEC/FSE 2015.

Roper, M., 2019. *Using Machine Learning to Classify Test Outcomes.* Newark, CA, USA, USA, IEEE.

Russell, S. & Norwig, P., 2016. *Artificial Intelligence: A Modern Approach, Third Edition.* s.l.:Pearson Education Limited.

Santiago, D., Alt, P., Clarke, P. J. & King, T. M., 2018. *Abstract Flow Learning for Web Application Test Generation.* Lake Buena Vista, FL, USA, ACM.

Santiago, D., Clarke, P., Alt, P. & King, T. M., 2018. *Abstract flow learning for web application test generation.* s.l., ACM.

SEWORKS Team, 2018. *The Official Launch Of Our AI-Powered Pen Testing Solution.* s.l.:SEWORKS.

Sofy, 2019. *Sofy.ai.* [Online]
Available at: https://sofy.ai/home

Spieker, H., Marijan, D., Gotlieb, A. & Mossige, M., 2017. *Reinforcement Learning for Automatic Test Case Prioritization.* Santa Barbara, CA, USA, ACM.

Splunk, 2019. *Artificial Intelligence and Machine Learning | Splunk.* [Online]
Available at: https://www.splunk.com/en_us/explore/machine-learning-artificial-intelligence.html

Subramanian, R., 2018. *How AI is transforming software testing - SeleniumConf Chicago.* [Online]
Available at: https://youtu.be/pMd1L1lZrxk

Sutton, R. S. & Barto, A. G., 2018. *Reinforcement Learning: An Introduction, second edition.* s.l.:The MIT Press.

Tatar, E., 2019. *Mabl - Blog - The truth about ML in test automation.* [Online]
Available at: https://www.mabl.com/blog/the-truth-about-ml-in-test-automation

Tatar, E., 2019. *Mabl - Blog - The truth about ML in test automation.* [Online]
Available at: https://www.mabl.com/blog/the-truth-about-ml-in-test-automation

test.ai, 2019. *test.ai: AI Automated Software Testing, Mobile App Index.* [Online]
Available at: https://www.test.ai/

TestCraft, 2019. *Codeless Selenium Test Automation with AI-Based Maintenance | TestCraft.* [Online]
Available at: https://www.testcraft.io/#about

Testim.io, 2019. *Continuous Testing Automation Software Tool | Testim.io.* [Online]
Available at: https://www.testim.io

Testrigor, 2019. *TestRigor - Autonomous Testing.* [Online]
Available at: https://testrigor.com/

University of Helsinki & Reaktor, 2019. *Elements of AI.* [Online]
Available at: https://course.elementsofai.com/
Van De Ven, T. et al., 2018. *World Quality Report 2018-19 - Artificial Intelligence - World Quality Report 2018-19 Findings:.* [Online]
Available at: https://www.sogeti.com/globalassets/global/wqr-201819/wqr-2018-19_secured.pdf
Zhang, J. et al., 2016. *Predictive mutation testing.* Saarbrücken, Germany, ACM.

Research Keywords

The keywords are collected from the ISTQB Glossary and Certification Syllabi of Foundation level, advanced level test analyst and technical test analyst.

- Software Testing Keywords
  - Software Testing Process
    - Planning, monitoring and control
    - Analysis and design
    - Implementation and execution
    - Evaluating exit criteria and reporting
    - Test closure activities
  - Test Approaches
    - Design-Based Testing
    - Risk-Based Testing
    - Model-Based Testing (MBT)
  - Test targets – Software quality characteristics
    - Functionality
    - Reliability
    - Usability
    - Efficiency
    - Performance (time behaviour), resource utilization, compliance
    - Maintainability
    - Portability
  - Test Techniques
    - Specification based techniques, also known as black-box techniques.
    - Structure based techniques, also known as White-box techniques
    - Defect based techniques
    - Experience based techniques
  - Analytical test techniques
    - Static analysis
    - Dynamic analysis
  - Test Tools
    - Test management tools
    - Requirements management tools

        - Defect management tools
        - Tools for the static testing
            - Review tools
            - Static analysis tools
    - Test design and implementation tools
    - Test data preparation tools
    - Test execution and logging tools
        - Test execution tools
        - Coverage tools (for example requirements or code coverage)
        - Test harnesses
        - Unit test framework tools
    - Tools for performance measurement and dynamic analysis
        - Performance testing tools
        - Monitoring tools
        - Dynamic analysis tools
    - Tools for specialized testing needs
        - Data quality assessment
        - Data conversion and migration
        - Usability testing
        - Accessibility testing
        - Localization testing
        - Security testing
        - Portability testing
- Software Test Automation
    - Test scripting techniques
        - Linear scripting
        - Process-driven scripting
        - Structured scripting
        - Data-driven testing
        - Keyword- or action word-driven testing
- Test levels
    - Component/Unit testing
    - Integration testing
    - System testing
    - Acceptance testing

- Regression testing

Standards on the Software Testing and Software Quality

Standards on the Software Testing

ISO/IEC/IEEE 29119 Software Testing

ISO/IEC/IEEE 29119-1: Concepts & Definitions

ISO/IEC/IEEE 29119-2: Test Processes

ISO/IEC/IEEE 29119-3: Test Documentation

ISO/IEC/IEEE 29119-4: Test Techniques

ISO/IEC/IEEE 29119-5: Keyword-Driven Testing

ISO/IEC 33063: Process Assessment Model for Software Testing

Standards on Software Quality

ISO/IEC 9126-1:2001 Software Engineering – Product Quality – Part 1: Quality Model

The Tools and Technologies to Base the AI Testing

Test driver examples:

- Selenium WebDriver
- Sikuli
- Robot
- Appium
- Calabash
- Robotium
- JUnit
- Nunit
- xUnit
- Custom built

Machine Learning Frameworks:

- Pandas
- Numpy
- Scikit-learn
- Matplotlib
- TensorFlow
- Keras
- Seaborn
- Pytorch & Torch
- AWS Deep Learning AMI
- Google Cloud ML Engine
- Theano
- Microsoft Azure Machine Learning
- IBM Watson Machine Learning
- Amazon SageMaker

Commercial and Open-Source Testing Tools Including AI/ML Features

Commercial AI testing tools

- Mabl
- Test.ai
- Applitools
- Sofy.ai
- Diffblue
- Eggplant
- Apptest.ai
- Functionize
- TestRigor
- Testim
- TestCraft

Open Source AI testing tools

- AI-based Generation and Exploration iN Test (AGENT/AGENT-X)
- Appium, AI element classifier
- DrQA
- SikuliX
- AppFlow