TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

DISTANCE AND INDEPENDENT
STUDIES CENTER

Herausragende Masterarbeiten
am Distance and Independent Studies Center

Studiengang: Software Engineering for Embedded Systems, M.Eng.

Masterarbeitstitel: Improving the Situation of Analytical Quality Assurance in Agile Product Developments

Autor*in: Johannes Christopher Rottmann

Declaration

Ich versichere, dass ich diese Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Place, date

Signature

Abstract

With growing prevalence, agile methodology also pervades domains which adhered to conventional models for decades. At the same time, the demand for safety critical applications and thus rigorous quality assurance increases. This raises the question whether agile methodology is able to support the required level of quality assurance.

This master's thesis aims at analyzing the situation of analytical quality assurance in agile environments in order to identify shortcomings and provide potential solutions. The author derives an initial hypothesis based on his own professional experience, stating that analytical quality assurance is not sufficiently considered by agile development models and agile transformation. This hypothesis is split into eight sub-hypotheses, each describing a particular problem or challenge. Qualitative interviews with seven experts and complementary literature researches are performed to confirm given hypotheses, identify further challenges, and collect appropriate solution proposals. Eventually, based on the elicited data, five hypotheses as well as the initial hypothesis are corroborated and five new challenges are added. Furthermore, twenty-six potential solutions for relevant hypotheses are collected and presented. The solutions comprise established approaches, such as Dynamic System Development Model or Explorative Testing but also innovative ideas, including the Three-Field Agile approach publicized by this thesis.

Altogether, it is found that agile methodology largely not supports traditional analytical quality assurance in its concepts and even worse, some of the core principles are contradictive. However, numerous solutions are found and presented that address particular discrepancies and have the capability to ease the pictured situation.

Table of Contents

Abbreviations

AQA	Analytical Quality Assurance
CL	Confidence Level (used in Table 8)
CQA	Constructive Quality Assurance
DEV	(Software) Development
DSDM	Dynamic Systems Development Model
FAL	Final Agreement Level (used in Table 8)
IQA	Integrated Quality Assurance
IVx	Interview Number x (used in Table 8)
LFAL	Lowest Possible Final Agreement Level
LIx	Literature Number x (used in Table 8)
MGMT	Management
PL	Project Leader
PO	Product Owner
QA	Quality Assurance
RQE	Requirement Engineering
SDS	Statement Documentation Sheet
SE	System Engineering
V&V	Verification and Validation

List of Tables

List of Figures

1 Introduction

This chapter presents the goal and structure of this thesis and provides necessary background information to help understanding the context and existing body of knowledge in the area of interest.

1.1 Context

Nowadays, eighteen years after the publication of the Agile Manifesto, agile is the dominant software development model (Flynn, 2022), even though the transformation was not always easy. Especially large, conservative companies struggled to familiarize with the agile mindset and the consequences it had on the different domains. Today, these problems seem to be solved and agile methodology is used by 86% of international software developers (Flynn, 2022). But are really all problems identified and addressed?

At the same time agile became popular, another aspect of development gained importance due to more complex and safety critical products - quality assurance (Liggesmeyer, 2020). This discipline was historically deeply integrated in stage-gate-models, conventional development processes, and strict schedules. “Processes, tools, documentation, and following a plan” were “the cornerstones of rigorous quality assurance and testing practices” (Itkonen, Rautiainen, & Lassenius, 2005, p. 1). In sharp contrast to these fundamental requirements, the agile approach emphasizes “individuals and interactions, customer collaboration, and responding to change” (Itkonen, Rautiainen, & Lassenius, 2005, p. 1). It clearly favorizes constructive quality assurance methods, while analytical practices with destructive attitude, such as traditional testing, are almost completely abandoned (Itkonen, Rautiainen, & Lassenius, 2005). In fact, the twelve agile principles contain the word “develop” five times and “software” three times, while “quality”, “test”, or “validate” is not mentioned at all (Beck, et al., 2001).

These discrepancies give room to the initial assumption that analytical quality assurance in agile projects might be subject to problems or at least procedural compatibility issues.

1.2 Initial Hypothesis

In his professional career the author worked in the domains requirements engineering, system engineering, and testing for a variety of companies from different business areas. During nine years the author took part in the transformation from conventional development models such as waterfall and V-model to agile or partly agile approaches multiple times. While many of the challenges related to agile transformation were more or less successfully solved by the majority of companies, the author noticed a rising level of unsolved and even undetected problems in the domain of analytical quality assurance. Even though the companies differed in size and structure, they all struggled to successfully integrate analytical quality assurance in their agile processes or did not even consider it at all. The resulting problems in quality assurance were diverse. To name just a few examples:

- Incapability to keep up with the fast pace of the development
- Inability to plan or estimate upcoming effort and cost
- Employees got frustrated by unnecessary reworks and the lack of information

Eventually, these problems decreased the efficiency and motivation in the quality assurance teams and caused friction at the interfaces to other domains, such as development or requirement engineering. Additionally, due to the decreased efficiency the pressure on the teams and associated employees rose which led to a spiral of demotivation in the worst cases.

The author believes these observations are recurrent and not limited to specific companies or business areas. In the opinion of the author this is particularly true for upper test levels, where complex products under test are composed of several sub-components and testing happens on a more abstract level. Even today there remain systematical issues in this regard that have not been addressed or even identified and analyzed. The author derives this opinion into the following hypothesis:

“Analytical quality assurance (higher level testing in particular) is not sufficiently considered and integrated by the agile approach and agile transformation. As a result, it faces an increased number of challenges when executed in an agile environment. This eventually leads to shortcomings and decreased efficiency in analytical quality assurance.”

1.3 Goal and Structure

The ultimate goal of this thesis is to elaborate whether the given hypothesis is valid and if so, provide solution proposals to improve the situation of analytical quality assurance in agile environments.

To achieve this goal, the given hypothesis is first derived into several sub-hypotheses with a deeper level of detail in Chapter 2, based on the experience of the author. In Chapter 3 field data is collected by means of expert interviews and literature research. The collected field data is analyzed and categorized in Chapter 4 in order to assess and validate the derived hypotheses in Chapter 5. Finally, the thesis provides solution proposals for the most relevant hypotheses based on literature research and experience in Chapter 6. The general conclusion on the initial hypothesis, supplemented by an outlook for potential future research on this topic completes the thesis in Chapter 7. The structure is also displayed in Figure 1.

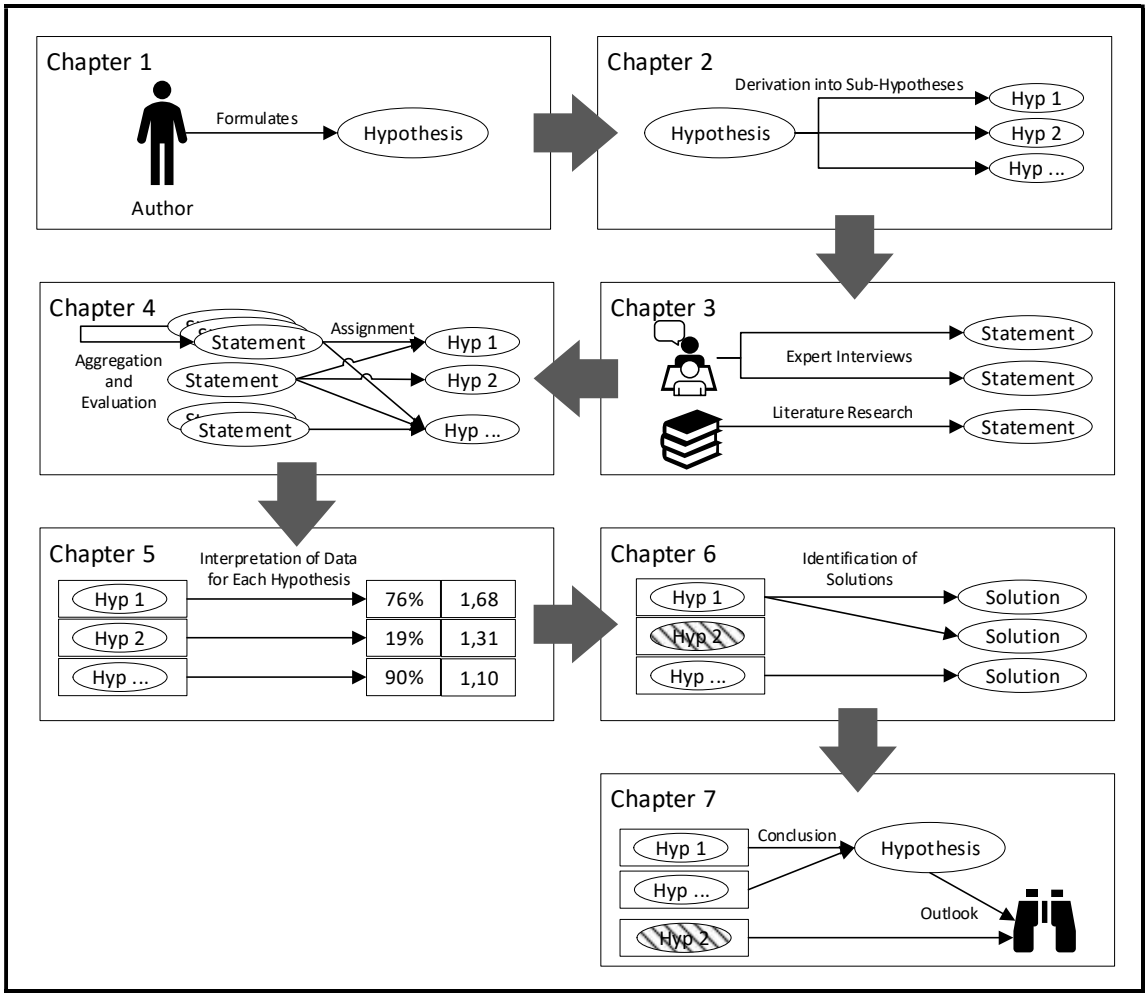


Figure 1: Structure of Thesis

1.4 Existing Body of Knowledge

This chapter explains terms and models which are crucial for the correct understanding and classification of the content of this thesis.

1.4.1 Quality and Quality Assurance

The terms quality, quality assurance, and integrated quality assurance are described first, followed by the explanation of constructive quality assurance and analytical quality assurance. The general dependencies of these terms are portrayed in Figure 2.

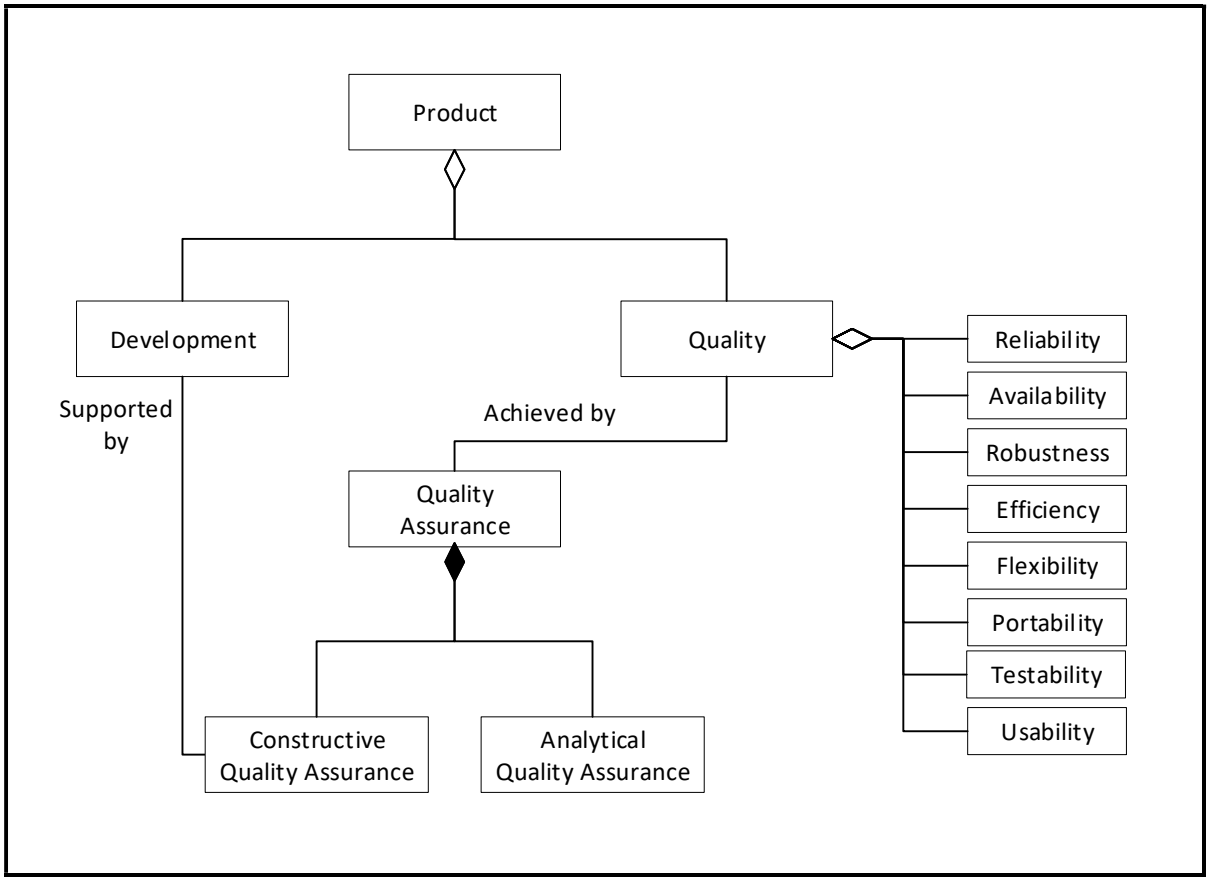


Figure 2: Dependencies of Quality Terms

Quality

Often people say that a product "has a certain quality" or "is of good quality" when they refer to the product as a whole from end customer point of view. In the engineering world the quality of a product is examined a bit more sophisticated and can be evaluated based on different characteristics and the product's ability to achieve these characteristics to a certain degree. Typical quality characteristics are security, reliability, availability, robustness,

memory and runtime efficiency, flexibility, portability, testability, and usability (Liggesmeyer, 2020, p. 5) but there a numerous publications and standards on quality that propose slightly different or more characteristics. The ISO 9000 defines quality as “degree to which a set of inherent characteristics of an object fulfils requirements” (International Organization for Standardization, 2015).

Quality Assurance

Quality assurance (QA) is a discipline whose main concern is to ensure that a product is able to deliver the desired maturity on relevant quality characteristics. In the early phases, modern QA was mostly limited to the validation of the product quality according strict, static guidelines and executed just before release to the end customer. The QA could either agree on releasing the product or rejected it due to insufficiencies. In the latter case, the product was either cancelled or the development had to revise the product until it was mature enough to pass the quality validation.

This limitation changed heavily in recent decades and companies started to blend the QA discipline into the actual development process. Instead of only being the judge who passes the final sentence, QA additionally became the teacher or mentor who helps the team and product to grow up. Today, the final quality validation is only a fraction of the scope of QA. Main duties are now to support other disciplines during development and to validate intermediate development artifacts in order to ensure that desired product quality can eventually be met. One of the concepts that mitigates these changes is called the “principle of integrated quality assurance” (Liggesmeyer, 2020, p. 2).

Integrated Quality Assurance

Being involved during the whole lifecycle of the product and performed in parallel to concurrent engineering integrates the QA. It has major influence on other activities during the development and many interfaces to other disciplines. Tasks which arise by this concept include, but are not limited to:

- Identification of relevant quality stakeholders
- Elicitation of quality related requirements from these stakeholders
- Identification, creation, and integration of appropriate methods and processes into the development model to ensure sufficient quality of intermediate and final products

- Validation of intermediate development products
- Validation of final product or set of products before market release

As outlined, the concept of integrated quality assurance (IQA) considers the different phases of a development project from support and validation point of view. To address the different needs, IQA divides related activities in two major domains: constructive quality assurance (CQA) and analytical quality assurance (AQA). Both domains present a unique set of methods and goals which can be applied to the different activities or artifacts during the course of a project (Liggesmeyer, 2020, pp. 3-4). When CQA and AQA methods are adequately combined, the quality of the final product can be significantly improved. The difference between CQA and AQA is essential: While CQA aims at achieving the desired quality in the intermediate artifacts, AQA aims at avoiding releases of artifacts/products with insufficient quality to the subsequent consumer (not necessarily the final customer). The general concept of IQA is outlined in Figure 3, inspired by (Liggesmeyer, 2020, p. 4).

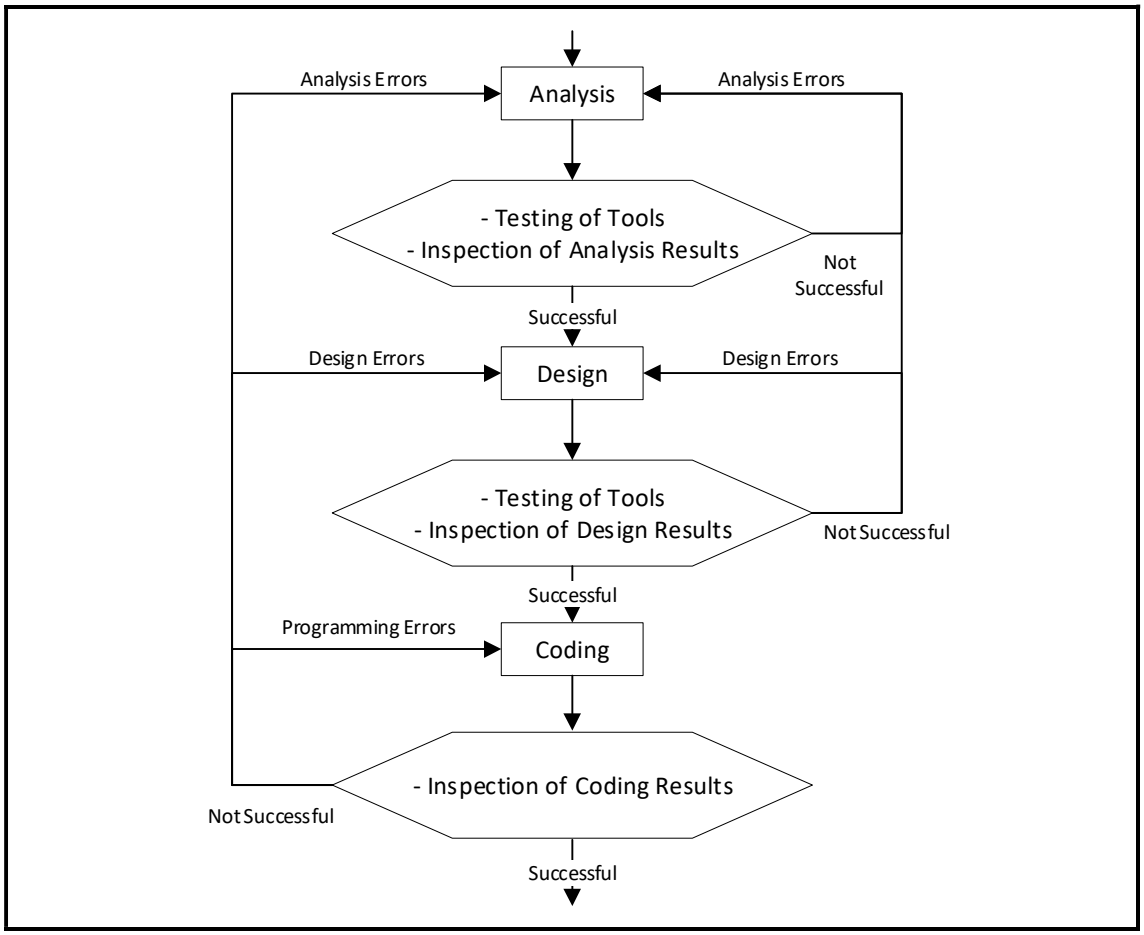


Figure 3: Integrated Quality Assurance

Constructive Quality Assurance

As it is not possible to test good quality into a product, good quality must be constructed. Consequently, agile methods move some QA responsibilities and work to the developers (Hue, Verner, Zhu, & Babar, 2004). Typical CQA methods are:

- IDE Configuration
- Team Training
- Improved Team Communication
- Proper Tooling
- Guidelines
- Well-documented Development Process
- Formalism in Code
- Risk management
- Structured Analysis
- Structured Design
- Unified Modeling Language
- Pair Programming (is actually 50% CQA and 50% AQA)

Especially in evolutionary development projects, the actual developers are now responsible to select and apply appropriate constructive methods autonomously without being supervised or managed by an external instance or QA domain. “The best architectures, requirements, and designs emerge from self-organizing teams” (Beck, et al., 2001). Generally, constructive methods tend to be more compatible with the agile mindset as it demands to “build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done” (Beck, et al., 2001). Liggesmeyer however still considers AQA useful because “no constructive procedure can guarantee error-free products. For this reason, the quality needs to be tested with analytical methods” (Liggesmeyer, 2020, p. 2).

Analytical Quality Assurance

In contrast to the mentioned CQA, AQA is concerned with the original task of QA; validation of the quality of products to be released. The difference with respect to traditional QA are the multiple entry points of the validation activities throughout the whole development process. Not only the final product, but also intermediate products which are output by the different

phases of a project are validated to ensure a consistent maturity during the development. In his book, Liggesmeyer describes the advantages:

> An intermediate product whose quality is inadequate would not be forwarded to the next phase until a defined, satisfactory quality level has been attained. This procedure facilitates early detection of errors and enables their correction at a time when this is possible with minimal expense. The aim is to detect as many errors as possible during the tests that immediately follow the development step where the error originated. The aim is to minimize the number of errors persisting over several phases.
> (Liggesmeyer, 2020, pp. 2-3)

Most of the methods which are facilitated in this context can be related to as “testing”. However, AQA includes some methods that lie beyond the scope of normal testing such as requirement reviews or code metric reviews. Typical examples of AQA methods are:

- Requirement Reviews
- Pair Programming (is actually 50% CQA and 50% AQA)
- Code Complexity Checks
- Static Testing Techniques
    - Static Code Analysis
    - Data Flow Anomaly Analysis
    - Slicing
- Dynamic Testing Techniques
    - Control-Flow Oriented Tests
    - Function Oriented Tests
    - Boundary Value Tests

The general character of these methods is typically rather destructive than constructive. The main goal is not to improve the product but rather to find ways to break it and avoid the release of a breakable product to the next phase or eventually the customer. This contradicts some of the most basic principles of agile development: “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done” (Beck, et al., 2001). Furthermore, they are traditionally executed by specialized validation teams, external to the development domain. Nowadays and particularly in agile

development projects, some of the AQA methods are also performed by the developers themselves. This is particularly true for methods which have a strong correlation with the actual software code, such as code complexity checks or code reviews. On the one hand, this yields an easier, faster integration of the methods but on the other hand, systematical misunderstandings cannot be detected. Methods which are applied to more abstract product levels like system test or user acceptance test are difficult to perform on development level. If a complex product is composed of several sub-components, each developed by a separate team, the tests cannot be carried out by sub-component developers due to a lack of information of the overall product or simply because the composed product is not available to them. For this reason, in projects with complex and/or composed products most of the higher level AQA activities are still carried out by specialized teams external to the development.

In the opinion of the author, these teams and their higher level AQA methods were not sufficiently considered by the agile transformation. Agile was, and is, primarily an approach for dynamically creating products rather than validating them. An approach for developers rather than testers. It relies heavily on build-in quality and fast user feedback loops while rendering external validation neglectable. This might be handy for some products like smart phone applications and webservices but it hits hard when it comes to complex or composed products, products with low release frequency, or safety critical domains.

1.4.2 Conventional Development Models

Conventional, sequential development models were dominant for decades and their relicts still can be seen in many companies, processes and regulations, such as Automotive SPICE or the ISO 26262. Additionally, there are projects, mostly safety critical ones, where conventional development approaches continue to play a role in development.

While there are also differences among the conventional development approaches, they all commonly intend to execute each phase of development only once and in a predefined, sequential order. The QA activities, especially the task of validation, is placed at the very end of the project, after the development is completed. This is a major drawback because errors are identified late in the project and according to Boehm's first law, the effort and cost for fixing them is very high (Rombach & Wehn, 2014). At the same time, it guarantees that on AQA start, all required inputs are available and final. This is beneficial for AQA.

Two prominent conventional development models are the Waterfall Model and the V-Model. Both are portrayed in Figure 4 and comply to a strict sequential development flow but the V-Model additionally locates some AQA preparation tasks in earlier phases and connects development phases logically to certain validation activities.

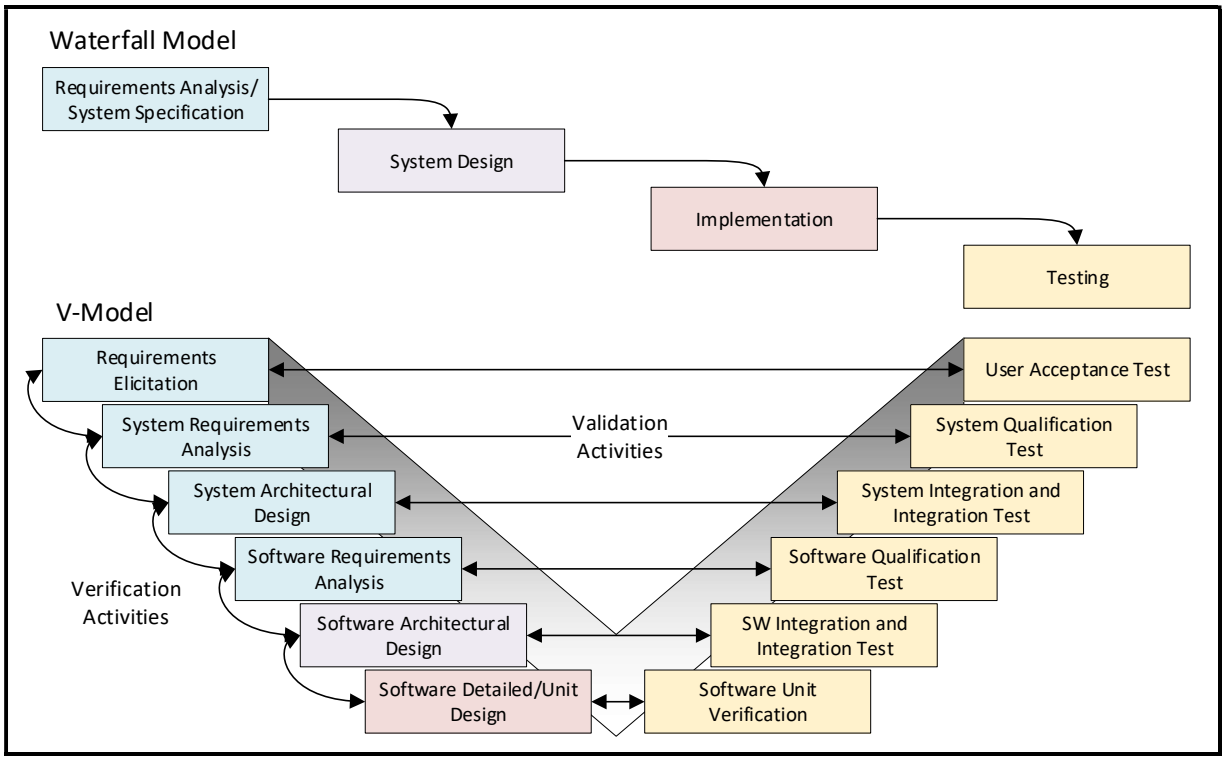


Figure 4: Conventional Development Models

1.4.3 Agile Development Models

Nowadays, many software related companies facilitate agile, or at least incremental development approaches for their software products. The difference compared to the conventional approaches lies in their commonly iterative nature. Generally, two different kinds of iterative models exist:

- If all development phases are continuously repeated until the product is mature enough to be released, it is called evolutionary development. The product evolves from iteration to iteration, meaning the requirements are also elicited in an iterative manner and the final product is not defined at project start (Pews & Möhrle, 2017). This makes it difficult to estimate effort or plan ahead but gives freedom to reshape the product very dynamically during development.
- In contrast to that, if only the design, implementation, and testing phases are repeatedly executed whereas the requirements phase is performed just once at project start, it is called incremental development. The definition of done is clear right from the start but the freedom to alter the requirements or deviate from the initial plan is limited (Pews & Möhrle, 2017).

The agile approach itself is mainly promoted as an evolutionary approach but often companies de facto implement incremental models and add Scrum as an agile process framework. Although Scrum is the most prominent agile framework, there are plenty of other agile approaches like Extreme Programming, Dynamic System Development Method, or Lean Software Development (Pews & Möhrle, 2017). Figure 5 gives an overview over the iterative core process of the agile approaches.

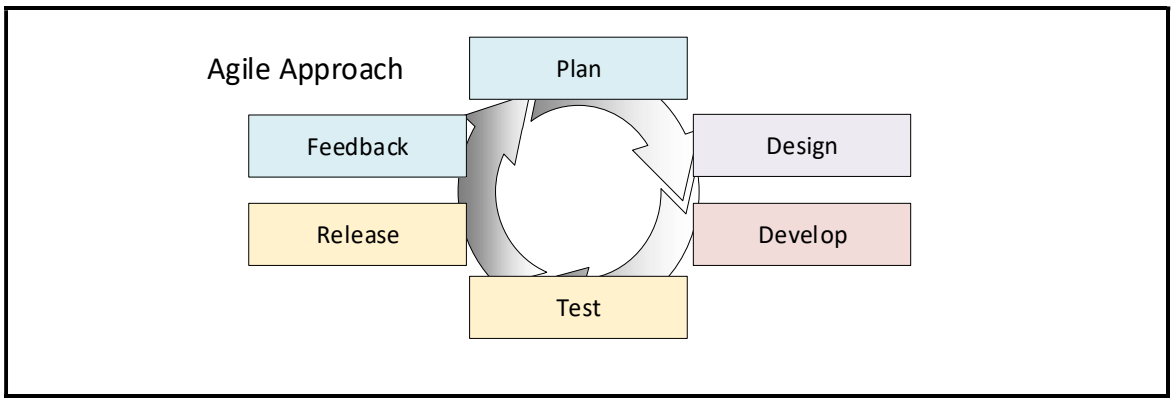


Figure 5: Agile Development Models

2 Derived Sub-Hypotheses

In this chapter sub-hypotheses are derived from the initial hypothesis, stated by the author in Chapter 1. The goal is to specify each derived hypothesis with sufficient detail in order to enable evaluating it based on literate research and expert interviews in the next chapters.

2.1 Hypothesis 1: Planning

2.1.1 Hypothesis

“Giving a reliable forecast on overall development effort and cost is very difficult in (real) agile projects. Forecasting overall AQA effort and cost is even worse and almost impossible but often still demanded.”

2.1.2 Problem Description

Based on the nature of a real evolutionary agile development, it is almost impossible to estimate the overall effort and cost at project start. Every new development cycle implies the opportunity to alter, add, or remove features and requirements. At the start of the project the end is not yet defined. This is a major characteristic of agile development and one of the biggest advantages and drawbacks at the same time. It allows very quick response times to customer feedback and supports a dynamic growth of the product but at the cost of being largely unpredictable with regard to effort and cost. Although this fact is widely understood and accepted for the actual development teams, many external AQA teams are still integrated in traditional, quality-based stage-gate-models and committed to give effort and cost estimations in advance. This is very problematic, because being a consumer of agile development products, estimating effort and cost is obviously even harder for AQA than for the actual development teams themselves.

Additionally, most AQA activities can only be performed after development completed. This also holds true for agile projects. AQA is still reliant on various deliveries from other domains. In agile environments it is just the frequency of deliveries that changed. Consequently, every delay in any other domain will eventually have an impact on the planning of AQA.

2.2 Hypothesis 2: Test Effort

2.2.1 Hypothesis

“If a hypothetical project is developed both ways, conventional and agile, the effort and cost spend on AQA in the agile project is significantly higher as is the product quality.”

2.2.2 Problem Description

The quality of an agile product in terms of customer satisfaction is supposed to be higher compared to a conventionally developed product due to the continuous feedback loops. If this is taken for granted, the agile approach either has a higher efficiency than conventional development models or there is an increased effort somewhere in the agile project. This additional effort seems to be hidden in AQA. In conventional projects a full test is carried out once or twice after development is completed. This means there is little time spend on AQA until the product is theoretically finalized. In contrast to that, in agile environments AQA is already conducted throughout the complete project but usually still contains a final full scope test run just like in conventional projects. This means all AQA performed during development comes extra as shown in Figure 6. Thus, the total effort spent on agile exceeds the total effort spend in the conventional project largely.

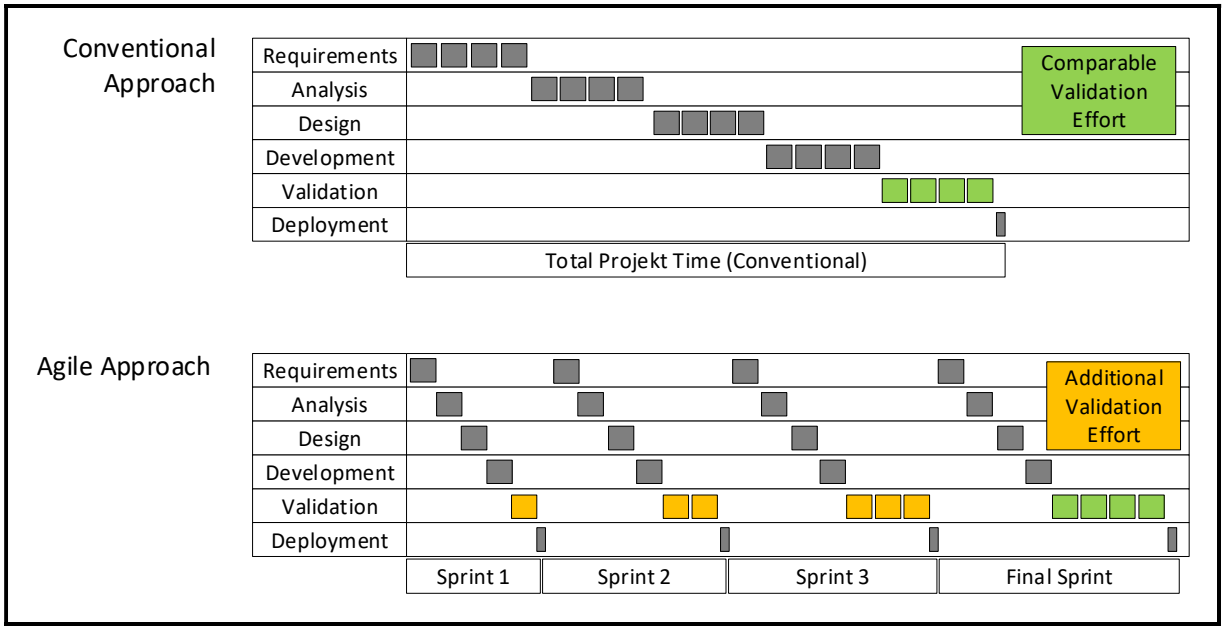


Figure 6: Additional AQA Effort

2.3 Hypothesis 3: Regression

2.3.1 Hypothesis

“Agile development potentially promotes regression which makes countermeasures necessary. Appropriate measures could be installed in CQA but mostly they are performed as AQA which is very ineffective and expensive.”

2.3.2 Problem Description

Regression is a major threat that generally has to be taken care of. Yet, in the agile world some factors come together that make regression an even bigger menace:

- The change embracing mentality comforts changes, even if they are intensive and happen late. As changes are one of the major root causes of regression bugs, subsequently more regression bugs will be introduced during agile projects.
- The lack of documentation and architecture can lead to an increased number of regression bugs due to the ripple effect (Wehn, De Assis, Kuhn, Jung, & Morgenstern, 2016).
- Every increment is released to the customer. To avoid losing trust at the customer side due to high number of bugs, regression has to be considered for each increment.
- The high release frequency also increases the frequency of potential countermeasures.

Although alternative countermeasures in CQA theoretically exists, these measures imply very coordinated and well-designed development. Instead, the consideration of regression is often left over for AQA, for example in the form of a full regression test on every release. As shown in Figure 6, this obviously causes a tremendous amount of effort in AQA. Not only has the delta to the previous increment to be tested, but at least a rough test coverage is expected for the full functional scope of the product. Many AQA teams try to master the situation by establishing test automation but this brings drawbacks itself, as explained in the next hypothesis. Often, a decision is made whether it is eligible to spend the effort of performing a full regression test on every increment or just consider the functional delta and take the risk to release a product with undetected regression bugs. If suitable countermeasures were introduced in CQA and combined with some AQA regression tests on smoke level, the effort in AQA but also the risk of remaining regression bugs could decrease to an acceptable level.

2.4 Hypothesis 4: Test Automation

2.4.1 Hypothesis

“Test automation is widely accepted as the only way for AQA to keep up with the fast-paced agile development but it also introduces new challenges itself.”

2.4.2 Problem Description

As explained by Hypothesis 2 and 3, the effort for AQA grows from sprint to sprint. At the beginning of a project only few features are implemented and sufficient AQA is quickly achieved. When the product matures and features are added, AQA effort increases. Not only the tests for scope added in the current sprint have to be prepared and executed but also regression tests should be run on the legacy functions. One potential solution to master the increasing AQA effort is the introduction of test automation but what sounds like a handy way of keeping work manageable, can become a burden itself.

Test automation is a complex task which requires not only domain knowledge but also diverse technical skills. For this reason, automation of AQA measures on low levels of abstraction is often delegated to the development team. On higher levels of abstraction usually specialized external teams take care of the automation but this yields some issues.

On the higher levels of abstraction, changes made in the sub-components accumulate. Unfortunately, test automation is very vulnerable to change. A human tester can intuitively adopt to uncommunicated changes in the product during test execution. Test automation has no such sense. Even minor unexpected changes will cause the tests to fail. To avoid a disconnect between actual product and test automation, extensive rework and maintenance are continuously carried out to keep the test automation strapped to the product under test.

In the worst case, the time spend on maintenance can even exceed the time that would have been spend on manual regression testing and thus render the test automation an issue itself.

2.5 Hypothesis 5: Changes

2.5.1 Hypothesis

“The agile way on how to deal with changes causes a serious decrease of AQA efficiency due to an excessive amount of rework and duplication.”

2.5.2 Problem Description

The agile approach is designed to “welcome changing requirements even late in development” (Beck, et al., 2001). At the same time, it values “working software over comprehensive documentation” and “individuals and interactions over processes and tools” (Beck, et al., 2001). This mindset has consequences on the projects:

- The number of changes is higher.
- Extensive changes can happen anytime, even late in the project.
- Changes are not thoroughly analyzed and evaluated with regard to necessity.
- Changes are performed on the fly and not communicated or documented.
- Legacy artifacts/ information are not updated or removed when obsolete.

In practice, this results in a very comfortable situation for parties which profit from flexible development with narrow formalism. Parties like customers, product owners, or developers, who primarily output artifacts instead of consuming them, can work unrestrained and efficient with low overhead. Unfortunately, AQA is mostly located on the opposite site. It requires many input artifacts from other domains but produces few outputs and is excessively vulnerable to bad or missing input artifacts, be it information, documentation, or code. The consequences mentioned above are root causes to following issues in AQA:

- AQA has to poll information on changes, causing overhead effort.
- AQA expectations deviate from actual and targeted product behavior.
- AQA delivers wrong validation results (false positive, false negative, test failure).
- Continuous rework of AQA artifacts causes high maintenance effort.
- Avoidable duplication work cannot be identified and time is wasted.
- Efficiency of AQA decreases, delays accumulate, and resentments spread.

2.6 Hypothesis 6: Test Levels

2.6.1 Hypothesis

"The effects of agile development on AQA are not evenly distributed or uniform. Some positive effects apply to activities on levels of lower abstraction, whereas levels with higher abstraction are to large extent confronted with challenges."

2.6.2 Problem Description

AQA activities can be structured in different levels, as outlined in Figure 7. The agile approach has varying effects on the different levels.

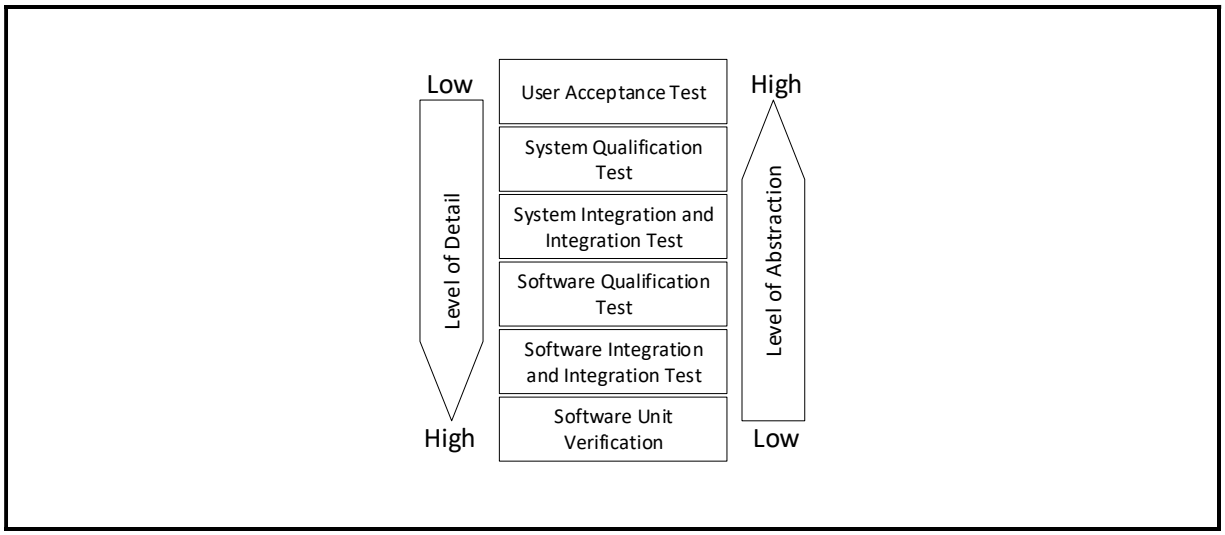


Figure 7: Test Levels

Low levels such as Software Unit Verification include tasks like static/dynamic analysis, code reviews, dataflow and unit testing. These are naturally closer to the actual source code, the fast-beating heart of agile development. For this reason, in agile environments they are often merged with CQA measures and then integrated by the development teams. The advantages of this procedure are short communication and good knowledge of the input artifacts while the most obvious disadvantage is blindness to systematic errors.

In contrast to that, the higher levels like System Qualification Test face challenges due to the agile approach. Sprint results from different teams have to be matched and combined to create the product under test. Additionally, the flow of information from development into these levels is not ideal and results in avoidable errors by misinterpretation or lack of knowledge.

2.7 Hypothesis 7: Tooling

2.7.1 Hypothesis

“If AQA is not operated in the same tool environment together with related domains, a lack of synchronization will be the result. The subsequent implications are especially problematic in agile projects and must be avoided.”

2.7.2 Problem Description

It is common understanding that project-wide use of a homogenous or at minimum a synchronized tool environment is beneficial. Independent from the utilized development model, the basic tasks of the tool environment remain the same:

- Enable effective, persistent, artifact-related intra-domain communication.
- Enable effective, persistent, artifact-related inter-domain communication.
- Provide means to link artifacts for traceability.
- Enable artifact-related, persistent documentation.
- Provide means for status tracking.

In conventional models, inter-domain communication is rare and limited to strict phases or seldom events in the course of a project. Additionally, the communicated content is mainly static with few changes over time, hence the requirements posed to the tooling are few and the proper tooling is not considered critical.

While joint tooling is an advantage in conventional projects, it is a crucial prerequisite in the agile world. The significance and frequency of inter-domain communication is way higher. In addition, the communicated content is less static but almost chaotic due to the dynamic character of evolutionary development. Particularly AQA as a consumer of many artifacts is dependent on a stable, persistent, and traceable communication. Especially in the event of changes it is otherwise difficult to identify affected artifacts. An insufficient tooling creates a gap between development and AQA which eventually leads to a lack of synchronization, misconception of testcases, and overall decreased efficiency.

2.8 Hypothesis 8: Safety

2.8.1 Hypothesis

“The AQA related demands posed by domain specific safety standards/regulations are incompatible or at least difficult to achieve with agile projects.”

2.8.2 Problem Description

Some domain specific safety standards/regulations pose very accurate requirements on processes and artifacts in order to achieve the desired safety maturity level. Especially for AQA this implies many constraints on how things have to be done. To large extends, these requirements are somewhat contradictive to what is postulated in the agile manifesto, as shown in Table 1.

Agile Methodology	ISO 26262 Safety Standard
User stories evolve during development.	Safety is paramount and must be considered right from the start.
“Working software over comprehensive documentation” (Beck, et al., 2001).	Detailed records that enable dependability analysis and provide evidence are crucial.
Changes are mainly communicated verbal or informal.	A thorough and explicit change management is required.
Customer acceptance is primal.	The safety goals must be met.
Is based on evolutionary development principles.	Is built around V-model development.

Table 1: Agile and ISO 26262

If safety standards are combined with agile development, tension rises. While the most CQA methods can be made more or less compatible to meet regulations, there is a sharp contrast in AQA between what is possible and what is required. Being reliant on rigorous input documentation to allow for evidence of sufficient test coverage, the quality of the input artifacts often does not meet expectations. The situation of not being able to deliver obligated test results due to lack of input documentation will cause tension between the teams.

3 Data Elicitation

This chapter explains the methodology used for elicitation of raw data to confirm or dissent the presented hypotheses. A secondary objective is the identification of further challenges or issues, from which new hypotheses can be derived. For this purpose, data is mined from two different sources. Expert interviews on the one hand and literature research on the other. The techniques of both methods are outlined in the following chapter.

3.1 Expert Interviews

3.1.1 Candidate Base

Criteria for Candidate Selection

The quality of the interviews and collected results depends on the right selection of candidates. With regard to the topic of this thesis the aspects shown in the Table 2 are of major importance. All candidates should at least fulfill the minimum requirements on the right side of the table but ideally meet the preferred requirements in the middle column.

Aspect	Preferred Requirements	Minimum Requirements
AQA Experience Level	High, by executing or managing AQA actively for more than two years	Sufficient, by being a provider to or consumer of AQA for at least 2 years or executing or managing AQA actively for more than one year
Agile Development Model Experience Level	High, by participating in at least one agile project	Sufficient, by participating in at least one pseudo-agile project or profound knowledge about agile development in theory
Conventional Development Model Experience Level	High, by participating in at least one conventional project	Sufficient, by participating in at least one conventional project or profound knowledge about conventional development in theory
Cross-Domain Experience (Systematical Understanding)	High, by having experience in at least one additional domain, other than AQA	Low, by having experience in only one domain

Table 2: Selection Criteria

Actual Candidate Composition

The figures below outline how the actual candidate base is staffed with regard to the above-mentioned aspects. While all candidates comply with the minimum requirements, Figure 8 shows that also 71% (five of seven) of the candidates meet the preferred requirements on AQA experience level. Furthermore, a partition of 57% (four of seven) has more than five years of work experience in total as it can be seen in Figure 9.

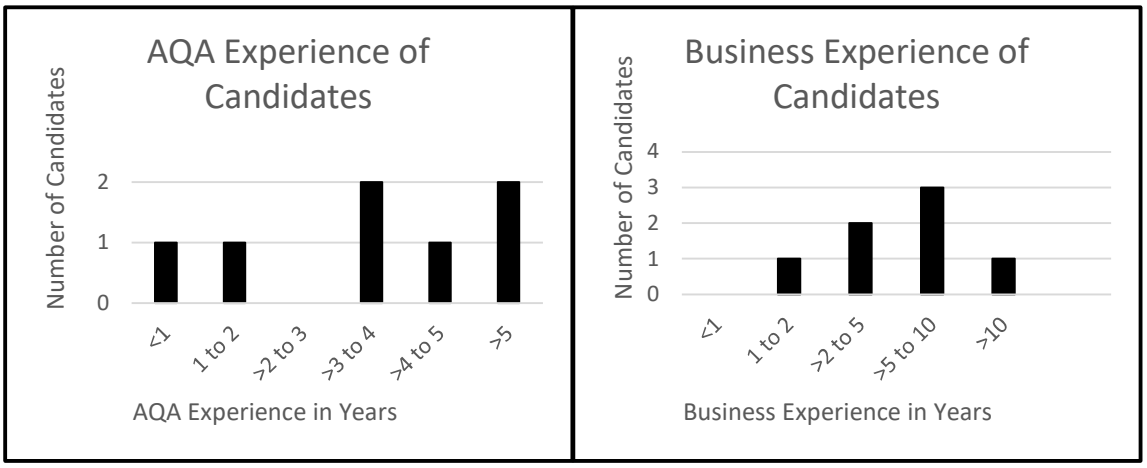


Figure 8: AQA Experience of Candidates
Figure 9: Business Experience of Candidates

Figure 10 shows that only 57% (four of seven) candidates work or worked in at least one real agile development project as per their own definition. Hence, they comply with the preferred level of experience. Another 29% of the candidates explain that they are at least familiar with pseudo-agile development projects and one candidate knows agile basically from theory. In contrast to that, 86% (six of seven) claim to have experience in conventional projects. One candidate knows conventional development only from theory, as displayed in Figure 11.

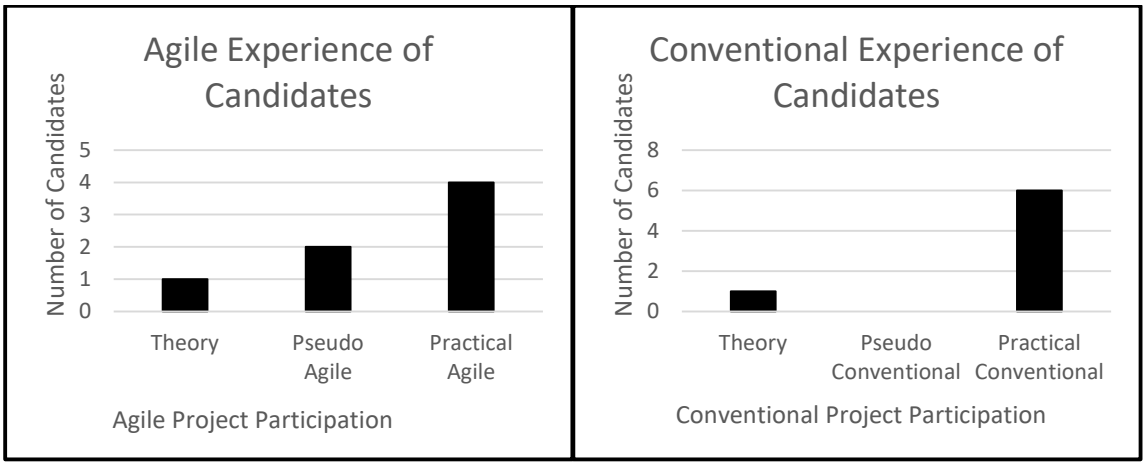


Figure 10: Agile Experience of Candidates
Figure 11: Conventional Experience of Candidates

To evaluate the candidate's expertise aside from AQA, Figure 12 points out how many different development perspectives (including AQA) the candidates professionally execute or executed. Every candidate knows at least one more development domain in addition to AQA. In Figure 13 the distribution of candidates per business area is shown. As it can be seen, a vast majority of the candidates works currently in the automotive business.

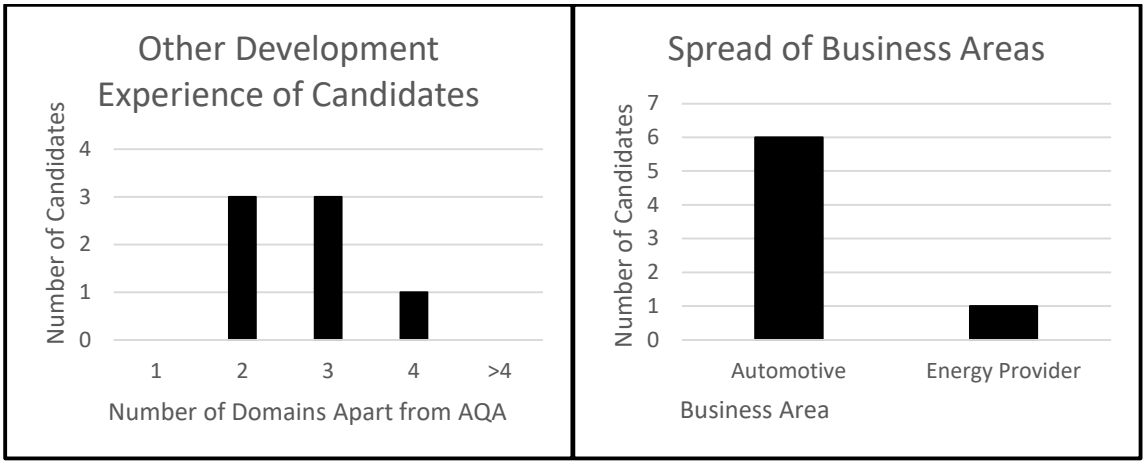


Figure 12: Development Domain Experience
Figure 13: Distribution of Business Areas

Conclusions and Concerns

When comparing the actual candidate composition with the initially postulated requirements, conclusions can be made about the expected quality of the interview results. Furthermore, specific concerns or constraints with regard to the interview interpretation can be derived:

- There is a satisfactory amount of AQA experience (Figure 8) and also cross-domain knowledge (Figure 12) to support good results.
- The background in agile development (Figure 10) is not optimal but sufficient. It seems that many candidates find themselves working in pseudo-agile projects. This should not be an issue with respect to the topic of this thesis, as especially those blended-style projects might be subject to challenges for AQA.
- The sense for conventional development (Figure 11) is very strong and absolutely sufficient.
- The distribution amongst different business areas is weak. There is a distinct dominance of automotive branch and consequently a dominance of embedded software over customer software businesses. This might influence the results, as embedded domains are traditionally less attached to agile development which

originated mainly from customer software branch. This could also be the root cause for the low level of real agile project experience in the candidate composition.

3.1.2 Structure of the Interview

The interview is structured in two parts in order to achieve the two stated goals. Confirming the derived hypotheses on one hand but also identifying new hypotheses on the other. The first part is an open interview which passes control over to the candidate. The second part is moderated by the interviewer and contains a fixed set of questions. Refer to Table 3 and 4 for further details on the separate parts of the interview.

Part One: Open Interview (Candidate Driven)

Objective	Mining of new, undiscovered challenges, issues, or hypotheses from the candidate without having influenced the candidate by prior interviewer questions.
Duration	Approximately thirty minutes
Methodology	After the organizational introduction, the interviewer passes control to the candidate. The candidate is requested to freely report about challenges and problems related to the combination of AQA and agile development that he or she faced in his or her career. If aspects are identified, the interviewer asks the candidate if the problem/challenge was solved and if so, which way.
Formalism	Solutions and aspects identified during the open interview are noted down by the interviewer briefly in the Statement Documentation Sheet (SDS). This SDS can be found in the appendix.

Table 3: Interview Guideline Part One - Open Interview

Part Two: Hypothesis based Interview (Interviewer Driven)

Objective	Receive confirmation or dissent on the derived hypotheses from Chapter 3, presented by the interviewer.
Duration	Approximately forty-five minutes
Methodology	Lead questions are defined based on the given hypotheses. The questions are presented by the interviewer and discussed with the candidate. The question catalogue can be found in the appendix under Interview Guide and Question Catalogue.
Formalism	Collected discussion results and potential solutions are also noted down by the interviewer briefly in the SDS.

Table 4: Interview Guideline Part Two - Hypothesis Based Interview

3.1.3 Execution Facts of the Interviews

Ultimately, from June to August 2022, seven interviews are executed with an average duration of one hour and twenty-seven minutes. Five of the seven interviews are held offline as personal meetings. The remaining two take place as online meetings in Jitsi, a tool provided by Technische Universität Kaiserslautern. Generally, all interviews are voice recorded and additional handwritten notes are taken with timestamps, to facilitate later analysis. For these notes the SDS is used to keep notes uniform and comparable. The SDS can be found in the appendix. Furthermore, before part one of the interview is started, the candidates are asked to answer some questions about relevant aspects of their own career and person. This information is documented in the Candidate Information Sheet which is also attached in the appendix.

3.2 Literature Research

Complementary to the expert interviews, literature research is performed to support and reflect the data received by the experts. As a further advantage, literature research unlocks more sources for potential solutions.

3.2.1 Search Criteria

To avoid unnecessary review effort, some high-level search criteria is established. The aspects should be met in order to rate the document as suitable for the desired research:

- The document should not be too old to avoid obsolete or outdated information. As hard limit, the document shouldn't be older than twenty years at the time this thesis is written.
- The document should be written in English or German to enable successful research.
- The document should not be too generic with respect to agile or AQA, a certain level of detail must be available.

3.2.2 Search Approach

The search for appropriate literature starts with a keyword search on Google Scholar. Utilized search strings are:

- [challenges; difficult; problematic] agile testing
- [analytical] quality assurance in agile projects
- [validation; V&V; validation] in agile
- agile vs traditional [testing; validation; analytical quality assurance]

As it turns out, the found literature is partly quite generic with regard to agile and AQA but from good generic literature another search technique can be spawned. A consecutive review is performed on the literature which is referenced in relevant sections of the generic literature. This approach is sketched in Figure 14. Eventually, the combination of both techniques reveals suitable documents for deeper analysis.

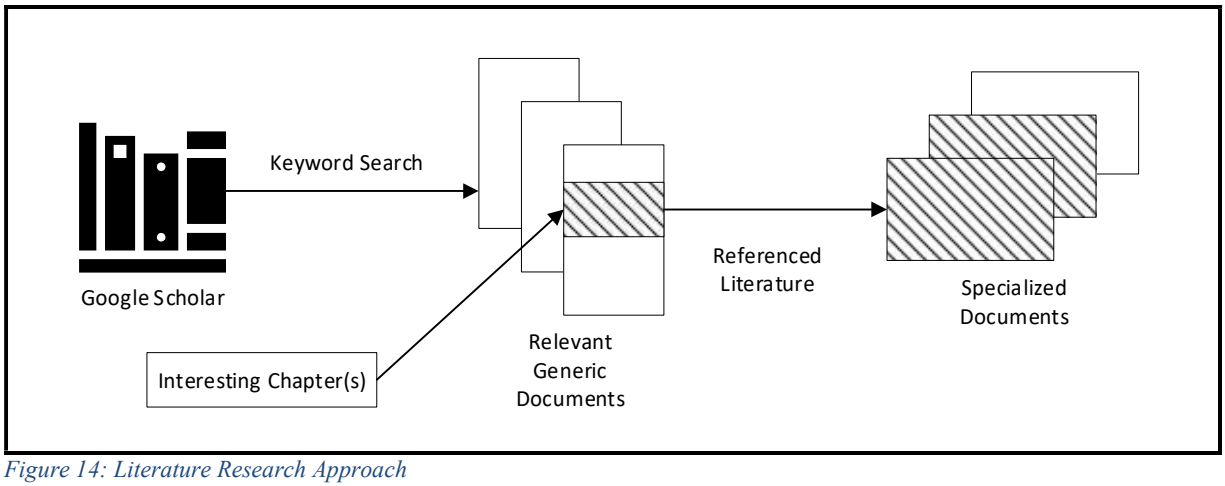


Figure 14: Literature Research Approach

3.2.3 Identified Literature

In June and July 2022, the following sources are identified by the described approach and screened for new hypotheses, solutions, and approval or dissent on the known hypotheses.

- Software Quality and Agile Methods
  (Hue, Verner, Zhu, & Babar, 2004)
- Towards Understanding Quality Assurance in Agile Software Development
  (Itkonen, Rautiainen, & Lassenius, 2005)
- How Not to Do Agile Testing
  (Puleio, 2006)
- Formal Versus Agile: Survival of the Fittest
  (Black, Boca, Bowen, Gorman, & Hinchey, 2009)
- Agiles Testmanagement
  (Sneed, 2013)

4 Data Analysis

This chapter is dedicated to the analysis of the elicited data. It is separated in two sections. The first section explains how the data is analyzed and provides the process framework to ensure good scientific practice. The second summarizes and presents an overview of the results.

4.1 Analysis Process

The interview audio captions, hand-written notes, and documents are analyzed candidate by candidate and document by document. The purpose of the analysis is to reduce the raw data to the essential core and bring it in an adequate form and order for later interpretation. For this reason, all raw data, independent from its source is examined using the following six step process, outlined in Figure 15.

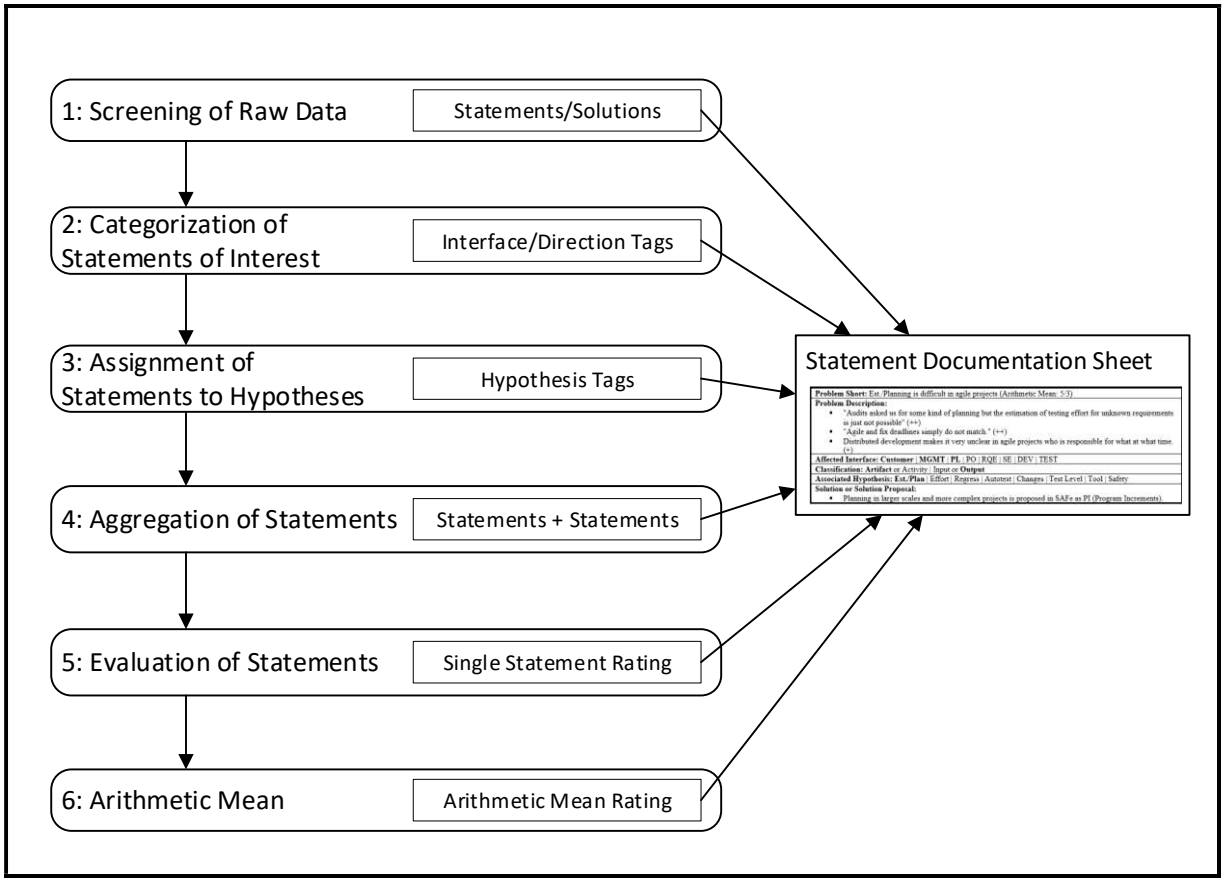


Figure 15: Analysis Process

All information that is extracted and gathered during these steps has to be documented in a homogenous and comparable format. In this case, the SDS that is used to document the hand-written notes during the interview already provides categorization means and is reused for findings from the literature research. Figure 16 shows a single SDS.

4.1.1 Step 1: Screening of Raw Data

First of all, key statements about problems and challenges are identified and extracted from the raw data by reading literature or listening to the audio captions. Whenever the source also mentions solutions, these are also extracted from the raw data and entered to the SDS. The summary and description of the extracted problem and solution proposal if applicable, is entered into the SDS as shown in Figure 16, marked in gray.

Problem Short: Est./Planning is difficult in agile projects
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible."
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 16: Example of SDS after Screening

4.1.2 Step 2: Categorization of Statements of Interest

As second step, the extracted problem is categorized to classify it and make it comparable with other problems. Three key questions are addressed by the categorization:

What interfaces are affected by this problem?

One or more interfaces between AQA and other domains can be affected. Interfaces shared with the following domains are considered by the SDS and can be tagged:

- Customer
- Management (MGMT)
- Project Leader (PL)
- Product Owner (PO)
- Requirement Engineering (RQE)
- System Engineering (SE)

- (Software) Development (DEV)
- Testing Internal (TEST)

Does the problem affect an activity or an artifact?

The examined statement can concern an activity, an artifact, or a combination of both.

Is this problem related to an input to or an output from AQA?

Furthermore, it can affect either an input to or an output from AQA. Depending on the direction, the aspect is root caused by AQA and causes issues in another domain (output) or the other way around (input). Figure 17 highlights in gray, how the results of the categorization are fed into the SDS.

Problem Short: Est./Planning is difficult in agile projects
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible."
Affected Interface: **Customer** | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: **Artifact** or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 17: Example of SDS after Categorization

4.1.3 Step 3: Assignment of Statements to Hypotheses

After the problem is categorized, the next step of the analysis is the assignment of the statement towards one or more hypotheses. If a relevant statement cannot be related to any of the existing hypotheses, the statement is tagged as new and a hypothesis is added. The following mapping as presented in Table 5 applies between the SDS and the naming in this thesis:

SDS Abbreviation	Related Hypothesis
Est./Plan	Hypothesis 1: Planning
Effort	Hypothesis 2: Test Effort
Regress	Hypothesis 3: Regression
Autotest	Hypothesis 4: Test Automation
Changes	Hypothesis 5: Changes
Test Level	Hypothesis 6: Test Levels
Tool	Hypothesis 7: Tooling
Safety	Hypothesis 8: Safety

Table 5: Mapping of SDS Abbreviations to Hypotheses

The assignment of hypotheses to a statement is demonstrated in Figure 18, marked in gray.

Problem Short: Est./Planning is difficult in agile projects
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible."
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 18: Example of SDS after Assignment

4.1.4 Step 4: Aggregation of Statements

The way the data analysis of this thesis is designed, each source is given the same weight when it comes to the single hypotheses. This approach requires every source to have either none or exactly one specified agreement level value for each hypothesis, independent from the number of singular statements it actually provides on the matter. As a preparation for the calculation of this arithmetic mean agreement level, the statements are aggregated in this step. If a source offers multiple statements that can be referred to one hypothesis, the singular statements are combined in one SDS. The possible outcome of this activity is shown in Figure 19, where the added statements are marked in gray for better visibility.

Problem Short: Est./Planning is difficult in agile projects
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible." ▪ "Agile and fix deadlines simply do not match." ▪ Distributed development makes it unclear in agile projects who is responsible for what, when.
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 19: Example of SDS after Aggregation

4.1.5 Step 5: Evaluation of Statements

As fifth step, the singular statements are evaluated in terms of agreement or disagreement level with the assigned hypothesis. This rating is the paramount step of the analysis and provides a crucial prerequisite for the evaluation of each single hypothesis. The five different levels of agreement or disagreement which can be assigned to a statement are displayed in Table 6. Each level also has a mathematical value related to it, which comes in play for the calculation of the arithmetic mean. The term "indirect" applies, whenever a statement does not directly match with the description provided by the author of this thesis but it describes a relatable matter that is either similar to or originates from the provided root cause.

Level of Agreement	Short	Value	Description
Direct Disagreement	(--)	-2	The statement directly addresses the issue of the hypothesis and disagrees.
Indirect Disagreement	(-)	-1	The statement indirectly addresses the issue of the hypothesis and disagrees.
Neutral Opinion	(0)	0	The statement neither agrees nor disagrees with the hypothesis.
Indirect Agreement	(+)	1	The statement indirectly addresses the issue of the hypothesis and agrees.
Direct Agreement	(++)	2	The statement directly addresses the issue of the hypothesis and agrees.

Table 6: Levels of Agreement

In early versions of the SDS, a rating of the problems was not yet considered and thus no dedicated fields are available. However, with proceeding analysis and growing need for support of evaluation, a simplistic rating system is added to the existing SDS. Each singular statement is evaluated and receives a rating as shown in Figure 20, highlighted gray.

Problem Short: Est./Planning is difficult in agile projects
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible." (++) ▪ "Agile and fix deadlines simply do not match." (++) ▪ Distributed development makes it unclear in agile projects who is responsible for what, when. (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 20: Example of SDS after Evaluation

4.1.6 Step 6: Arithmetic Mean

In the final step, the mean agreement level per source and hypothesis is calculated and added to the SDS, as shown in Figure 21. For this purpose, the ratings of the singular statements aggregated in one SDS are summarized and then divided by their number. It has to be mentioned that not every source necessarily provides an agreement level for all hypotheses. This can be explained by the different focus of the sources. It is quite relatable that not every candidate or literature has an opinion or valuable statement on every topic. However, in case the source offers statements for a specific hypothesis, the arithmetic mean is documented in the SDS as outlined in Figure 21, marked in gray.

Problem Short: Est./Planning is difficult in agile projects (Arithmetic Mean: 5/3)
Problem Description: ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible." (++) ▪ "Agile and fix deadlines simply do not match." (++) ▪ Distributed development makes it unclear in agile projects who is responsible for what, when. (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Figure 21: Example of SDS after Arithmetic Mean

To finalize the evaluation, the final agreement level for each hypothesis is calculated based on all available mean agreement levels of different sources using the following formula.

$$\boldsymbol{FinalAgreementLevel}(\boldsymbol{x}) = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{MeanAgreementLevel}(\boldsymbol{i}, \boldsymbol{x})$$

$$with: MeanAgreementLevel(i, x) = agreement\ level\ of\ source\ i\ on\ hypothesis\ x$$

$$and: N = \ number\ of\ sources\ contributing\ to\ hypothesis\ x$$

A last aspect that has to be taken into account is the confidence level for each final agreement level value. The confidence level indicates how many of the total of twelve sources are involved in the respective final agreement value. It is provided as percentage value. The higher the confidence level, the more reliable is the final agreement level for this hypothesis.

4.2 Analysis Results

Following the given procedure, a total number of seventy-eight aggregated statements is documented and analyzed. All aggregated statements are listed in SDS format in the section Extracted and Categorized Statements of the appendix, including their mean agreement level. The overall distribution between statements from interviews versus literature is calculated and reveals that interviews are the major origin of data with its fraction slightly exceeding 70%.

To display the summarized results, a heatmap is created. Table 7 resembles the heatmap and contains all relevant analysis outcomes. The mean agreement level values from each interview (IVx) and literature (LIx) as well as final agreement levels (FAL) and confidence level scores (CL) are shown. The color coding of the heatmap is explained in Figure 22 and Figure 23.

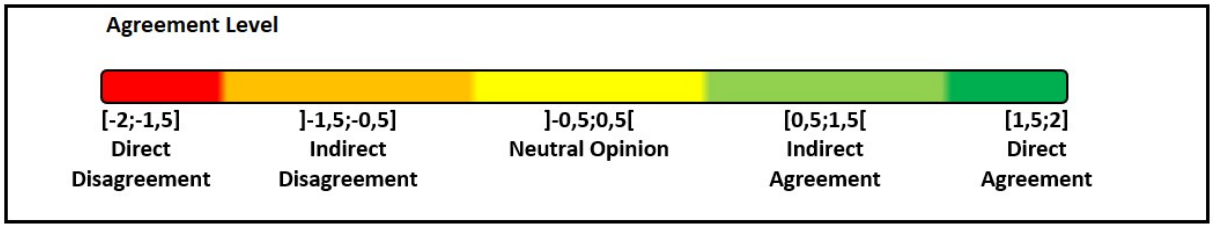


Figure 22: Agreement Level Color Coding

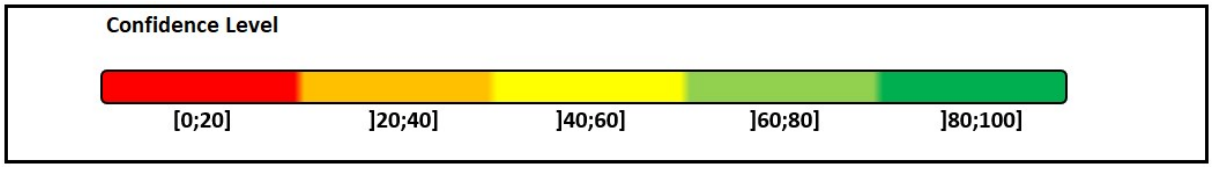


Figure 23: Confidence Level Color Coding

In order to provide scientifically reliable results, it finally has to be determined whether a hypothesis can be considered corroborated based on the results of the analyzed data, namely the combination of FAL and CL. For this reason, a calculation is performed that simulates the worst-case scenario, a maximum level of disagreement by all yet unconsidered sources, and returns the lowest possible FAL (LFAL). The calculation is based on the following formula:

$$\boldsymbol{LFAL} = \frac{\boldsymbol{FAL} * \boldsymbol{CL} + (\boldsymbol{-2}) * (\boldsymbol{100} - \boldsymbol{CL})}{\boldsymbol{100}}$$

Only if the LFAL is still positive, the hypothesis is considered corroborated. The results can be found in the last row of Table 7. Sufficient LFAL values are marked green, the others in red, meaning the related hypotheses cannot be reliably corroborated.

Source \ Hypothesis	1: Planning	2: Test Effort	3: Regression	4: Test Automation	5: Changes	6: Test Levels	7: Tooling	8: Safety	9: Variants	10: Trimmed Agile	11: AQA Timeout	12: Independent AQA	13: Professional AQA
IV1	N/A	0,67	-1,00	N/A	N/A	0,50	2,00	N/A	2,00	N/A	N/A	N/A	N/A
IV2	2,00	-0,33	1,50	1,67	0,00	1,75	1,67	-0,33	N/A	N/A	2,00	N/A	N/A
IV3	2,00	2,00	1,00	2,00	1,00	-2,00	2,00	N/A	N/A	2,00	N/A	N/A	N/A
IV4	1,50	-1,00	1,00	2,00	1,67	2,00	2,00	1,00	N/A	1,00	N/A	N/A	N/A
IV5	1,50	0,00	2,00	1,00	2,00	0,50	-0,50	-2,00	N/A	1,00	N/A	N/A	N/A
IV6	2,00	1,50	1,00	2,00	2,00	2,00	N/A	1,00	2,00	N/A	N/A	N/A	N/A
IV7	2,00	2,00	2,00	N/A	0,75	-0,50	0,33	-0,25	N/A	1,50	N/A	N/A	N/A
LI1	N/A	2,00	2,00	N/A	1,00	2,00	1,00	N/A	N/A	N/A	N/A	N/A	N/A
LI2	1,00	1,50	1,50	N/A	1,33	1,60	N/A	N/A	N/A	1,00	2,00	2,00	2,00
LI3	2,00	1,50	1,50	1,00	2,00	N/A	N/A	N/A	N/A	N/A	2,00	N/A	N/A
LI4	N/A	0,67	N/A	N/A	0,00	N/A	N/A	1,00	N/A	N/A	N/A	N/A	N/A
LI5	1,57	1,00	1,00	N/A	N/A	1,50	N/A	N/A	N/A	N/A	2,00	N/A	N/A
FAL	1,73	0,96	1,23	1,61	1,18	0,94	1,21	0,07	2,00	1,30	2,00	2,00	2,00
CL	75	100	91,7	50	83,3	83,3	58,3	50	16,7	41,7	33,3	8,3	8,3
LFAL	0,80	0,96	0,96	-0,20	0,65	0,45	-0,13	-0,97	-1,33	-0,62	-0,67	-1,67	-1,67

Table 7: Heatmap with Summarized Analysis Results

In addition to the original interview questions, the candidates are also asked about their personal attitude towards agile development. Applying the same evaluation criteria as explained in Figure 22, it turns out that most candidates comment the question predominantly neutral, when it comes to agile without references to specific projects. This is interesting, as it seems that many candidates theoretically see advantages in agile methods and at the same time the disadvantages prevail in most of the practical projects. The exact results of the question are shown in Figure 24.

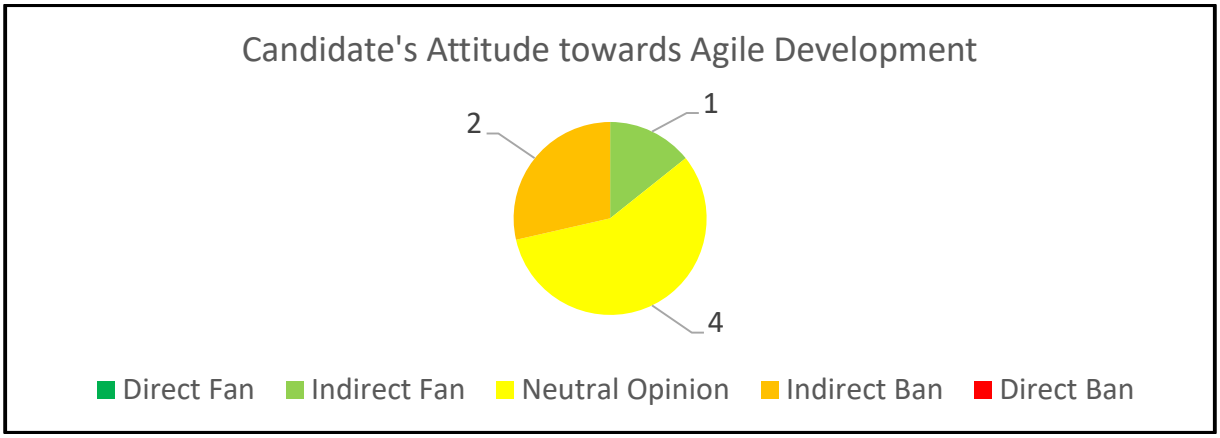


Figure 24: Candidate's Attitude on Agile Development

5 Data Interpretation

In this chapter, the analysis results presented by the previous chapter are applied to single hypotheses. Based on the data, individual decisions are made for each hypothesis, whether the data sufficiently corroborates it and if there is reasonable interest on this topic to justify the search for solutions.

The chapter is divided in two parts. The first part is concerned with the hypotheses provided by the author of this thesis in advance of the data elicitation. The second part presents and interprets the hypotheses subsequently derived from the mined data.

The scheme for interpreting the data is nevertheless the same for both parts and displayed in Figure 25. It is independent from the origin of the hypotheses and works as following: First, the hypothesis is repeated or, if the hypothesis is new, initially formulated. Second, the individual analysis results are displayed. Third, remarkable statements that resemble significant opinions on the topic are mentioned and last, the decision is made and discussed, whether the hypothesis is dropped or further considered.

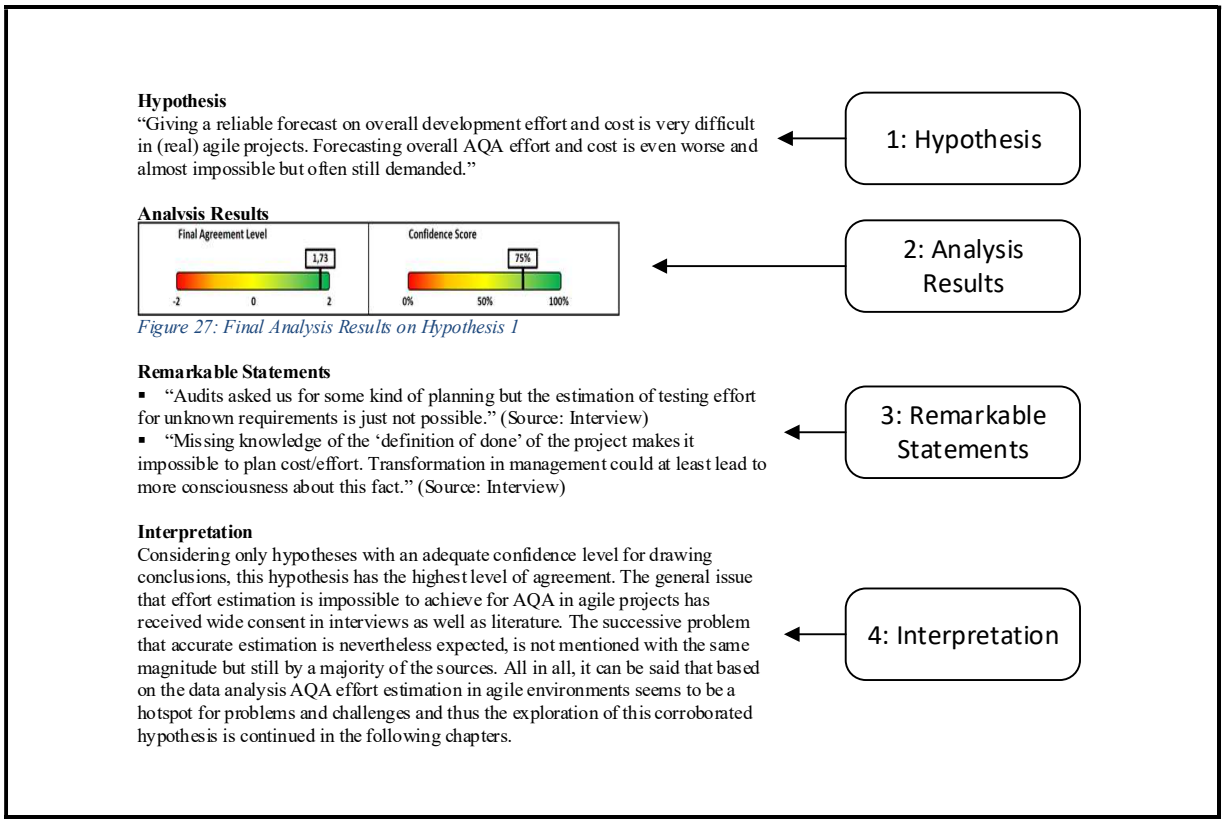


Figure 27: Final Analysis Results on Hypothesis 1

Figure 25: Interpretation Scheme

5.1 Existing Hypotheses

5.1.1 Hypothesis 1: Planning (Corroborated)

Hypothesis

"Giving a reliable forecast on overall development effort and cost is very difficult in (real) agile projects. Forecasting overall AQA effort and cost is even worse and almost impossible but often still demanded."

Analysis Results

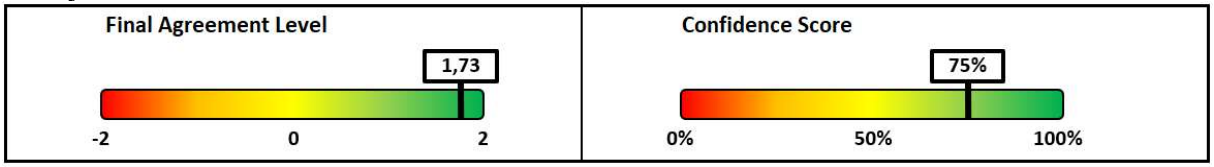


Figure 26: Final Analysis Results on Hypothesis 1

Remarkable Statements

- "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible" (Source: Interview).
- "Missing knowledge of the 'definition of done' of the project makes it impossible to plan cost/effort. Transformation in management could at least lead to more consciousness about this fact" (Source: Interview).

Interpretation

Considering only hypotheses with an adequate confidence level for drawing conclusions, this hypothesis has the highest level of agreement. The general issue that effort estimation is impossible to achieve for AQA in agile projects has received wide consent in interviews as well as literature. The successive problem that accurate estimation is nevertheless expected, is not mentioned with the same magnitude but still by a majority of the sources. All in all, it can be said that AQA effort estimation in agile environments seems to be a hotspot for problems and challenges and thus the exploration of this corroborated hypothesis is continued in the following chapter.

5.1.2 Hypothesis 2: Test Effort (Corroborated)

Hypothesis

"If a hypothetical project is developed both ways, conventional and agile, the effort and cost spend on AQA in the agile project is significantly higher as is the product quality."

Analysis Results

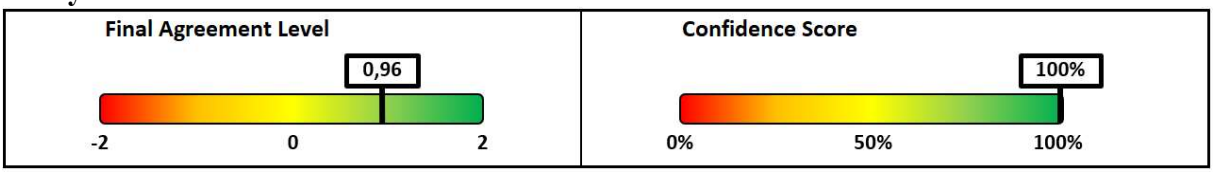


Figure 27: Final Analysis Results on Hypothesis 2

Remarkable Statements

- If agile is done correctly, the effort is not more and can even be equal or less (Source: Interview).
- If the same amount of effort is spent for testing in an agile and a conventional project, the test coverage in the agile project will be lower but the product will still be better from user perspective (Source: Interview).
- "[…] but we must be careful not to incorrectly assume that speed is of the essence with agile methods - nor should it be assumed that less effort is involved" (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p. 42).

Interpretation

This hypothesis has the highest possible confidence level, meaning that every source contributes to this topic, underlining the high relevance of the depicted topic. The agreement level of 0,96 resembles a general overweight of agreement but is has to be mentioned that three sources don't agree with this hypothesis. Still, looking at the bare figures, this hypothesis can be considered corroborated and is an eligible point of interest for further examination in terms of solution exploration.

5.1.3 Hypothesis 3: Regression (Corroborated)

Hypothesis

"Agile development potentially promotes regression which makes countermeasures necessary. Appropriate measures could be installed in CQA but mostly they are performed as AQA which is very ineffective and expensive."

Analysis Results

Figure 28: Final Analysis Results on Hypothesis 3

Remarkable Statements

- Generally, the amount of regression is higher in agile, also the retest number is higher (Source: Interview).
- "[…] but to take advantage of the agile development, every increment should be tested as complete as possible" (Source: Interview).
- "How can you run regression tests, new functional tests, security tests, etc. every sprint without increasing the test window from a week to three weeks" (Puleio, 2006, p. 6)?
- Regression should not be a matter for the AQA team (no regression tests) but for the developers (prevent regression by experience, excellence of work, architecture, …) (Source: Interview).

Interpretation

Not only the high contribution of 92% combined with an acceptance level of 1,23 makes this corroborated hypothesis a very good candidate for further analysis. Additionally, the interviews reveal that many different opinions on how to encounter this challenge exist. Some sources state the need to delegate original QA tasks from AQA to CQA and thus shift the responsibility for regression towards the development team. Other sources hold onto the necessity of higher level AQA even on sprint level and see the solution in form of automized acceptance testing. However, the fact that the solution of this problem is controversial, is a good indicator that more research might be required.

5.1.4 Hypothesis 4: Test Automation

Hypothesis

"Test automation is widely accepted as the only way for AQA to keep up with the fast-paced agile development but it also introduces new challenges itself."

Analysis Results

Figure 29: Final Analysis Results on Hypothesis 4

Remarkable Statements

- "For test automation it is better to not swim in the very fast flowing river of agile" (Source: Interview).
- Automated testing is not suitable for agile projects because the effort for maintaining the automated tests and the test environment is just too high in an agile project due to high number of changes (Source: Interview).
- Maintenance of automation is not possible if done by the same testers than testcase creation and testcase execution (Source: Interview).
- "The biggest area of contention the team encountered was testing. Neither the Product Owner nor the developers on the team understood the challenges and difficulty inherent in testing a complex software system in an automated manner" (Puleio, 2006, p. 2).

Interpretation

While there was remarkable feedback in the interviews on the challenge of test automation, the selected literature does almost not consider or mention this aspect. This leads to the confidence level of 50%. At the same time, the acceptance with the hypothesis amongst the interviewed candidates is astonishing and diverse risks and drawbacks of test automation in agile projects are revealed, as shown in the cited statements. This makes the hypothesis one of the most promising candidates for deeper examination even though it can scientifically not be considered corroborated.

5.1.5 Hypothesis 5: Changes (Corroborated)

Hypothesis

"The agile way on how to deal with changes causes a serious decrease of AQA efficiency due to an excessive amount of rework and duplication."

Analysis Results

Figure 30: Final Analysis Results on Hypothesis 5

Remarkable Statements

- "Documentation is worse than in conventional projects, intermittent and immature" (Source: Interview).
- "Agility demands that we should welcome changing requirements, even in late stages of development. Testing and static QA methods have traditionally been based on specifications that are completed in a certain phase of development and after that point can be used as a basis for test design and other QA activities. If those documents are allowed to change even in late phases of the development cycle, it clearly challenges the traditional way of doing QA" (Itkonen, Rautiainen, & Lassenius, 2005, p. 3).
- In agile, things are just changed without proper communication or feedback into stories or requirements due to time pressure. This leads to much useless work being done by testers due to false or obsolete information (Source: Interview).

Interpretation

The confidence and agreement level already make this hypothesis an interesting pick for investigation and the total absence of disagreement among all sources even further corroborates the hypothesis. Another factor that increases the relevance of this topic is the lack of tangible solution proposals and the fundamental influence it has on other aspects described in this thesis. Poorly communicated and documented changes in high frequency are root cause to successive issues. It appears that while many AQA experts suffer under the described problem, there is still no procedural solution at hand to properly handle change management in agile projects.

5.1.6 Hypothesis 6: Test Levels (Corroborated)

Hypothesis

"The effects of agile development on AQA are not evenly distributed or uniform. Some positive effects apply to activities on levels of lower abstraction, whereas levels with higher abstraction are to large extent confronted with challenges."

Analysis Results

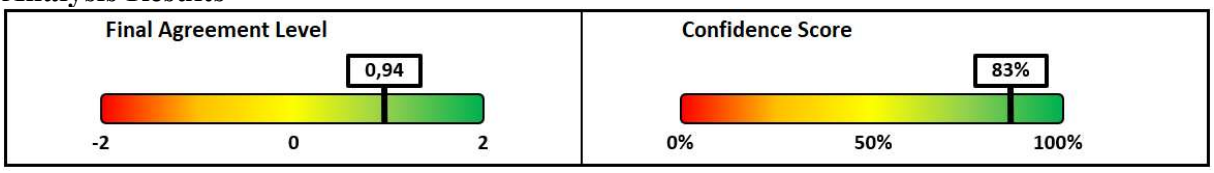


Figure 31: Final Analysis Results on Hypothesis 6

Remarkable Statements

- "[…] the methods include a set of good practices for developers, including automated unit testing. Still, only a few methods give any guidance for higher test levels than unit and integration testing" (Itkonen, Rautiainen, & Lassenius, 2005, p. 2).
- "[…] agile methods emphasise constructive quality building practice. Quality evaluating practices, based on a destructive attitude, are few, if any" (Itkonen, Rautiainen, & Lassenius, 2005, p. 2).
- "Shall I invest all this effort to make an end-to-end testcase, just to realize the system evolved away in the next sprint and the testcase is futile" (Source: Interview)?
- Upper right levels (V-model related) are more affected, because here all issues and changes potentize (Source: Interview).
- SW unit testing is a catastrophe in agile projects (Source: Interview).

Interpretation

One interview candidate significantly disagrees with this hypothesis but the agreement level of 0,94 based on a high confidence level of 83% still resembles wide acceptance and corroborates the hypothesis. While the assumption that agile development has a positive effect on the lower levels of QA is only supported by few sources, a dominant majority mentions negative implications on higher level AQA activities. These are mainly related to incompatible sub-system increments, release timing and potentized requirement changes. This hypothesis is certainly a good candidate for deeper analysis and solution space exploration.

5.1.7 Hypothesis 7: Tooling

Hypothesis

"If AQA is not operated in the same tool environment together with related domains, a lack of synchronization will be the result. The subsequent implications are especially problematic in agile projects and must be avoided."

Analysis Results

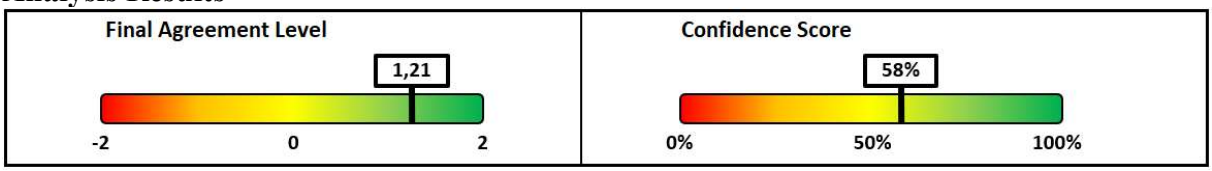


Figure 32: Final Analysis Results on Hypothesis 7

Remarkable Statements

- In agile projects or fake agile projects, it is often not clear what comes when. This is caused by insufficient communication and bad meeting habits but also insufficient tools or the use of multiple different tools (Source: Interview).
- Especially when the project is distributed among different teams, companies and maybe even countries, the testers are disconnected easily from the development team if multiple tools are used or the tools cannot enable proper collaboration (Source: Interview).
- As communication is much more important in agile projects, a good tool that supports reviews is also a necessary prerequisite (Source: Interview).
- Appropriate tooling is not crucial within the testing team but essential for coordination between testing and other domains (Source: Interview).

Interpretation

Even if the confidence and agreement level look promising, they are not sufficient to corroborate the hypothesis and thus it is not considered relevant for further investigation. Additionally, the lack of discrepancy between the single statements leads to the assumption that further work on this topic does probably not reveal any new findings. Instead, it seems that the importance of uniform tooling especially in agile environments is well known and does not need to be elaborated.

5.1.8 Hypothesis 8: Safety

Hypothesis

"The AQA related demands posed by domain specific safety standards/regulations are incompatible or at least difficult to achieve with agile projects."

Analysis Results

Figure 33: Final Analysis Results on Hypothesis 8

Remarkable Statements

- Agile can be compatible and even better if regulations change, due to a faster adaption to changes (Source: Interview).
- Even if agile is often too focused on function, rather than on processes and artifact quality, it will generally have a positive effect on safety (Source: Interview).
- Regulations like ISO 26262 were created based on V/waterfall models, maybe they have to be reshaped (Source: Interview).
- Agile should only be used for safety uncritical parts of software or in very early stages (Source: Interview).
- "Clearly, however, a possible cultural divide currently makes the agile and formal methods communities' incompatible collaborators"
  (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p. 44).

Interpretation

The elicited and analyzed data are not sufficient to corroborate the hypothesis. With only a very slight tendency of agreement based on a medium confidence level the topic is not further investigated and considered invalid.

5.2 New Hypotheses

5.2.1 Hypothesis 9: Variants

Hypothesis

"Variants of products are not considered in testcase creation. This leads to problems in AQA" (Source: Interview).

Analysis Results

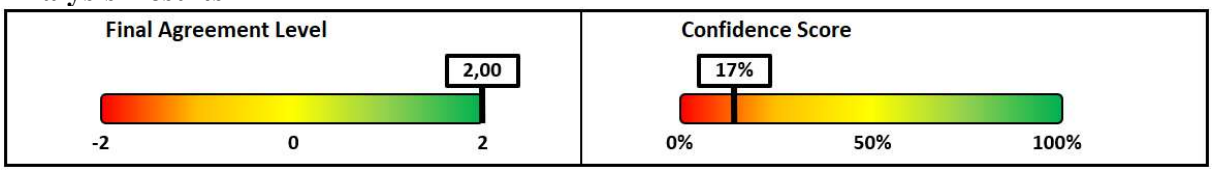


Figure 34: Final Analysis Results on Hypothesis 9

Remarkable Statements

- High number of variants poses an issue on development but even more on testing. A full coverage is almost impossible (Source: Interview).

Interpretation

This new aspect is derived based on the statements of two interview candidates. They mention that variants in terms of product line engineering pose challenges on AQA. While this general fact certainly holds true, the author of this thesis believes that this problem is not specifically caused by or limited to agile development projects. Also, the candidates do not mention agile explicitly in their statements. On the other side, it is still possible that agile is a factor that worsens the problems due to the high number and frequency of changes. Eventually, the confidence score is too low to corroborate or further consider this aspect within this thesis but it might be worth to analyze the effect of agile on product line AQA in a separate work that considers the problem right from the start.

5.2.2 Hypothesis 10: Trimmed Agile

Hypothesis

"The plan of the 'how' to has to be well setup before the project, the content can then be created agile way" (Source: Interview).

Analysis Results

Figure 35: Final Analysis Results on Hypothesis 10

Remarkable Statements

- The plan of the 'how' to has to be well setup before the project, the content can then be created agile way (Source: Interview).
- It is good to do larger style planning before sprints in the form of Program Increments as proposed by the Scaled Agile Framework (Source: Interview).
- "Take agile transformation serious, on organizational and personal level!" If this invest is not taken, the result of agile will be more conflicts, problems and bugs compared to conventional development. This could be improved by defining processes and tools ahead of the project and defining the project content sprint by sprint (Source: Interview).

Interpretation

For a newly added hypothesis, this topic receives a surprisingly high agreement level but an insufficient confidence level in order to be scientifically corroborated. It seems that the AQA experts identify a lack of agile transformation and project preparation as root cause for many successive issues that generally appear in agile projects and ultimately also affect the AQA domain. Some of the candidates demand at least a rough process framework and change management to be defined before a project starts while others go even further and request the whole scope to be planned upfront. The latter request, does not align with the principles of agile development and would trim them. In the strict sense, this topic does not describe a problem but mainly a solution. It is therefore indirectly considered in the next chapters as solution for other problems but not as a separate hypothesis.

5.2.3 Hypothesis 11: Testing Timeout

Hypothesis

"This [agile] is a challenge for testing because the rapid release cycle puts fixed deadlines for testing activities and does not allow extending the testing period if more defects are found than estimated" (Itkonen, Rautiainen, & Lassenius, 2005, p. 3).

Analysis Results

Figure 36: Final Analysis Results on Hypothesis 11

Remarkable Statements

- Software is delayed or has bad quality because development didn't estimate the scope right. This leads to time pressure on testing, too. The delay can sum up in the worst cases. Resulting effects are: Insufficient testcase creation, test execution, test results (Source: Interview).
- "Estimation is challenging. Most developers are quite poor at estimating work, usually under-estimating the time required to complete development of a set of features. This usually leaves little time for testing" (Puleio, 2006, p. 4).

Interpretation

This hypothesis is initially extracted from (Itkonen, Rautiainen, & Lassenius, 2005). While all of the related statements confirm the core of the hypothesis, some statements suggest a slightly different root cause for the problem. They state, the time pressure is caused by accumulated development delays rather than fix deadlines on the whole. Whatever the exact root cause for the time pressure in a particular project may be, this aspect obviously receives direct agreement by four of the twelve sources and scores an agreement level of 2,0 with a confidence level of 33%. Unfortunately, the confidence level is insufficient to fully corroborate the hypothesis, but as the core problem is very prominent, the author believes that the hypothesis is a point of interest for further investigations, nevertheless.

5.2.4 Hypothesis 12: Missing Independency

Hypothesis

"One of the fundamental principles of testing is independency. Myers states that programmers should avoid testing their own programs and that a programming organisation should not test its own programs" (Itkonen, Rautiainen, & Lassenius, 2005, p. 4).

Analysis Results

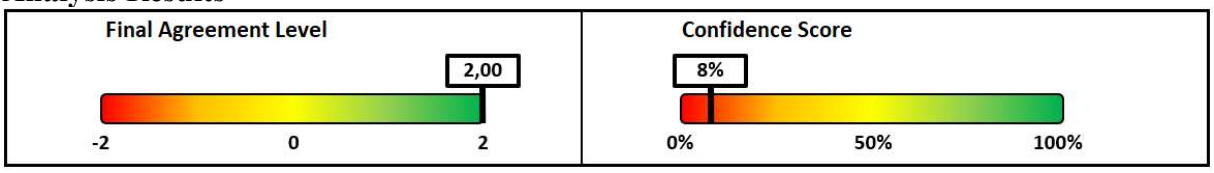


Figure 37: Final Analysis Results on Hypothesis 12

Remarkable Statements

- See stated hypothesis above.

Interpretation

Only mentioned by (Itkonen, Rautiainen, & Lassenius, 2005), this hypothesis describes a sound principle of testing that is often disobeyed by the agile approach - Independency of Testing.

Independent testers are necessary to detect systematic errors and to test the product unbiased. The shift from AQA to CQA in agile often leads to situations where developers carry out most QA tasks, even analytic ones. A solution could be to introduce dedicated tester roles even into developer teams or to apply practices like pair programming. Even though the low confidence level of 8% makes it impossible to scientifically corroborate the hypothesis based on the collected data, the author considers this problem to be valid and of high significance. At the same time, the principle of independent AQA is well-known and the discrepancy with agile projects probably already discussed in many other papers. For this reason, the hypothesis is not part of the solution assessment.

5.2.5 Hypothesis 13: Testing Expertise

Hypothesis

"Software testing is a creative and intellectually challenging task that requires a lot of specific skills and experience. It is a profession and requires a professional tester, in order to perform effectively and efficiently" (Itkonen, Rautiainen, & Lassenius, 2005, p. 4).

Analysis Results

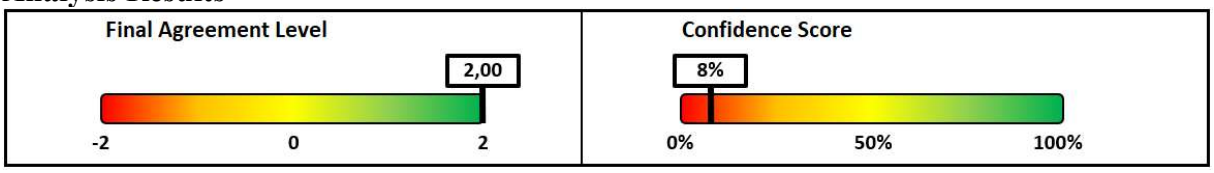


Figure 38: Final Analysis Results on Hypothesis 13

Remarkable Statements

- See stated hypothesis above.

Interpretation

Also introduced by (Itkonen, Rautiainen, & Lassenius, 2005), this hypothesis states that the agile approach largely disobeys another sound principle of testing - Professionalism of Testing. The principle calls for skilled, specialized, and educated testers because testing is "intellectually challenging" (Itkonen, Rautiainen, & Lassenius, 2005, p. 4). As resonance, multiple organizations like the International Software Testing Qualifications Board developed in the recent years, taking care of education and certification of testers. The author believes that the demand for such certificates further grows due to increasing complexity of systems combined with more safety critical applications.

The agile development approach does not favor this principle of professional testers but it also does not fully impede it. Agile development projects often assign QA tasks to developers, who are not specialized in the domain of AQA. A solution could be to either educate the developers on professional testing or to introduce separate, professional testers in agile projects, too. The author believes this topic is relevant but also easy to solve. Additionally, the confidence level is only 8% and as a result, this hypothesis is neither corroborated nor further considered.

6 Solution Assessment

This chapter deals with the presentation and evaluation of proven or potential solutions for the problems and challenges presented by the hypotheses with major relevance. The chapter is structured according the remaining hypotheses and their solutions. Each solution is put into context, rated, and eventually presented tabularly, including the description, number of sources, advantages and disadvantages, and last but not least a brief discussion.

6.1 Hypothesis 1: Planning

All in all, the examined solutions for this hypothesis are not revolutionary. No innovative procedural or sample solution is identified.

The intended ongoing evolution of estimates and plans, depicted in Table 8, is the favorized approach of the author. It should be common understanding that not only the product but also the plans alter in ongoing development projects, and even more, if it is done agile. Wrong estimates are normal, plans are dynamic. Crucial for the acceptance of this approach is a company-wide understanding of agile development based on a successful agile transformation.

Don’t try to estimate or plan agile in detail upfront but evolve the estimates and plans.
This solution was mentioned by one source.
⮝ A very rough estimate can be given based on early data. ⮝ The later in the project, the more accurate things can be estimated and planned.
⮟ Even rough estimates and plans tend to be taken for granted and might cause dispute when missed. ⮟ A lack of resources can cause severe problems when identified too late.
Generally, a very promising approach that should be common understanding in an evolutionary development. One aspect to enable this solution is a proper agile transformation to establish management understanding for superficial or wrong estimates in early project phases.

Table 8: Solution for Hypothesis 1 - Evolve Planning

An interesting approach for certain companies might be the backshift from real evolutionary development towards incremental development, as presented in Table 9 and Hypothesis 10. This can be helpful in domains, where requirements are usually static or most work is done as contractual work by suppliers, demanding a solid and reliable definition of scope. In many

scenarios agile is neither the best option nor required and affected companies have to be aware of this.

Trim agile away from evolutionary to incremental development by fixating the requirements upfront and only structure the development in iterations.
This solution was mentioned by three sources.
⮝ Early knowledge about the definition of done and the features to be implemented makes estimation and planning more reliable.
⮟ The solution decreases the flexibility of the development drastically, even though it is a key advantage that agile approaches aim to achieve. ⮟ Ultimately, it can no longer be spoken of agile development.
The advantages of limiting the flexibility are tremendous and even though many companies claim to develop agile, it has to be assumed that in most cases this solution is already at place, providing an acceptable compromise between flexibility and plannability.

Table 9: Solution for Hypothesis 1 - Trimmed Agile

Joint, experience-based estimation techniques, such as Wideband Delphi, are already best practice in any development model. If not yet introduced, a well-attuned blend of these solutions as presented in Table 10 and 11 can probably improve the situation in any company.

Base estimations and planning on practical experience of experts and data from previous projects.
This solution was mentioned by three sources.
⮝ Providing estimations based on experience is probably the best approach in any development model and has proven to be useful in the past. ⮝ Including multiple experts and the data of previous agile projects enables data-driven decisions which are justifiable.
⮟ Different projects are not necessarily comparable in terms of effort. ⮟ Essential factors that influence estimations might not be reproducible, such as staffing, experience of employees, and customer.
Estimations and planning should always be left to experts, independent from the development model. Especially in agile projects, experts play an even more important role. It is advisable to document derived estimates and actual results for future projects.

Table 10: Solution for Hypothesis 1 - Involve Experts

Planning and estimation are done jointly by all effected domains and members.
This solution was mentioned by two sources.
⮝ Estimations and planning are carried by the whole team and everyone takes responsibility to meet them. This can enhance the team spirit. ⮝ Plans are visible to all involved parties right from the start which improves the team communication and understanding generally.
⮟ Involving more people in these tasks is more expensive. ⮟ Aligning on meetings and opinions becomes harder with more participants. ⮟ The joint estimations might be root cause to conflicts between the team members.
While there are also some disadvantages, it can generally be assumed that the advantages overweight. However, the efficiency of the solution probably varies with the size of the project and gets increasingly complex with larger teams. It might be worth for any company to try joint planning and estimation and see where it goes.

Table 11: Solution for Hypothesis 1 - Plan Jointly

The idea to extend given estimates by large safety factors as proposed in Table 12 is not very appealing. On the one hand, the advantage of reducing the number of potential bad surprises is clear. Scaling down an overstaffed project is easy while accelerating a late project by scaling up is not. But on the other hand, the price is literally too high. The company might lose business to competitors or cancel promising projects for low profitability.

Add large safety factors to estimations to account for uncertainties.
This solution was mentioned by one source.
⮝ Largely extended estimations and plans decrease the risk for underestimation. ⮝ Overestimation is less risky in terms of project failure. It is always easier to reduce teams later rather than recovering from delay.
⮟ If estimates are unrealistically high, projects might not be started due to an insufficient return on invest. ⮟ As a supplier it is often essential to quote with competitive prices which is almost impossible if large safety factors apply. The project might be lost to a competitor.
It is comfortable to overestimate effort and cost at the beginning of projects and scale it to reasonable numbers later but the disadvantages are not neglectable. In the worst case, whole projects will not be started or sourced due to unacceptably high cost and duration.

Table 12: Solution for Hypothesis 1 - Excessive Safety Factors

6.2 Hypothesis 2: Test Effort

The most efficient and obvious way to reduce the effort spent on AQA is to simply reduce the test coverage. Two different solution proposals are based on this basic approach. One idea, presented in Table 13, is to limit the test coverage in early sprints and expand it with progressing product maturity. In the opinion of the author this is a promising solution due to additional positive side effects and acceptable constraints. The other idea, explained in Table 14, is to generally refrain from using agile whenever high test coverage is required and limit the AQA activities to an acceptable minimum.

Reduce the test coverage in early sprints, test in more detail later.
This solution was mentioned by one source.
⮝ Less overall effort spent in testing. ⮝ Time saved in the early project phase can be spend on other tasks. ⮝ Reduced number of created bug tickets caused by incomplete features.
⮟ Test feedback for early sprints is not very detailed. ⮟ Some fundamental errors might reside in the product until later sprints. ⮟ Less hands-on experience with the product at early project phase. ⮟ Difficult to catch up with development later.
The idea is quite appealing from AQA point of view. It does not only decrease the overall test effort but at the same time the potential redundant work, caused by AQA activities with immature products. Additionally, the number of false bug tickets created against de facto incomplete features drops. One major counterpoint is the risk of AQA not being able to catch up with development progress in the final development phase.

Table 13: Solution for Hypothesis 2 - Reduced Early Coverage

Use agile only for projects if low test coverage is sufficient, such as A sample grade or safety uncritical products, and resign from extensive AQA.
This solution was mentioned by one source.
⮝ Testing and test effort can be reduced to a reasonable amount. ⮝ Project cost is lower while product flexibility is still high.
⮟ Even in uncritical projects a decent product quality might be desired.
If it is eligible for product, project, and customer, this can be a very practical and easy to achieve solution that drastically lowers the AQA effort. In the strict sense of agile, the customer is anyway the most important tester and separate AQA not necessarily required.

Table 14: Solution for Hypothesis 2 - Uncritical Low Coverage

Another solution with great potential is to improve the product quality by engineering excellence in the development team, as explained in Table 15. Consequently, this reduces AQA effort by better testability and less regression. The proposed solution is theoretically the best course of action because it primarily improves the product and indirectly helps the AQA

team. Unfortunately, it is quite difficult to realize in practice. In fact, this solution shifts effort from AQA to CQA, where time is usually a scarce resource.

Reduce overall AQA effort by engineering excellence in the development team.
This solution was mentioned by one source.
▲ A well-designed product is easier to test and sponsors less regression. ▲ Certain aspects can be proven by design and need less validation. ▲ Less bug tickets have to be created due to better software quality.
▼ Experienced developers are a cornerstone of this concept but a scarce resource. ▼ Time pressure keeps developers from finding ideal solutions.
The interdependency is obvious. A decent product is not only desirable for the customer but also for AQA. Many factors can have a positive effect on the efficiency of testing, such as rigorous documentation, well-defined architectures, high cohesion, and low coupling. In the opinion of the author, this approach has high potential, as it is the intend of agile to increase the quality by CQA rather than validating it by AQA. Unfortunately, it usually fails due time pressure in the development team.

Table 15: Solution for Hypothesis 2 - Better Software by Excellence

In the eye of the author, the popular approach to reduce AQA effort by test automation, as depicted in Table 16, is a failure. There are unchallengeable positive aspects about test automation in agile and it should be considered on lower levels of abstraction and even on functional and acceptance test level. But to assume, the overall effort can be reduced by it, is untenable. The initial and maintenance effort for test automation is tremendous, especially for flexible, agile products.

Use test automation to reduce (manual) test effort.
This solution was mentioned by three sources.
▲ Tests can be executed with minimal manual interference or support. ▲ The tests deliver results much faster when an increment is released. ▲ Test frequency can be adapted even to daily development builds.
▼ Test automation needs a specialized execution environment. ▼ Test automation especially on higher levels needs a stable product or lots of rework. ▼ The human sense is lost in the tests, only binary pass and fail criteria is possible.
The experts and literature jointly consent on test automation as a key factor for agile testing. The author of this thesis agrees that automated testing on certain levels is beneficial for agile projects but the negative aspects of test automation cannot be neglected. The maintenance effort for automated tests with higher level of abstraction can easily supersede the overall effort it would have taken to execute these tests manually.

Table 16: Solution for Hypothesis 2 - Test Automation

6.3 Hypothesis 3: Regression

Each of the three solution proposals inherits drawbacks and chances. It seems that a sophisticated look on the company, the project, and the product is required to determine which solutions suits the specific needs best.

A sophisticated test automation as proposed in Table 16 for Hypothesis 2 can cover the tests for legacy scope in every sprint and yields the highest certainty in terms of uncovering regression, while added features are tested manually. This way, the risk for released regression is minimized but the required invest must not be underestimated. Another drawback is the binary character of these tests. Factors such as usability, look and feel, or feedback performance cannot be validated.

The theoretical approach of building regression free software offers less certainty with regard to regression detection. Even with good documentation, architectures, and engineering discipline, regression well be introduced to the product. However, one major bonus of the solution presented in Table 17 are the positive side effects on the quality of the actual product, such as improved maintainability and testability. In order to establish an adequate CQA, large investments have to be made.

Avoid regression by engineering excellence in the development team.
This solution was mentioned by three sources.
⮝ Avoid regression where it is introduced instead of locating it afterwards. ⮝ The product's long-term maintainability is improved. ⮝ The product comes with a higher built-in quality.
⮟ Impossible to guarantee freedom from regression, a risk for regression resides. ⮟ Remaining regression might be released into the market undetected. ⮟ High effort is required in development team for appropriate CQA activities.
Also known from Hypothesis 2, this theoretical approach is very appealing but unfortunately also not very common in practice. It means a shift of responsibility and effort from AQA to CQA but the developers are usually too short in time for optimal designs.

Table 17: Solution for Hypothesis 3 - Excellence in Engineering

Compared to the previous ones, the solution presented in Table 18 is very pricy but also yields a harsh risk of regression being released to the market. Considering regression only by smoke tests for legacy features might be an option for products, where hot fixes can be rolled out instantaneously but most embedded companies are probably not willing to take the risk.

Reduce test coverage in regression tests to a smoke level.
This solution was mentioned by one source.
▲ The AQA effort is kept low. ▲ Focus can be shifted towards new features from the current sprint.
▼ High risk of undetected regression in the released product. ▼ Regression might be detected late in the project, causing more expensive fixes.
This solution demands to limit the regression tests to a smoke test level. Only superficial tests are executed for legacy features and a strong focus is put on new content. Putting the focus of AQA activities on newly introduced features is common practice but it bears a level of remaining risk that might be inacceptable for most companies.

Table 18: Solution for Hypothesis 3 - Smoke Regression Tests

6.4 Hypothesis 4: Test Automation

Some of the provided proposals resemble best practices for test automation in any development model, still it is questionable if they are equally applicable and relevant to agile projects. Two solutions aim at improving the situation particularly for agile developments.

Automated testcase creation as proposed in Table 19 is most promising, if the prerequisites are available. The topic receives growing attention in recent years. A number of tools and languages is meanwhile available that offers requirement based testcase generation as core features, such as Squish or Gherkin.

Take automation to the next level and use automated testcase creation.
This solution was mentioned by one source.
▲ Testcases can be automatically created and adapted based on requirements. ▲ Requirement changes can automatically be translated to testcase modifications. ▲ Massively reduces effort for test automation and gives room for other AQA tasks.
▼ Very rigorous, formal requirement specifications or models are required that usually don't exist in agile. ▼ The creation of very precise requirements and models might cause additional effort. ▼ A suitable formal language has to be selected and trained in order to enable this approach.
This solution is very promising for projects with a strong focus on model-based development and rigorous specifications. If the prerequisites are set up anyway, the idea of deriving testcases from these inputs automatically yields savings in maintenance effort.

Table 19: Solution for Hypothesis 4 - Automated Testcase Creation

Another favorable approach, closely related to the previous one and outlined in Table 20, is to put the automation focus only on testcases that are sufficiently well-written and detailed. Not only does this significantly reduce redundant work but also improve the overall product quality by early identification of requirement gaps and errors. Generally, it seems that better documentation is beneficial for AQA but at the same time it conflicts with the agile principles.

Only rigorously described and manually executed testcases with high quality are considered for automation.
This solution was mentioned by one source.
▲ If only well-written testcases are automated a lot of redundant effort is avoided. ▲ This approach encourages rigorous testcase description, allowing for identifying specification and requirement errors as a side effect. ▲ A single manual execution can validate the testcase itself and reveal errors before the time-consuming automation of the test is started.
▼ The majority of product features might be left for manual testing because of insufficient testcase maturity. ▼ Often rigorous testcases cannot be written due to lack of input documentation.
When this solution is implemented, it will split effort between AQA and requirement engineering domain. Additionally, putting a focus on rigorous testcase and requirement documentation will not only support test automation due to reduced amount of redundant work but also reveal specification gaps and errors early.

Table 20: Solution for Hypothesis 4 - Only Rigorous Testcases

The idea from Table 21, to exclude test automation from the project and run it as a separate team seems distracting at the first place but two experts describe great improvements after test automation is decoupled from the fast-paced sprint rhythm. In order to conclude, further research is necessary, comparing external test automation with test automation run by the actual project team.

A separate, project external team of specialists is taking care of test automation.
This solution was mentioned by two sources.
▲ Specialists are able to create a better test automation more efficiently. ▲ Reasonable, technical facts rather than project plans drive the automation progress. ▲ The complex process of test automation is detached from the sprint frequency.
▼ The specialized automation team lacks project specific knowledge that has to be transferred, causing a communication overhead. ▼ Huge manual test scope might be left over for the AQA team of the project. ▼ The focus of test automation might deviate from actual implementation goals.
This approach is based on the practical experience of two experts. Both state that being external to the project teams is a relief. Their automation teams concentrate on the automation of technically feasible aspects instead of being constraint by the rhythm and content of the sprints. As drawback, this might lead to situations where projects will not reach the level of automation that is intended by the project team.

Table 21: Solution for Hypothesis 4 - External Test Automation Team

Probably the most practical and yet effective approach is to dedicate full sprints to AQA activities, whenever it is necessary to reconnect the development and test automation progress. This concept is proposed in Table 22 and directly helps to keep all teams on track. In many cases, there will be sufficient clean-up tasks to keep the developers busy during a test

sprint. If the test automation is additionally defined as a joint project goal as depicted in Table 23, the collaboration can be enhanced directly.

Introduce dedicated test sprints, when the test automation begins to lose track.
This solution was mentioned by one source.
⮝ Keeping the product stable for some sprints can help AQA to catch up and improve the coverage of test automation drastically.
⮟ Causes a general delay of the project. ⮟ The dedicated test sprint might put development in a standby state.
Practically, this happens almost incidental at the end of many agile projects. The development is factually completed but the test debt is too high to immediately release the product. Possibly, the introduction of test sprints in earlier iterations can help to keep up with the development and bring test automation and product back together.

Table 22: Solution for Hypothesis 4 - Dedicated Test Sprints

Preparing the automation framework ahead of the project as proposed in Table 23 might sound like a good idea but it is only feasible if sufficient knowledge about the course of the project and product are known in advance. Tools and specialists are very expensive and the risk of wasting resources for preparing a framework that doesn't match the implementation is high.

Test automation is introduced before project start and embedded in the project goals.
This solution was mentioned by one source.
⮝ If the automation framework is created before the project starts, a lot of time can be saved for actual test automation during the project. ⮝ Embedding test automation into project goals, provides a better drive and commitment from the whole project team in favor of test automation.
⮟ Early decisions and frameworks bear the risk of not meeting the requirements later. ⮟ In exceptional situations the productive project goals might suffer under the additional goal of test automation.
It appears convenient to prepare the test automation framework in advance of the project but the author believes this can only work if experience in test automation for similar products already exists. Otherwise, the risk of wasting resources on potentially useless endeavors is not acceptable. The additionally mentioned promotion of test automation as a project goal could be a good way to fixate the goal jointly for the whole project team.

Table 23: Solution for Hypothesis 4 - Test Automation as Goal

6.5 Hypothesis 5: Changes

Changes are the root cause for many of the discovered challenges and all proposed solutions can contribute to enhance the situation. Three of them aim at modifying the agile project framework in order to make it more sustainable for AQA while one alters the AQA approach to better fit in the agile framework.

The solution which tries to solve the sketched problem the other way around is presented in Table 24. Instead of trimming agile to fit AQA, it alters AQA to fit agile. All companies should overcome their objections and try to move from specification-based testing to explorative testing. It yields many advantages, especially for agile projects. In fact, explorative testing could be the missing piece to make AQA compatible to agile on broad front. It resembles the core principles of agile – but for AQA rather than development.

Use explorative testing instead of inflexible, specification-based tests.
This solution was mentioned by one source.
▲ The testing is very flexible and can perfectly adapt to changes in the product. ▲ Reduced amount of preparation and documentation for AQA. ▲ The concept of explorative testing inherits the spirit of the agile approach.
▼ No definition of done possible and essential aspects might be missed by AQA. ▼ Executed tests are not easy to repeat due to lack of procedural documentation. ▼ Explorative AQA is not sufficient for safety critical applications.
This is the most conforming solution in terms of agile development. Explorative testing incorporates many agile principles itself, enhancing individuals over processes and documentation. It seems very appealing to facilitate explorative testing in agile projects whenever the context and product allows it.

Table 24: Solution for Hypothesis 5 - Explorative Testing

As Table 25 shows, five sources demand to introduce change managers and management. The author of this thesis strongly agrees with that request. A limiting, controlling factor is needed to bring order to the change chaos and make coexistence of AQA and sprint teams feasible. Unfortunately, this might lower the flexibility. The Three-Field Agile approach, referenced by Table 26 claims to find the balance between control and flexibility by proposing an extended agile processes framework.

Install a change manager and well-defined change management processes (upfront).
This solution was mentioned by five sources.
▲ Dedicated change manager raises awareness for impact of changes. ▲ The process reduces the number of changes to the inevitable minimum. ▲ Crosscutting communication about changes with all affected parties.
▼ Change manager is an additional role that needs to be paid. ▼ Change managers and management controvert the core aspects of agile. ▼ The development loses flexibility.
Mentioned by five sources, this is one of the most prominent solutions. Change management will bring order to the chaos and make changes tangible. The price has to be paid in the form of decreased flexibility. Many consider change management as a burden but it can be seen as a positive threshold, too. It prevents unnecessary changes for the sake of the crucial ones. Additionally, having clear processes and responsibilities to treat changes is very helpful for all domains.

Table 25: Solution for Hypothesis 5 - Change Management

Use the Three-Field Agile approach to enforce change processes.
This solution is proposed by the author of this thesis.
▲ Intermittent changes during sprints are prevented to stabilize increments. ▲ Changes are mandatorily considered and treated as regular backlog items. ▲ Clear communication paths, responsibilities, and interfaces with regard to changes.
▼ Poses constraints on the flexibility of the development. ▼ Extends the reaction time to incoming change requests.
This solution can be seen as an instantiation of a specific change management process, modified to fit agile development projects. It introduces principles on how to deal with changes in agile and tries to confine as little as possible and as much as necessary. The approach is described in more detail at the end of Chapter 6.

Table 26: Solution for Hypothesis 5 - Three-Field Agile

Using a joint tooling that supports traceability and change management as outlined in Table 27 is beneficial in any model and in the author's opinion the benefit for agile projects is even higher. Expensive tools will pay off after some time due to a reduced communication overhead and smoother interdisciplinary collaboration; especially, when it comes to changes.

Provide a joint tooling that supports rigorous traceability and change management.
This solution was mentioned by two sources.
▲ Changes are automatically visible at all affected artifacts. ▲ Changes can be thoroughly documented and decisions captured. ▲ Reduced communication overhead due to tool-based information sharing.
▼ Tools that support full product-lifecycles tend to be more expensive. ▼ Each tool will not be ideal for any domain, tradeoffs will be necessary. ▼ Time has to be spent on trainings when a domain team changes to another tool.
It might seem difficult to find a tool that complies to the needs of all domains but once it is found and used, it significantly improves the situation for all domains. The advantages of a joint tool with rigorous traceability go far beyond change management.

Table 27: Solution for Hypothesis 5 - Joint Tooling

6.6 Hypothesis 6: Test Levels

The description of the Dynamic Systems Development Model (DSDM) in terms of system increments sounds very promising and the fact that DSDM is a ready-to-use model renders it even more suitable. Furthermore, it is mentioned that "DSDM is now prospering as agile has come of age. This is due to organisations needing more rigour and control when running agile in situations where the 'ideal agile environment' doesn't really exist" (Richards, 2022). DSDM is shown in Table 28 and addresses the challenges mentioned by Hypothesis 8, too.

Facilitate testable system increments by concerted development, based on the Dynamic Systems Development Method (DSDM).
This solution was mentioned by one source.
▲ DSDM is a documented and ready-to-use model that evolved over years. ▲ AQA situation improves by more sophisticated and aligned system increments.
▼ DSDM is not evolutionary but incremental by definition.
One of the few explicitly mentioned proposals is DSDM. It is supposed to address the challenge of coordinatively building system increments even in distributed development teams. The author has no professional experience with DSDM and a detailed analysis exceeds the scope of this thesis but on his website, Richards describes DSDM as "an agile method that addresses the needs of […] more complex projects involving multiple distributed teams working in different countries and time zones. DSDM can therefore scale up to address multiple teams working in distributed environments, something that scaled agile frameworks only recently started to offer" (Richards, 2022).

Table 28: Solution for Hypothesis 6 - DSDM

The approach of explorative testing was already mentioned as solution for Hypothesis 5, as a possible answer to the high change frequency in agile. In this context, explorative testing is of particular interest for higher level AQA, such as functional system tests or acceptance tests, because here all changes accumulate and potentize. As Table 29 points out, explorative testing is closer to the way an actual user approaches the system and might reveal design flaws which would remain undiscovered if specification-based testing was used.

Use explorative testing for higher level and acceptance tests.
This solution was mentioned by one source.
▲ "Exploratory testing is effective for testing applications from the viewpoint of the end-user" (Itkonen, Rautiainen, & Lassenius, 2005, p. 7). ▲ "The nature of exploratory testing helps reach the destructive attitude required in effective testing" (Itkonen, Rautiainen, & Lassenius, 2005, p. 7). ▲ Refer to Table 24 for further positive aspects.
▼ Refer to Table 24 for negative aspects.
Refer to Table 24 for a discussion on explorative testing.

Table 29: Solution for Hypothesis 6 - Explorative Testing

The separation of AQA in dedicated teams and sprints as proposed by Table 30, can be considered a backshift to conventional development. The task of system validation is withdrawn from the iterations and added as a separate, subsequent phase after the iterative development. Furthermore, it is removed from the sprint team and reassigned to a team of specialists. It has to be clear, that this solution is a step away from agile development. Still, it is a valid solution to decrease the redundant effort spent on specification-based testing for unfinished, unstable system increments.

AQA for product releases and higher test levels is outsourced to separate teams and dedicated stabilization sprints at the end of a project to overcome challenges.
This solution was mentioned by two sources.
▲ No redundant effort is spent on AQA for unstable system increments. ▲ Backs the fundamental AQA principles of independency and professionalism.
▼ Contradicts the agile principle of continuous delivery from sprint to sprint. ▼ System errors and architectural problems are discovered late and fix costs are high. ▼ Communication overhead caused by information transfer to separate team.
The proposal is closely related to the solutions presented in Table 21 and Table 22. It appears that higher level AQA generally suffers under the effects of fast-paced development and high change frequency - just as test automation. The question is whether it is desirable to cut AQA cost by testing the system only subsequent to development and accept higher expenses in late bug fixing in return.

Table 30: Solution for Hypothesis 6 - External Test Teams

6.7 Hypothesis 11: Testing Timeout

The author is in favor of the introduction of dedicated testing sprints whenever reasonable, as proposed by Table 31. It is a straightforward approach. First some test debt is accumulated and then an appropriate period of time is granted to AQA for catching up again. This way, decisions can be made based on the actual situation instead of forecasts.

Introduce dedicated testing sprints to reduce accumulated test debt.
This solution was mentioned by one source.
⮝ AQA team can get back on track and catch up with development progress.
⮟ Refer to Table 22 for negative aspects.
The discussion from Table 22 also applies here but in this case the dedicated sprint is used to reduce general test debt, instead of considering test automation debt exclusively.

Table 31: Solution for Hypothesis 11 - Dedicated Test Sprints

In contrast to that, the second solution, depicted in Table 32, aims at improving the situation by limitation of future sprint scope based on estimations and forecasts. There are two problems to this. First, it requires a tremendous amount of experience in the team to achieve to-the-point scope definitions. In practice, this prerequisite is often not given. Second, estimations are better, as long as the same feature is being developed. For completely new features these estimations tend to become guesses.

Whenever AQA cannot be completed within the sprint, reduce future sprint scopes.
This solution was mentioned by one source.
⮝ The test debt, a gap between development and AQA, is kept low directly. ⮝ Less scope for development can cause better increment quality. ⮝ Scope can be adjusted easily from sprint to sprint, just as required.
⮟ Whatever domain is the slowest, it will define the overall project pace.
This approach is designed to keep the test debt consequently low, even on short sight, by adjusting the scope of the next iteration if necessary. In the opinion of the author, it is usually difficult to estimate the exact scope in advance and backlogs rather tend to be overloaded than half-empty.

Table 32: Solution for Hypothesis 11 - Reduce Sprint Scope

6.8 Three-Field Agile

6.8.1 Goal

As emphasized in the previous chapters, the high occurrence rate of changes in agile projects paired with the low priority of adequate documentation poses challenges on numerous aspects of AQA. Essentially, the problems described by Hypotheses 3, 4, 5 and 6 are all caused or at least intensified by the change management situation in agile.

Additionally, as a consumer of multiple input artifacts, such as requirements, user stories, and software increments, it is very hard to estimate, plan, and integrate AQA properly due to the dependency on future artifacts. This situation is elaborated by Hypotheses 1 and 11 in detail.

The Three-Field Agile approach shall help to overcome these two major problems by:

- Offering a distinct structure in terms of interfaces and chronological order for improved collaboration between different project domains.
- Introducing a lightweight change management that achieves balance between procedural constraints and development flexibility.

6.8.2 Background

The idea of a three-field rotation system originates from the agriculture and dates back as far as the ninth century. “It consisted of planting one field, usually with a winter cereal, a second with a summer annual legume, and leaving a third field fallow. The following year a switch would occur” (Bruns, 2012, p. 25). The saturation of the soil with essential nutrients is kept high by the alternation of different crops and fallow, so the total harvest increases. Similar techniques are still in use today (Bruns, 2012). The technique is sketched in Figure 39.

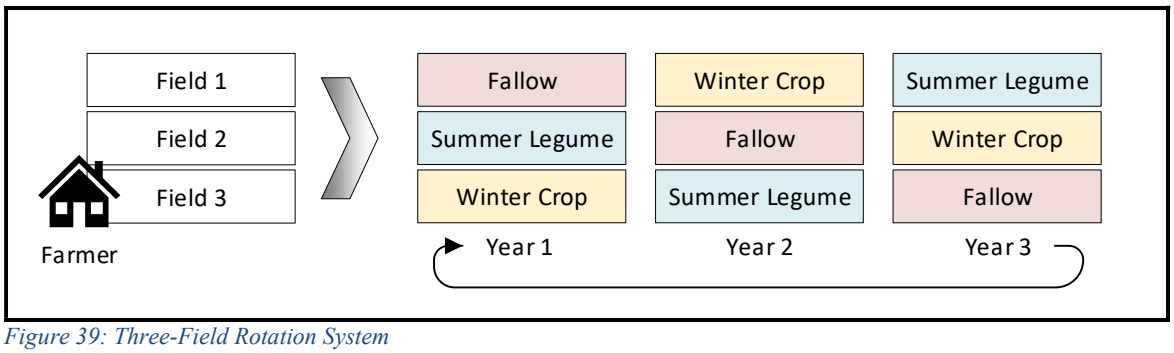


Figure 39: Three-Field Rotation System

6.8.3 General Concept

The Three-Field Agile approach is an extension to the known agile models. It essentially consists of two pieces:

- The first part is a structural **model** proposal which is inspired by analogies to the agricultural three-field rotation system and addresses the problems of separate domain teams, communication, chronology and inter-domain dependency.
- The second part introduces some core **principles** based on laws of nature that result in a lightweight change management when followed in combination with the model.

6.8.4 The Model

The model is based on analogies to agriculture. The lifecycle in agriculture is sow, grow and harvest. These basic activities also define the core phases of the Three-Field Agile approach. Their mapping to software development terminology is explained in Table 33. The activity weeding is added to consider changes but weeding happens while growing.

Activity	Agricultural Meaning	Three-Field Agile Translation	Domain
sow	Seeds are spread on the ground.	Definition and selection of requirements or user stories as scope for the next iteration.	RQE, PO
grow	Plants grow from the seeds.	Implementation of the scheduled scope in the iteration.	DEV
weed	Unwanted weeds are removed.	Removing requirements or user stories from scope during the iteration.	RQE, PO, DEV
harvest	The ripe crops are cut and gathered.	Validation of the implemented scope after the iteration.	AQA

Table 33: Three-Field Agile Terminology

Table 33 shows that the development cycle can be split in three major phases and increments must pass through each of these phases. The Three-Field Agile approach allows one iteration for each phase of every increment. It starts with “sowing” the requirements or user stories by the RQE or PO domain. When the iteration is over, the results are handed over as increment definition to the DEV team for “growing” it. In parallel, RQE, PO and DEV jointly “weed” the definitions by removing unwanted or overloaded content from the increment and the AQA team prepares the “harvest” by creating test cases and arranging environments. After the iteration is over again, the “ripe” increment is handed over to the AQA team for harvesting it by test execution, acceptance tests and overall validation activities.

Needless to say, that there is no idling because while increment n is grown, increment n+1 is sown and increment n-1 harvested. Ultimately, this results in a consecutive order as outlined in Figure 40, quite similar to the agricultural three-field rotation system shown in Figure 39.

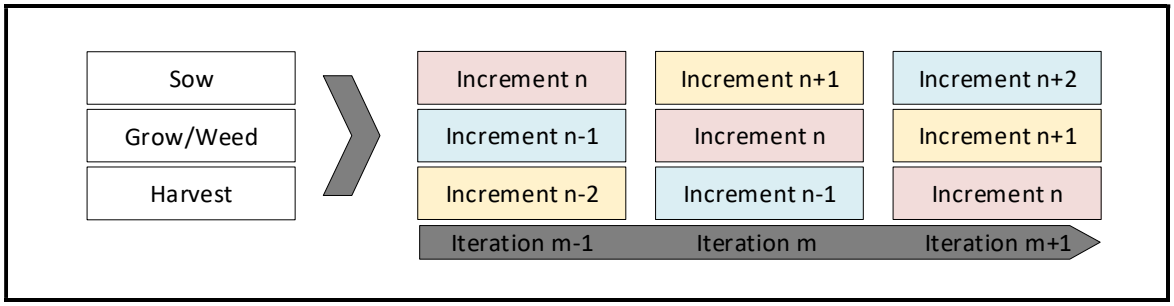


Figure 40: Three-Field Agile Activities

The approach applies separation and distinct chronological order to the phases and domains of agile software development. Each discipline is granted the defined timebox of one iteration for completing its tasks. Resources can be allocated domain-internal just as required to finish in time. Additionally, well-defined interfaces are provided to support collaboration.

6.8.5 The Principles

In addition to the presented model, three simple but crucial principles are provided to enhance the situation with regard to change management. They are derived from basic laws of nature.

- **You can only harvest what was sown and grown.**
  This principle ensures that the strict order of phases is taken into account for every single aspect of the product. There is no bypass or shortcut. New scope or even any small change must be properly sown and grown, each phase must be respected and passed. Without seeds or with seeds spread late on the field, there is no harvest.
- **Weeding removes the weeds; it does not sow.**
  Weeding allows to remove unwanted and surplus scope from an increment but it is impossible to add or alter scope once sowing is done. This also follows the rules of nature, because weeding cannot turn wheat into rye on the field, it just removes weeds.
- **Remove weeds early to support the crops.**
  If surplus scope is removed early from the increment definition, the teams are less likely wasting time on it and can focus on the essential scope instead. Again, this resembles an analogy to nature. Weeds increasingly disturb the crop growth by casting shadow and stealing nutrients. Therefore, it is advisable to perform weeding early.

7 Conclusion and Outlook

7.1 Conclusion

This thesis aimed to improve the situation of analytical quality assurance (AQA) in agile development environments by answering the question, whether fundamental issues exist that have negative impact on AQA in such settings and if so, provide solution proposals for identified problems and challenges.

Based on the performed qualitative research on the topic, the initially stated hypothesis was corroborated. AQA is indeed insufficiently considered by the agile approach and transformation, still resulting in numerous problems and challenges. It was pointed out, that certain core principles are not only incompatible but even contradictive, such as the constructive mindset in agile and the destructive attitude of traditional AQA. The results also indicate that these shortcomings lead to an overall decrease of AQA efficiency in comparison to conventional environments, mainly due to an overall increase of AQA effort and redundant work. Ultimately, the author was able to identify thirteen problematic aspects, scientifically confirm five of them and collect twenty-six solution proposals, addressing the most relevant problems. One of the presented solutions, the Three-Field Agile approach, was invented by the author and is first published by this thesis.

The chosen approach to derive the initial hypothesis into eight sub-hypotheses, each addressing a separate aspect, turned out as beneficial during the qualitative analysis. It was possible to relate collected evidence directly to corresponding problems. A very distinct picture of the situation was drawn this way. At the same time, the facilitated data acquisition techniques of expert interviews and literature research allowed for a profound, in-depth analysis of the topic, even with a small sample size. Based on seven interview candidates and five literary sources it was possible to extract seventy-eight statements, revealing five further problem hypotheses in addition to the eight derived sub-hypotheses and twenty-six solution proposals. The open, candidate-driven interview part supplied many practical insights and delivered numerous solutions approaches. It was possible to provide evidence that even eighteen years after the release of the Agile Manifesto, the integration of AQA into agile models is still an unresolved issue. Out of the thirteen ultimately stated hypotheses, twelve

received at least indirect agreement and six even direct agreement on an average. On the basis of a worst-case agreement calculation, the following five sub-hypotheses were corroborated, based on the analysis performed in this thesis:

- **Planning:** Forecasting AQA effort is very difficult in agile, but still demanded.
- **Test Effort:** Better product quality in agile is paid by higher AQA effort.
- **Regression:** Agile promotes regression, and regression related AQA is expensive.
- **Changes:** Changes in agile projects cause redundant work and duplication in AQA.
- **Test Levels:** Higher level AQA exceedingly suffers from negative effects by agile.

Many of the identified AQA problems were traced back to common root causes. It turned out that especially the agile flexibility in combination with high release frequencies and low priority on documentation negatively impacts the efficiency of many AQA activities, such as planning, testcase creation, and test automation due to more and redundant effort.

Unfortunately, the identified root causes were, as expected, some of the most fundamental aspects of agile development which cannot simply be disobeyed. This raised the question, whether there are solutions available to balance the integration of AQA into agile projects with precise adjustments rather than sabotaging the whole agile model. In the course of the thesis, twenty-six solutions were identified, some of which indeed deeply corrupt the actual agile concept, like the trimmed agile approach for Hypothesis 1 or the installation of change managers to solve the problem of Hypothesis 5. Others interfere with crucial concepts of AQA, such as the reduction of test coverage to smoke levels, proposed for Hypothesis 2 and 4. On the other side, some very promising solution proposals were found, to precisely address problems while leaving the fundamental agile and AQA concepts intact, too. In the opinion of the author, the most promising approaches in this regard are:

- On product or system level, rely on **explorative testing** instead of specification-based testing. This drastically reduces the redundant effort spend for changing requirements and additionally better fits to the agile mindset.
- **Regression** should in first instance be **addressed by CQA** in the development team rather than subsequent AQA. Not only is the effort for AQA drastically reduced, but also is the total product quality enhanced by better architectures and designs.

- The **Three-Field Agile** approach extends the common agile processes by a simplistic model and three basic principles to improve the inter-domain collaboration and change management. It does not contradict agile principles and potentially enhances the AQA with few lightweight modifications.
- Introduce **dedicated AQA sprints** whenever necessary, even midterm in the project. This way, the AQA team can catch up to the development after delays occurred due to changes, insufficient test automation or overloaded iteration scopes. As positive side-effect, the development team can perform project clean up during this break.

In the end, this thesis has shown that one of the most prevalent development models still lacks a systematical integration of AQA. Considering how widespread agile methodology is nowadays, the negative economic impact of such systematical blind spot can be tremendous. It is surprising that only few sources analyzed this issue in detail. Most sources simply expect the AQA to be part of the agile development teams but for test levels of higher abstraction and globally distributed development teams, separate teams are not unlikely. Thanks to the high amount of interview candidates with a background in the automotive industry, this thesis was able to put a light on this gap and return detailed shortcomings and practical solutions at the same time.

7.2 Outlook

This whole thesis was based on qualitative research and many single aspects and points of interest were found. The primary goal of consecutive research could be to quantify the prominence of these aspects in order to rank their urgencies and reinforce the findings of this thesis, for example by surveys that are deduced from the presented results. Another interesting topic for future research might be the refinement and practical application of the Three-Field Agile approach, to examine its viability and effectiveness in the field. Other questions that came up during this thesis and might be of relevance for future research are: Is it possible to enhance CQA to a level so that AQA regression tests become fully dispensable? How do external test automation teams perform compared to test automation by the sprint team? Does agile amplify the AQA challenges caused by product variants?

Generally, the successful integration of AQA into agile is systematically problematic and the author encourages any further research in this regard to further improve the situation.

References

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Principles behind the Agile Manifesto*. Retrieved from https://agilemanifesto.org/principles.html

Black, S., Boca, P. P., Bowen, J. P., Gorman, J., & Hinchey, M. (2009). Formal Versus Agile: Survival of the Fittest. *Computer Volume 42*.

Bruns, A. (2012). Concepts in Crop Rotations. In G. Aflakpui, *Agricultural Science.* InTechOpen.

Flynn, J. (2022). *16 Agile Statistics [2022]: What You Need To Know About Agile Project Management*. Retrieved from Zippia.com: https://www.zippia.com/advice/agile-statistics/

Hue, M., Verner, J., Zhu, L., & Babar, M. A. (2004). Software quality and agile methods. *Proceedings of the 28th Annual International Computer Software and Applications Conference 2004.* Hong Kong: IEEE.

International Organization for Standardization. (2015). ISO 9000. *Quality management systems - Fundamentals and vocabulary*.

Itkonen, J., Rautiainen, K., & Lassenius, C. (2005). Towards Understanding Quality Assurance in Agile Software Development. *International Conference on Agility 2005.* Helsinki.

Liggesmeyer, P. (2020). *Textbook V-M. 1.1: Software Engineering for Embedded Systems: Software Quality Assurance.* TU Kaiserslautern.

Pews, G., & Möhrle, F. (2017). *Textbook B-M.2: Software Engineering for Embedded Systems: Project Management.* TU Kaiserslautern.

Puleio, M. (2006). How not to do agile testing. *AGILE 2006 (AGILE'06).* Minneapolis: IEEE.

Richards, K. (2022). *Introduction to DSDM (Dynamic Systems Development Method)*. Retrieved from agileKRC Ltd.: https://agilekrc.com/agile-training-courses/agilepm-agile-project-management-training-courses/dsdm-agile-dynamic-systems-development-method

Rombach, D., & Wehn, N. (2014). *Textbook B-M.1.1: Software Engineering for Embedded Systems: Software Engineering Introduction.* TU Kaiserslautern.

Sneed, H. M. (2013, 2). Agiles Testmanagement. *Rundbrief des Fachausschusses Management der Anwendungsentwicklung und -wartung (WI-MAW)*.

Wehn, N., De Assis, P. O., Kuhn, T., Jung, M., & Morgenstern, A. (2016). *Textbook V-M.2.2: Software Engineering for Embedded Systems: Software Architectures for Embedded Systems.* TU Kaiserslautern.

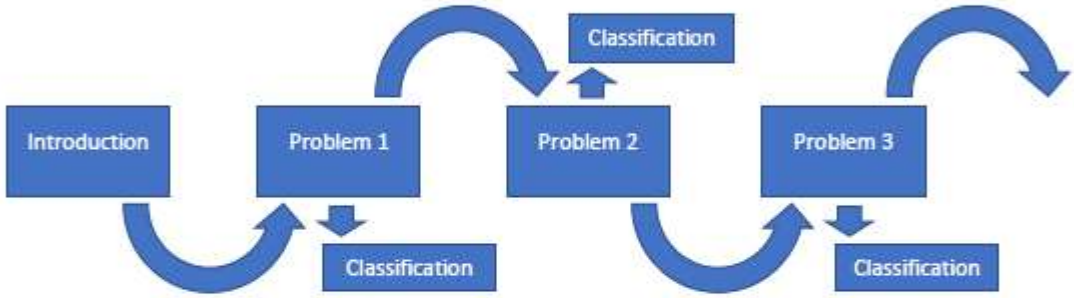
Appendix

Interview Guide and Question Catalogue

Interview Catalogue – Variant B (MT Rottmann)

Version: 02.07.2022

This interview type has a short introduction to classify and get to know the candidate. After that, the ball is passed over to the candidate and the candidate is just asked to name challenges/problems on the topic. The candidate can either go by relevance of the problem or by chronological order of his own professional career. When a specific problem is explained, additional questions can be asked by the interviewer to deepen the understanding of the problem. For example, questions about the company at that time, the project setting or about the interface affected by the problem.



Organisatoric Pre:

- Introduce myself
- Introduce Topic "Improving the situation of analytic quality assurance in agile projects"
- Purpose of the interview: Collect issues and potential solutions from AQA point of view
- Organizational aspects of the interview
  - Duration
  - Can be cancelled at any time
  - Everything will be used anonymously
  - Will be recorded, ask for agreement
- Explain Structure of the review
  - Part 0: Collect general information about candidate for background
  - Part 1: Open problem discussion
  - Part 2: Guided problem discussion
- Kick off

Interview – Part 0: Getting to know the candidate

This interview is about your experience. I am happy to have you here as an expert on the matter of testing in agile environments or associated to agile environments. The goal of this interview is to identify problems and challenges that analytic quality assurance (testing) is facing in agile environments.

As you are my source of information, I would like to ask you to first briefly introduce yourself so that I can get an idea of you and your professional career.

- Name, Age
- Recent position
- General experience associable to AQA (testing)
  - Stations/Jobs
  - How long in total?
- General work experience in agile or agile related projects?
- General work experience in conventional projects?
- Agile: Fan or ban?

Interview – Part 1: Open problem description round

I would now like to get more insights on your work and experience. The goal is to identify challenges and problems that are related to AQA (testing) in agile contexts. If you managed to solve the problems, the information on how this was done is also very welcome and a desirable part of this interview. For every problem/challenge that you want to bring up, consider that also the setting, your position and other aspects are of interest.

You may strive through your thoughts chronologically or by relevance for this topic. Keep in mind, it is not necessary, that your team or the whole project was working agile. For this interview it is sufficient that some aspects of the project were developed with agile methods.

I would now just hand the ball over to you and ask you: What problems and challenges have you experienced, while working on AQA (testing) in an agile project context?

1) Please sketch the problem briefly!
2) What was the setting of the problem?
    a. What kind of company have you been working for?
        i. How many employees does the company have (roughly)?
        ii. What is the main business domain of the company?
        iii. Is the company considered to work agile?
            1. Since when did the company adapt agile methods?
        iv. Where multiple companies involved in the project?
            1. Did the dev-model differ from company to company?
    b. What kind of project was affected?
        i. What was the domain of the project?
        ii. What kind of project was it (upgrade, extension, prototype, ...)?
        iii. Did the project have to fulfill a safety level?
        iv. What domains/teams were involved in the project?
            1. Did the dev-models differ within the project teams?
            2. Which team was working agile, which was working conventional?
                a. Why?
    c. AQA (your Testing team)
        i. Was your team working agile or conventional?
            1. Was your team (or you) a part of the sprint team?
        ii. What was the team size of your AQA team in this project?
3) What was your role in this project?
    a. What was your position?
    b. What were your tasks?
    c. Have you observed the problem or have you been affected yourself?
4) What was the problem in detail?
    a. Interfaces/Affected domains
        i. Which domains did the problem affect (apart from AQA)?
        ii. Was the problem directly caused at an interface to another domain?
    b. Was the problem primarily related to an artifact or to an activity?
    c. From AQA point of view, was this an input or output problem?
5) Was the problem solved? / Could you imagine how the problem could have been solved or improved?#

Interview – Part 2: Directed questions (hypothesis and topic based)

Hypothesis-based questions (derived from potential problem list)

1) **Are there problems/challenges with regard to the planning in terms of estimation of effort and cost?**
2) **Are there problems/challenges with regard to general effort spend on the AQA activities? (is it more or less compared to conventional AQA?)**
3) **How do you handle potential regression in your AQA? Are there any problems/challenges with regard to that?**
4) **Are you using test automation in your domain?**
    a. **What problems/challenges are you facing for test automation in AQA?**
5) **Are you having problems/challenges with changes / change management in your project?**
    a. **How are changes communicated?**
    b. **How much effort has to be spend to react on changes?**
6) **Would you say that different test levels face different amounts of problems/challenges when executed in agile environments?**
    a. **What levels face more problems/challenges, which less?**
7) **Are there any challenges/problems related to tooling for interaction with other domains?**
8) **Are you having problems/challenges related to the combination of agile development paired with safety regulations or other standards that you have to obey?**

Interface-based questions

1) Have there been issues specifically in any of the above-mentioned areas (repeat area 1 to 8 from above if necessary)?
2) What are specific issues that you experienced in the interaction with or caused by the following teams (only teams applicable)? Where the problems process related or artifact related?
    a. **Higher Management**
    b. **Project Leader**
    c. **Product Owner**
    d. **Requirements Engineering**
    e. **System Engineering**
    f. **Development**
    g. **Testing (Boss, colleagues, other testing domains)**

Candidate Information Sheet (Print Page)

Time:
Name, Age
Current Position
General experience associated to AQA total in years
Is it required to explain AQA?
Relevant Stations/Jobs in AQA domain
General work experience in agile or partly agile projects?
General work experience in conventional projects?
Agile fan or ban?
Which way are the problems presented? Relevane / Chronological

Statement Documentation Sheet (SDS) (Print Page)

Time:								
Problem Short								DIGGI
Problem Description								
Affected Interface	Customer	MGMT	PL	PO	RQE	SE	DEV	TEST
Affected Hypothesis	Est./Plan	Effort	Regress	Changes	Testlevel	AutoTest	Tool	Safety
Classification	Artifact	Activity	Input	Output				
Was it solved? Solution?								

Time:								
Problem Short								DIGGI
Problem Description								
Affected Interface	Customer	MGMT	PL	PO	RQE	SE	DEV	TEST
Affected Hypothesis	Est./Plan	Effort	Regress	Changes	Testlevel	AutoTest	Tool	Safety
Classification	Artifact	Activity	Input	Output				
Was it solved? Solution?								

Extracted and Categorized Statements from Interviews

Extracted from Interview with Candidate 1:

Problem Short: Effort (Arithmetic Mean: 2/3)
Problem Description: ▪ "The effort spend on testing in a conventional project, I used to work in, was less than it would have been in an agile setting" (+) ▪ "It might be more effort (agile) but could lead to cost savings and improved product quality on the long run."(+) ▪ "Ultimately, the effort for agile AQA is not much higher compared to conventional testing, due to less intensive final qualification testing."(0)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Regression (Arithmetic Mean: -1/1)
Problem Description: ▪ Regression shall be handled in the final qualification test. (N/A) ▪ Candidate prefers the theoretical solution of "regression free software development" as a treat for this problem. (-)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Test Level (Arithmetic Mean: 1/2)
Problem Description: ▪ No difference or cannot be said in general. (0) ▪ System Test is potentially not executable with every sprint, because sprint artifacts might be incompatible to each other. (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | **SE** | **DEV** | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Tool (Arithmetic Mean: 4/2)
Problem Description: ▪ The tooling should be generally considered before the project starts, no matter if conventional or agile BUT the impact a late tool decision or simple a bad tool has on a project is less problematic in conventional projects (less communication dependent due to lack of flexibility). (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Problem Short: Hypothesis 9 added by this Interview (Arithmetic Mean: 2/1)
Problem Description: ▪ Variants of products are not considered in testcase creation and this leads to problems in AQA. (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Hypothesis 9 added by this Interview
Solution or Solution Proposal:

Extracted from Interview with Candidate 2:

Problem Short: Est./Planning (Arithmetic Mean: 4/2)
Problem Description: ▪ In a real agile environment, where large sets of requirements are not known ahead of the project, it does not make sense to plan effort/cost of testing. (++) ▪ "I would never estimate cost/effort without requirements" (++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ RQ not agile, maybe just development agile (incremental instead of evolutionary). ▪ Plan with huge buffers.

Problem Short: Effort (Arithmetic Mean: -1/3)
Problem Description: ▪ "Effort is not a lot less in waterfall." (--) (The candidate assumes that only sprint deltas are tested and not regression) ▪ A similar test effort has to be done (# of RQ is same?). (0) ▪ If the sprint pressure is too high, the effort might become more in agile vs. waterfall. (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Regression (Arithmetic Mean: 3/2)
Problem Description: ▪ A real full regression test for every sprint is "realistically not possible unless fully automated." (++) ▪ Generally, number of regressions is higher in agile, so also retest number is higher (of found bugs). (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Manual or automated smoke tests, automated full test and manual delta test on every sprint.

Problem Short: Autotest (Arithmetic Mean: 5/3)
Problem Description: ▪ Problem with automation of tests: Automation of "bad" unfinished testcases or features is useless effort. (++) ▪ Testcases that are written for manual tests are hard to use for automated testing, as more "formal" description is required". (+) ▪ "Test automation is too slow to cope with sprints. Testcases have to be created, maintained and executed. This takes too much time to fit in one sprint. (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Testcases are only automated after manual execution to ensure quality of the feature and testcase. ▪ Solution (from candidate's practical work experience): "For test automation it is better to now swim in the very fast flowing river (of agile) ..." So, test automation in his company is separated from sprints and can decide themselves in which projects they want to step in for automation and at what time and scope. "We are getting into projects when testcases are usually more mature." So, candidate thinks it does not make sense to try to automize testcases closely related to the sprints because they are too fragile. ▪ Automated testcase creation. (for example for website UIs etc.) will be a product in the future. Link to PLE here, variant testcases in dependency on variant requirements.

Problem Short: Changes (Arithmetic Mean: 0/2)
Problem Description: ▪ Timing of changes is difficult (what changes are introduced in current SW). This is due to a disconnect between sprint planning and change management). (+) ▪ However, the content of changes is always well known, because good tools are in use for dependency relationships and change communication. Also, in agile, a good change management can be done, if good process and environment is provided. (-)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: Also, in agile, a good change management can be done, if good process and environment is provided.

Problem Short: Test Level (Arithmetic Mean: 7/4)
Problem Description: ▪ Overall agile blurs the lines between SYS and SYS Integration level, because every SYS test is actually a SYS INT test in agile. (++) ▪ "SYS is not possible, actually it is more SYS INT." [...] while these levels are clearly divided in waterfall. Real SYS test now only for stress test, field test, ... (++) ▪ Usually, SYS INT was also tested by supplier, now it is tested by customer as part of SYS. (+) ▪ The lower levels are often tested by developers themselves, so not much affected. (++)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Tool (Arithmetic Mean: 5/3)
Problem Description: ▪ "Agile without proper tooling is worse, because the frequency is higher and projects are generally more chaotic." (++) ▪ Everything should be related to each other (artifacts), JIRA is very good with that. (+) ▪ [...] (From Hyp. 4) However, the content of changes is always well known, because good tools are in use for dependency relationships and change communication. (++)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | **SE** | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Problem Short: Safety (Arithmetic Mean: -1/3)
Problem Description: ▪ It can be compatible and even better if regulations change (faster adaption of changes). (--) ▪ "It all depends on how good your agile processes are running." (0) ▪ Regulations like ISO 26262 were created based on V/waterfall models, maybe it has to be reshaped. (+)
Affected Interface: Customer | **MGMT** | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal:

Problem Short: Hypothesis 11 added by Literature (Arithmetic Mean: 2/1)
Problem Description: ▪ Software was delayed or bad quality because development didn't estimate the scope right. This leads to time pressure on testing, too. The delay can sum up in the worst case. Effects are: Insufficient test case creation, test execution, test results. (++)
Affected Interface: Customer | MGMT | **PL** | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Hypothesis 11 added by Literature
Solution or Solution Proposal:

Extracted from Interview with Candidate 3:

Problem Short: Est./Planning (Arithmetic Mean: 2/1)
Problem Description: ▪ Planning cost/effort without requirements and release timing is like looking into the future with a crystal ball. (++)
Affected Interface: Customer | MGMT | **PL** | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Estimate cost/effort based on experience on previous agile projects. ▪ Wait first few sprints until things become a little clearer and plan then, based on better base of knowledge.

Problem Short: Effort (Arithmetic Mean: 2/1)
Problem Description: ▪ The effort for full test of every sprint is incredibly high in agile projects. But to take advantage of the agile development, every increment should be tested as complete as possible. (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Test Automation

Problem Short: Regression (Arithmetic Mean: 1/1)
Problem Description: ▪ To take advantage of the agile development, every increment should be tested as complete as possible. (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Test Automation, that is established (proven in use) already before the project has started.

Problem Short: Autotest (Arithmetic Mean: 2/1)
Problem Description: ▪ Automated Tests suffer under changes in requirements. "Automated tests require maintenance effort to adapt to changes, and in agile project many changes happen." (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Changes (Arithmetic Mean: 1/1)
Problem Description: ▪ In agile projects, change management is even more important than in conventional projects due to the high frequency with which the changes apply. (+)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | **SE** | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Change Manager is required and a solid Change Management Process with CR analysis in all teams

Problem Short: Test Level (Arithmetic Mean: -2/1)
Problem Description: ▪ Unit tests are more affected by agile development because the artifacts on the low-level changes more often than changes in the end-to-end behavior of the system. The unit tests often have to be adjusted, while system tests remain stable. System tests are more inert with regard to changes. (--)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Tool (Arithmetic Mean: 2/1)
Problem Description: ▪ In agile projects or "fake agile projects" it is often not clear what comes when. This is caused by insufficient communication (bad meeting habits) and insufficient tools or the use of multiple different tools. (++)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | **SE** | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Problem Short: Hypothesis 10added by Literature (Arithmetic Mean: 2/1)
Problem Description: ▪ The plan of the how to has to be well setup before the project, the content can then be created agile way. (++)
Affected Interface: Customer | MGMT | **PL** | PO | **RQE** | SE | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Hypothesis 10 added by Interview
Solution or Solution Proposal:

Extracted from Interview with Candidate 4:

Problem Short: Est./Planning (Arithmetic Mean: 3/2)
Problem Description: ▪ Missing knowledge of the "definition of done" of the project makes it impossible to plan cost/effort. (++) ▪ Transformation in Management could at least lead to more consciousness about this fact. (+)
Affected Interface: Customer | **MGMT** | PL | PO | RQE | SE | DEV | TEST
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ A solid transformation in agile might better the situation.

Problem Short: Effort (Arithmetic Mean: -1/1)
Problem Description: ▪ If agile is done correctly, the effort is not more and can even be equal or less. (-)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ A solid transformation in agile might better the situation.

Problem Short: Regress (Arithmetic Mean: 1/1)
Problem Description: (Look at solutions) (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Automated Testing is a chance but only reasonable if product is very stable, otherwise very high cost for maintenance. ▪ Perform as thorough and good as possible on the architecture to create a regression free system. ▪ Only Smoke Test with every Sprint + Delta. ▪ Test automation (with given constraints in agile).

Problem Short: Autotest (Arithmetic Mean: 4/2)
Problem Description: ▪ Automated Testing is a chance but only reasonable if product is very stable, otherwise very high cost for maintenance. (++) ▪ Automated testing is not suitable for agile projects because the effort for maintaining the automated tests and the test environment is just too high in an agile project due to high number of changes (and deep changes). (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Changes (Arithmetic Mean: 5/3)
Problem Description: ▪ Changes are simply thrown into the backlog. (+) ▪ Agile encourages changes and working agile with unexperienced colleagues leads to even more changes due to recovery changes. (++) ▪ Documentation is worse than in conventional projects, intermittent and immature. (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | **DEV** | **TEST**
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Good product owner and the understanding that even in agile environments, changes have to be handled carefully (analyze, review, gauge, introduce). Also changes in agile should follow a change management process just like in conventional development projects.

Problem Short: Test Level (Arithmetic Mean: 6/3)
Problem Description: ▪ Upper right levels are more affected (V-Model related), because here all issues and changes potentize. (++) ▪ Lower levels are closer to development. (++) ▪ Upper levers might require good timing and coordination of multiple sprint teams, so sometimes functions are available on system level at a later point in time. (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Tool (Arithmetic Mean: 2/1)
Problem Description: ▪ Especially when the project is distributed among different teams, companies and maybe even countries, the testers are disconnected easily from the development team if multiple tools are used or the tools cannot enable proper collaboration. This is true for conventional projects but even more for agile projects. It is then difficult to find out "what currently is in scope". Testers have to poll this information and this leads to overhead and also rework on testing site. (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal: ▪ Again, the main issue is a lack of communication. This could be improved by defining processes and tools ahead of the project and defining the project content sprint by sprint.

Problem Short: Safety (Arithmetic Mean: 1/1)
Problem Description: ▪ It is not trivial to apply agile development methods on very complex or distributed projects and not-so-well implemented agile processes make it even worse. (+)
Affected Interface: Customer | **MGMT** | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal: ▪ A solution could be to invest more time (ahead of the project) in planning of agile processes and artifacts while leaving the project content free for evolutionary definition in agile style. ▪ Also, it would be better to also create artifacts that follow the user stories. This often does not happen in agile projects.

Problem Short: Hypothesis 10 added by this Interview (Arithmetic Mean: 2/2)
Problem Description: ▪ Starting to work agile (agile transformation) requires more effort for processes and learning than conventional developments. Still often this effort is not invested, which is a huge barrier for successful agile projects. "Take agile transformation serious, on organizational and personal level!" If this invest is not taken, the result of "agile" will be more conflicts, problems and bugs compared to conventional development. (0) ▪ This could be improved by defining processes and tools ahead of the project and defining the project content sprint by sprint. (++)
Affected Interface: Customer | MGMT | **PL** | PO | **RQE** | SE | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Hypothesis 10 added by this Interview
Solution or Solution Proposal:

Extracted from Interview with Candidate 5:

Problem Short: Est./Planning (Arithmetic Mean: 6/4)
Problem Description: ▪ "Agile and fix deadlines simply do not match." (++) ▪ "Audits asked us for some kind of planning but the estimation of testing effort for unknown requirements is just not possible" (++) ▪ Distributed development makes it very unclear in agile projects who is responsible for what at what time. (+) ▪ Additionally, a once planned PI (program increment) is thrown over by short hand customer wishes that "have to happen in the next sprint". A committed plan (in form of PI) was abandoned by customer just days after it was established. This makes planning of testing activity even worse to plan. (+)
Affected Interface: Customer | **MGMT** | PL | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or Activity | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Use KANBAN instead of sprints and agile. ▪ Planning in larger scales and more complex projects is proposed in SAFe as PI (Program Increments).

Problem Short: Effort (Arithmetic Mean: 0/2)
Problem Description: ▪ The input artifacts in form of requirements where either not available or of terrible quality due to time pressure (focus on representative functionality) within the sprints. This led to a situation where testing just could verify documentation and do some explorative tests. Less effort for testing but also insufficient testing. (0) ▪ The effort on testing is comparable in agile vs. conventional development. In agile it is slightly higher in the beginning to get organization done, but later it is more efficient. (0)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Regression (Arithmetic Mean: 2/1)
Problem Description: Look at solution (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "Regression Testing in agile projects has to happen automated."

Problem Short: Autotest (Arithmetic Mean: 2/2)
Problem Description: ▪ Maintenance of automation is not possible (due to workload) if done by the same testers than testcase creation and test case execution. (+) ▪ Once you've fallen behind with automation of testcases, there is no way to recover. (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ A separate team for test automation has to be formed that is concerned with continuous testing. This separates the effort and tasks of test automation from common testing tasks and gives more capacity for test automation

Problem Short: Changes (Arithmetic Mean: 8/4)
Problem Description: ▪ If features or customer wishes change during the development, RQs and stories are usually not updated due to time pressure in the sprints. (Bad to no change management). (++) ▪ Even fixed and committed PIs were changed on short hand by the customer and lead to chaos and obsolete work. (++) ▪ The testing was separated as an own sprint team (which was too large). Then the testing team disconnected more and more from the development teams and requirements where increasingly unclear. (++) ▪ In agile, things are just changed without proper communication or feedback into stories/RQs due to time pressure. Testing also left out in the communication via mail. This leads to much useless work being done by testers due to false RQ or information. (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ If changes are to happen, all stakeholders have to be informed. Ideally this shall happen automatically in a tool and based on artifact-dependencies (like in Doors).

Problem Short: Test Level (Arithmetic Mean: 1/2)
Problem Description: ▪ The problems that arise from using agile methodology are same for all levels. On some level the effect just might be worse. (0) ▪ At composed system level, it has been very unclear who was responsible for what (in a multi company system development). This has led to unmatching increments on system level and no matching increments. For testing this means that less tests were executable and testing was reduced to a minor smoke test (because nothing else feasible). (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | **SE** | **DEV** | **TEST**
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Tool (Arithmetic Mean: -1/2)
Problem Description: ▪ The tools are only means to an end. They shall be supportive but agile is more about direct communication and short direct talks between team members are more important than documentation. (-) ▪ Still, it is important to document things in tools. (0)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Problem Short: Safety (Arithmetic Mean: -4/2)
Problem Description: ▪ Agile could even lead to a better safety implementation. (--) ▪ Even if in agile often the focus is too much on function than on processes and artifact quality, it will generally have a positive effect on safety. (--)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal:

Problem Short: Hypothesis 10 added by an Interview (Arithmetic Mean: 1/1)
Problem Description: ▪ It is good to do larger style planning before sprints (i.e., SAFe in form of PIs) but unfortunately the PI plannings are sacrificed for the sake of immediate customer wishes. (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | **RQE** | SE | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Hypothesis 10 added by an Interview
Solution or Solution Proposal:

Extracted from Interview with Candidate 6:

Problem Short: Est./Planning (Arithmetic Mean: 2/1)
Problem Description: ▪ Predictions cannot be made with regard to cost and effort in agile projects, especially if AQA is performed by a team that is separate from the development team. (++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Effort (Arithmetic Mean: 3/2)
Problem Description: ▪ Testing in agile environments is more expensive, due to (1) idle times when waiting for increments and (2) because of changes that make created tests obsolete. (++) ▪ If the same amount of effort is spent for testing in an agile and a conventional project, the test coverage in the agile project will be lower but the product will still be better from user perspective. (+)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | **DEV** | **TEST**
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Regression (Arithmetic Mean: 1/1)
Problem Description: ▪ In the candidate's point of view, regression should not be a matter for the AQA team (no regression tests) but for the developers (prevent regression by experience and excellence of work, architecture, …). (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ This refers to proposed solution 3 of hypothesis (theoretical solution).

Problem Short: Autotest (Arithmetic Mean: 2/1)
Problem Description: ▪ The candidate thinks that solution 2 (automated testing) is not feasible for agile projects due to the "high testcase maintenance effort and low system stability with regard to changes" in agile projects (system unstable and changes happen to often). (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Changes (Arithmetic Mean: 2/1)
Problem Description: ▪ Candidate thinks that change management is always bad (conventional and agile). However, he also thinks that agile projects encourage changes more which leads to higher number of changes. Also, the candidate thinks that "in agile projects changes are less formal" than in conventional projects, where mostly every change has to be paid for separately. This leads to less changes and better change quality in conventional projects, which is ultimately better for the AQA activities. (++)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Test Level (Arithmetic Mean: 8/4)
Problem Description: ▪ "End to End testing (System Test) is very difficult to achieve in agile projects and it is very expensive, while test levels closer to software are easy to achieve and even can be done by developers." (++) ▪ "End to end tests are extremely expensive (in agile projects)." (++) ▪ "Shall I invest all this effort to make an end-to-end testcase, just to realize the system evolved away in the next sprint and the testcase is futile?" (++) ▪ Requirements/Stories on the end to end (System) level are changing in agile projects and the effort spend on testing in this area can just not be justified. (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | **SE** | **DEV** | **TEST**
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Safety (Arithmetic Mean: 1/1)
Problem Description: ▪ Some projects like safety projects often can be developed better in conventional ways because all requirements are fix anyways. (+)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | **SE** | DEV | **TEST**
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal:

Problem Short: Hypothesis 9 added by an Interview (Arithmetic Mean: 2/1)
Problem Description: ▪ High number of variants poses an issue on development but even more on testing. A full coverage is almost impossible. (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | **SE** | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Hypothesis 9 added by an Interview
Solution or Solution Proposal:

Extracted from Interview with Candidate 7:

Problem Short: Est./Planning (Arithmetic Mean: 6/3)
Problem Description: ▪ Agile projects in the wrong form of contract (Werksvertrag) impose heavy risk on the supplier because the scope of projects is impossible to determine beforehand. (++) ▪ "Planning cost and effort of agile is only possible of at least a rough content frame and contract is given." (++) ▪ Effort is not estimable. Scope changes again and again and testing is not able to follow up. (++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Constraints and an agreed framework + rough scope is required to make some kind of planning viable, also in advance of more detailed sprint planning.

Problem Short: Effort (Arithmetic Mean: 6/3)
Problem Description: ▪ Testing in agile projects has a bad efficiency (cost vs. effect) when compared to conventional projects. (++) ▪ Here (in A-Sample grade project) the effort is comparable to conventional development in agile testing. "If you continue agile in B- or C-Sample, agile will crash you!" (++) ▪ Feature focused testing is forced by agile orientation on features (complete a feature first while others are not even touched on basic level). This leads to many changes in the software and testing later on. This increases the effort of testing because testcases have to be maintained. (++)
Affected Interface: Customer | MGMT | PL | PO | **RQE** | SE | **DEV** | TEST
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ In early sprints only perform some lightweight testing on End-to-End level and go into details (Unit) and increased coverage in later sprints. ▪ Using agile is only efficient for A-Sample development, where soft scope (many changes) meets weak process requirements (i.e., NOT aSpice). Here the effort is comparable to conventional development in agile testing. "If you continue agile in B- or C-Sample, agile will crash you!" ▪ A good architecture and design will help to decrease the effort for agile testing.

Problem Short: Regression (Arithmetic Mean: 2/1)
Problem Description: ▪ Regression test can only be done in a certain abstract level, for example only 1 test for feature. (This would be valid in an early stage like A-Sample, refer to Hyp2 here). (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | **Regress** | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ Test Automation is potential solution to the problem of rising effort for regression tests. Generally, only Blackbox testing and abstract tests on SW and SYS level are reasonable to automize in agile projects (+), Unit tests would have to high maintenance effort. ▪ Regression should at first place be avoided by good software architecture and design.

Problem Short: Changes (Arithmetic Mean: 3/4)
Problem Description: ▪ In agile projects, reviews are an essential method to ensure to communicate changes properly through all teams. (0) ▪ While waterfall development offers a clear structure of who is involved in a change, the (responsibility and communication) structure in agile projects is less defined and more communication overhead is required. (+) ▪ Agile favors changes, the testing has to cope with it. (++) ▪ The lead engineers need to be good in technical understanding and diplomatically, to discuss and negotiate changes. (0)
Affected Interface: Customer | MGMT | **PL** | PO | **RQE** | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ A framework for changes is defined ahead of the project. Also discuss changes in the whole team, this is time consuming in the first place but in the end a time saver with respect to the whole project (due to better quality and change management).

Problem Short: Test Level (Arithmetic Mean: -1/2)
Problem Description: ▪ Complex systems are very difficult to synchronize and coordinate in agile with respect to timing and scope. But SYS melts into SYS INT (SYS6 to SYS5), there are less differences. SYS INT (SYS5) becomes the most important aspect of the project together with the architecture of the system. (+) ▪ "SW4 (Unit Testing) is a catastrophe in agile projects." (--)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal: ▪ Software interfaces and also process interfaces have to be well defined and maintained to make an agile system development possible. ▪ Only Performance and other non-functional tests in Unit Test level, despite of that no unit tests in agile projects.

Problem Short: Tool (Arithmetic Mean: 1/3)
Problem Description: ▪ Appropriate tooling is not crucial within the testing team but essential for coordination between RQE, SE, DEV (left V) and Testing (right V). (+) ▪ A tool that supports relationships between artifacts is equally important in agile and conventional development. (-) ▪ As communication is much more important in agile projects, a good tool that supports reviews is also a necessary prerequisite. (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Problem Short: Safety (Arithmetic Mean: -1/4)
Problem Description: ▪ Agile and aSpice are not generally incompatible. A fixed framework and experienced employees can make agile and aSpice a match. (-) ▪ "With regard to safety development agile is only feasible with extensive End-To-End testing." Especially with inexperienced testers. (0) ▪ Agile should only be used for safety uncritical parts of software or in very early stages. (+) ▪ Also, the candidate does not think that safety regulations should be made more suitable to agile development but instead favors the idea of having safety critical architecture and parts fixated very early in the project. (-) ▪ For safety reasons, the author highly appreciates the patience and support by his girlfriend and family. This thesis wouldn't exist without your commitment. Thank you all.
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal: ▪ Only exact packages and very well-defined interfaces (software architecture) are necessary to combine agile development with safety.

Problem Short: Hypothesis 10 added by an Interview (Arithmetic Mean: 3/2)
Problem Description: ▪ Constraints and an agreed framework + rough scope is required to make some kind of planning viable, also in advance of more detailed sprint planning. (++) ▪ A framework for changes is defined ahead of the project. (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | **RQE** | SE | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Hypothesis 10 added by an Interview
Solution or Solution Proposal:

Extracted and Categorized Statements from Literature Research

Extracted from Literature (Hue, Verner, Zhu, & Babar, 2004):

Problem Short: Testing is done more often (more effort) (Arithmetic Mean: 8/4)
Problem Description: ▪ "The difference between agile acceptance testing and traditional acceptance testing is that acceptance testing occurs much earlier and more frequently in an agile development; it is not only done once." (Hue, Verner, Zhu, & Babar, 2004, p.5) (++) ▪ "1) many of the agile quality activities occur much earlier than they do in waterfall development, 2) the frequency of these activities is much greater than in waterfall model" (Hue, Verner, Zhu, & Babar, 2004, p.5) (++) ▪ "Frequency with which these agile QA practices occur is higher than in waterfall development" (Hue, Verner, Zhu, & Babar, 2004, p.6) (++) ▪ "However, is difficult, sometimes even not realistic to compare software quality resulting by the use of a waterfall model with agile methods because their initial development conditions, especially the cost, are not comparable" (Hue, Verner, Zhu, & Babar, 2004, p.6) (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Agile favors changing requirements what leads to many changed requirements (Arithmetic Mean: 1/1)
Problem Description: ▪ "Agile methods achieve software quality under time pressure and in an unstable requirements environment" (Hue, Verner, Zhu, & Babar, 2004, p.1) (+)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | SE | DEV | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: QA shifted from AQA to CQA and development (Arithmetic Mean: 4/2)
Problem Description: ▪ "In agile methods, there are some practices that have both development functionality and as well as QA ability. This means agile methods move some QA responsibilities and work to the developers" (Hue, Verner, Zhu, & Babar, 2004, p.3) (++) ▪ "The conclusion we draw here is: 1) agile methods do have practices that have QA abilities, some of them are inside the development phase and some others can be separated out as supporting practices" (Hue, Verner, Zhu, & Babar, 2004, p.6) (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Agile requires much faster communication (Arithmetic Mean: 1/1)
Problem Description: ▪ "This means that the two-way communication speed in agile methods is faster than in waterfall development." (Hue, Verner, Zhu, & Babar, 2004, p.3) (+)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | **SE** | **DEV** | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | **Tool** | Safety
Solution or Solution Proposal:

Extracted from Literature (Itkonen, Rautiainen, & Lassenius, 2005):

Problem Short: Testing effort not predictable (Arithmetic Mean: 1/1)
Problem Description: ▪ "This testing phase often has unpredictable length and effort requirements." (Itkonen, Rautiainen, & Lassenius, 2005, p.1) (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Testing has to be repeated on higher frequency (Arithmetic Mean: 3/2)
Problem Description: ▪ "Working software is the primary measure of progress. This means testing cannot be left as a phase in the end of an iteration since it must provide information on achieved quality early." (Itkonen, Rautiainen, & Lassenius, 2005, p.2) (++) ▪ "This rapid pace creates challenges for quality assurance, especially testing." (Itkonen, Rautiainen, & Lassenius, 2005, p.2) (+)
Affected Interface: Customer | MGMT | PL | **PO** | RQE | **SE** | **DEV** | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Late changes challenge traditional testing (Arithmetic Mean: 4/3)
Problem Description: ▪ "Agility demands that we should welcome changing requirements, even in late stages of development. Testing and static QA methods have traditionally been based on specifications that are completed in a certain phase of development and after that point can be used as a basis for test design and other QA activities. If those documents are allowed to change even in late phases of the development cycle, it clearly challenges the traditional way of doing QA." (Itkonen, Rautiainen, & Lassenius, 2005, p.2) (++) ▪ This means that the documentation, the traditional testing is based on, does not necessarily exist." (Itkonen, Rautiainen, & Lassenius, 2005, p.2) (+) ▪ "Without up-to-date information it is hard to make scoping decisions in order to get the iteration QA activities performed so that the required product quality is reached by the end of the iteration." (Itkonen, Rautiainen, & Lassenius, 2005, p.6) (+)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | SE | **DEV** | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "An example of agile testing on the iteration time horizon is session-based exploratory testing. Exploratory testing is a testing approach that is not based on pre-specified test cases and seems to embody the agile philosophy." [...] "exploratory testing is effective for testing applications from the viewpoint of the end-user" [...] "The nature of exploratory testing helps reach the destructive attitude required in effective testing" (Itkonen, Rautiainen, & Lassenius, 2005, p.6)

Problem Short: Agile favors CQA over AQA and doesn't even consider AQA (Arithmetic Mean: 8/5)
Problem Description: ▪ "Agile methods emphasise constructive quality building practice. Quality evaluating practices, based on a destructive attitude, are few, if any." (Itkonen, Rautiainen, & Lassenius, 2005, p.1) (++) ▪ "Testing approaches have traditionally included a set of activities concentrated to an integration and testing phase at the end of the development project." (Itkonen, Rautiainen, & Lassenius, 2005, p.1) (+) ▪ "However, several methods include a set of good practices for developers, including automated unit testing. Still, only a few methods give any guidance for higher test levels than unit and integration testing." (Itkonen, Rautiainen, & Lassenius, 2005, p.1) (++) ▪ "In agile methods, testing activities are mostly based on conforming to certain good practices and tracking that these practices are followed. This does not provide direct information on product quality." (Itkonen, Rautiainen, & Lassenius, 2005, p.3) (++) ▪ "In agile methods, quality assurance on the heartbeat time horizon is very strong" (Itkonen, Rautiainen, & Lassenius, 2005, p.5) (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal: ▪ "Exploratory testing is effective for testing applications from the viewpoint of the end-user" [...] "The nature of exploratory testing helps reach the destructive attitude required in effective testing" (Itkonen, Rautiainen, & Lassenius, 2005, p.6) ▪ From CoC Integration: "Tasks on the release time horizon include testing that cannot be completed in the iteration schedule, tasks of separate testing group, and testing in multiple environments. A common way of including release QA is to have separate stabilisation iteration at the end of the release project." (Itkonen, Rautiainen, & Lassenius, 2005, p.6) ▪ "DSDM describes that in very large projects, or by contractual constraints, there might be cases where separate (acceptance) testing activities are needed outside the iterations, i.e., on the release time horizon." (Itkonen, Rautiainen, & Lassenius, 2005, p.6)

Problem Short: Testing and agile is not compatible but there are connectors that can be used (Arithmetic Mean: 1/1)
Problem Description: ▪ "Combining testing practices with agile is challenging. Applying, e.g., the V-model is hard. It is based on sequential phases where activities in each phase use the completed work products (documents) of the previous phase. [...] DSDM, and FDD emphasise the importance of building quality into the system with low-level developer QA practices and testing early and often throughout the development life cycle. However, it is not clear how the testing activities are related and synchronised with the other development tasks." (Itkonen, Rautiainen, & Lassenius, 2005, p.3-4) (+)
Affected Interface: Customer | **MGMT** | **PL** | **PO** | RQE | **SE** | DEV | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Hypothesis 10 added by Interview
Solution or Solution Proposal: ▪ "We have used a temporal pacing framework, the Cycles of Control (CoC), in order to better understand the dynamic nature of agile software development and the QA practices as part of an agile development process." (Itkonen, Rautiainen, & Lassenius, 2005, p.4) ▪ "We showed that describing the development process and the used practices with the help of time horizons makes it easier to recognise spots where the process can be enhanced by testing practices." (Itkonen, Rautiainen, & Lassenius, 2005, p.6)

Problem Short: Agile does not leave enough time for testing, because focus is on function (Arithmetic Mean: 2/1)
Problem Description: ▪ "First, the highest priority of agile development is to deliver valuable software to customers early and continuously with a rapid release cycle. This is a challenge for testing because the rapid release cycle puts fixed deadlines for testing activities and does not allow extending the testing period if more defects are found than estimated." (Itkonen, Rautiainen, & Lassenius, 2005, p.2) (++)
Affected Interface: Customer | MGMT | **PL** | **PO** | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: NEW Hypothesis 11 added by this statement
Solution or Solution Proposal:

Problem Short: Testing done in agile (by developers) is not independent enough for good effectiveness (Arithmetic Mean: 2/1)
Problem Description: ▪ "One of the fundamental principles of testing is independency. Myers states that programmers should avoid testing their own programs and that a programming organisation should not test its own programs." (Itkonen, Rautiainen, & Lassenius, 2005, p.3) (++)
Affected Interface: Customer | MGMT | **PL** | **PO** | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: NEW Hypothesis 12 added by this statement
Solution or Solution Proposal: ▪ A separate tester role even in agile

Problem Short: Developers performing testing in agile are not qualified enough for effective testing (Arithmetic Mean: 2/1)
Problem Description: ▪ "Software testing is a creative and intellectually challenging task that requires a lot of specific skills and experience. It is a profession and requires a professional tester, in order to perform effectively and efficiently" (Itkonen, Rautiainen, & Lassenius, 2005, p.3) (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: NEW Hypothesis 13 added by this statement
Solution or Solution Proposal: ▪ "The DSDM method recognises the need for testing skills and Stapleton recommends that at least one lead developer or tester in each team has received training in testing." (p.3) (Itkonen, Rautiainen, & Lassenius, 2005, p.3)

Extracted from (Puleio, 2006):

Problem Short: Planning and estimation is difficult to achieve in agile (Arithmetic Mean: 8/4)
Problem Description: ▪ "We had difficulties with planning, estimation, task breakdown, managing requirements and working together" (Puleio, 2006, p.1) (++) ▪ There were several areas regarding testing that we had challenges with: communication, estimation, and automation." (Puleio, 2006, p.2) (++) ▪ "The tester in the above communication did not, and still does not, believe that testing can be broken down into small estimable tasks like development work." (Puleio, 2006, p.4) (++) ▪ Estimation is tough" (Puleio, 2006, p.7) (++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "Increased communication, more experience decomposing features into both development and test tasks, and a concerted effort by the entire team to figure out a solution to the problem." (Puleio, 2006, p.5) ▪ "But it can be done fairly well. Estimating testing for agile development need not be too difficult. Techniques like Wideband Delphi estimation help, as does listening to each other, and learning from your mistakes" (Puleio, 2006, p.7)

Problem Short: Regression tests explode the effort in agile (Arithmetic Mean: 3/2)
Problem Description: ▪ "How can you run regression tests, new functional tests, security tests, etc. every sprint without increasing the test window from a week to three weeks?" (Puleio, 2006, p.6) (++) ▪ "One of our often-quoted passages was Chapter 19, which is best summarized as ""No manual tests."" Manual testing slows down regression testing and the feedback loop." (Puleio, 2006, p.5) (+)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "Test automation covers regression and functional tests. If your tests can be setup to run quickly on demand and/or overnight, regression testing and functional testing can be done every day on every check-in." (Puleio, 2006, p.6) ▪ "With good, solid automated functional and acceptance tests, it should only take minutes to determine if there are any regression bugs in the existing functionality, leaving much more time for working on other aspects of testing." (Puleio, 2006, p.5)

Problem Short: Test automation is challenging and problematic itself (Arithmetic Mean: 5/5)
Problem Description: ▪ "The biggest area of contention the team encountered was testing. Neither the Product Owner nor the developers on the team understood the challenges and difficulty inherent in testing a complex software system in an automated manner." (Puleio, 2006, p.2) (++) ▪ "This tool allowed very simple scripting to be done, but it required a lot of manual setup and interaction from test engineers." (Puleio, 2006, p.5) (++) ▪ "Also, this test tool required a lot of interaction and manual effort to determine if a particular test passed or failed." (Puleio, 2006, p.5) (+) ▪ "As a result, over time, the team accumulated what we termed ""test debt"", or test items that needed to be done, but were deferred" (Puleio, 2006, p.5) (+) ▪ "Automation is the key to agile testing [...] you will have made an investment that will pay off later in the project." (Puleio, 2006, p.7) (-)
Affected Interface: Customer | **MGMT** | **PL** | **PO** | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | **Autotest** | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "Over the course of three 2-week Sprints, the team completely listed out and removed the accumulated test debt from the backlog. This required a shift in focus from approximately 70/30 split between development work and test work in a sprint to a split of approximately 10/90." (Puleio, 2006, p.5) ▪ "We found that if automation of acceptance testing will be part of a sprint definition and ""done"" criteria, and test automation is beyond the skill of the testers, the team must develop a test strategy and automation in advance of development." (Puleio, 2006, p.5)

Problem Short: Bad estimation by developers generates time pressure in testing (Arithmetic Mean: 2/1)
Problem Description: ▪ "Estimation is challenging. Most developers are quite poor at estimating work, usually under-estimating the time required to complete development of a set of features. This usually leaves little time for testing." (Puleio, 2006, p.4) /(++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | **DEV** | TEST
Classification: Artifact or **Activity** | **Input** or Output
Associated Hypothesis: Hypothesis 11 added by Literature
Solution or Solution Proposal:

Extracted from (Black, Boca, Bowen, Gorman, & Hinchey, 2009):

Problem Short: Cost, time and effort in agile projects not always lower (Arithmetic Mean: 2/3)
Problem Description: ▪ "Proponents of agile methods claim that they decrease development time and lower development cost. While there have been several great success stories, the jury has yet to rule on critical applications." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.42) (++) ▪ "That agile developers emphasize working code, rather than documentation, might give the impression that development is proceeding more quickly because tangible progress is being made. We do not consider this is a bad thing, but we must be careful not to incorrectly assume that speed is of the essence with agile methods - nor should it be assumed that less effort is involved." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.42) (++) ▪ "This means that neither formal nor agile methods offer significant reductions in effort over the competing approach." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.43) (--)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | DEV | TEST
Classification: Artifact or Activity | Input or Output
Associated Hypothesis: Est./Plan | **Effort** | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Changing requirements are root cause to a lot of overhead work due to rework (Arithmetic Mean: 0/2)
Problem Description: ▪ "In the agile world, requirements change rapidly - developers expect this and are not fazed by the possibility of having to discard their work and start over." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.39) (+) ▪ "Many developers abhor anything to do with documentation, but this is not a problem in the agile world. [...] Some might therefore regard failure to document as a risk associated with using the agile development process." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.40) (-)
Affected Interface: Customer | MGMT | PL | **PO** | **RQE** | **SE** | **DEV** | TEST
Classification: Artifact or Activity | **Input** or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | **Changes** | Test Level | **Tool** | Safety
Solution or Solution Proposal: ▪ "There must requirements traceability, otherwise there can be little guarantee that the end product will meet the customers' requirements." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.39) ▪ "By introducing some formality into the agile process, we can get documentation for free, such as assertions and specifications of tests to be generated." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.40)

Problem Short: Agile does not really raise confidence (i.e., for dependability issues) (Arithmetic Mean: 2/2)
Problem Description: ▪ "Rather, formal methods bring the advantage of certainty in dealing with critical applications, assurance, and solid documentation, while agility brings the benefit of flexibility, customer satisfaction, and tangible progress." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.43) (+) ▪ "Clearly, however, a possible cultural divide currently makes the agile and formal methods communities' incompatible collaborators." (Black, Boca, Bowen, Gorman, & Hinchey, 2009, p.44) (+)
Affected Interface: Customer | MGMT | **PL** | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or Activity | Input or **Output**
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | **Safety**
Solution or Solution Proposal:

Extracted from (Sneed, 2013):

Problem Short: Management agile vs. traditional does differentiate (Arithmetic Mean: 11/7)
Problem Description: ▪ "Die Frage stellt sich, ob es überhaupt möglich ist das Management der Testaktivitäten getrennt zu betrachten. In konventionellen Projekten war dies möglich, weil das Testen ein eigenes Subprojekt bildete." (Sneed, 2013, p.43) (+) ▪ "…werden für das ganze Projekt Aufwände geschätzt und Kosten ermittelt. In einem agilen Projekt ist dieser Gesamtheitsansatz zum Testen nicht möglich. Da die Anforderungen Stück für Stück während des Projektes erst ermittelt werden, können wir nicht wissen, was wir testen sollen." (Sneed, 2013, p.44) (++) ▪ "Wir können auch nicht zu Beginn des Projektes eine endgültige Schätzung der Testkosten abgeben." (Sneed, 2013, p.44) (++) ▪ "Ohne zu wissen, wie umfangreich der Test wird und wie produktiv die Tester arbeiten, kann der Testmanager nur grobe Analogien zu vergleichbaren Testprojekten ziehen." (Sneed, 2013, p.45) (++) ▪ "dass sie zu Beginn der Entwicklung nicht wissen kann, wie umfangreich der Test insgesamt wird. Das kann sie nur raten, oder allenfalls Vergleiche zu anderen agilen Projekten ziehen." (Sneed, 2013, p.46) (++) ▪ "Johannes Mainusch, Leiter der Software Entwicklung bei der Otto GmbH stellt die Frage, ob Agil und Management sich nicht grundsätzlich widersprechen:" (Sneed, 2013, p.57) (+) ▪ "Agilität ist eigentlich der Totengräber der Manager." (Sneed, 2013, p.58) (+)
Affected Interface: Customer | **MGMT** | PL | PO | RQE | SE | DEV | TEST
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal: ▪ "Für ein agiles Projekt machen wir also nicht einen endgültigen Testplan, sondern nur einen Plan des Plans - eine Schablone, die für jedes Release neu ausgefüllt wird.“ (Sneed, 2013, p.44) ▪ "Wir können allenfalls aufgrund der Erfahrungen aus ähnlichen Projekten einen Kostenrahmen anstecken. Darin werden eine Ober- und eine Untergrenze an Testaufwand zunächst provisorisch festgelegt und später nach jedem Release korrigiert." (Sneed, 2013, p.44)

Problem Short: Agile testing can be conducted in separate team but is still has much higher frequency (Arithmetic Mean: 1/1)
Problem Description: ▪ "Bei größeren agilen Projekten mit über 20 Mitarbeitern wird es möglich sein ein separates Test-Team zu bilden. Dieses Team arbeitet in klassischer Weise neben den Entwicklerteams her. Der Unterschied zu früher ist eben, dass die Release-Intervalle kürzer sind." (Sneed, 2013, p.48) (+)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | DEV | **TEST**
Classification: Artifact or **Activity** | **Input** or **Output**
Associated Hypothesis: Est./Plan | **Effort** | **Regress** | Autotest | Changes | Test Level | Tool | Safety
Solution or Solution Proposal:

Problem Short: Agile does not really favor separate test groups (Arithmetic Mean: 3/2)
Problem Description: ▪ "Bei konventionellen Projekten hat es eine separate Testgruppe mit einem eigenen Testmanager, ein Projekt im Projekt. So etwas gibt es nicht in der agilen Entwicklung." (Sneed, 2013, p.48) (+) ▪ "In solchen Projekten (kleinere agile Projekte mit bis zu 4 Personen) liegt der Test in der Verantwortung aller Teammitglieder." (Sneed, 2013, p.49) (++)
Affected Interface: Customer | MGMT | PL | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or Output
Associated Hypothesis: Est./Plan | Effort | Regress | Autotest | Changes | **Test Level** | Tool | Safety
Solution or Solution Proposal:

Problem Short: Testing often cannot be completed in the rest of the time of a sprint (Arithmetic Mean: 2/1)
Problem Description: ▪ "Falls die erforderliche Testfallanzahl über die Zahl hinausgeht, die der Tester in einem Release testen kann, werden sie in das Test-Backlog eingetragen und bleiben in der Warteschlange. […] Auf diese Weise wächst die Zahl der ungetesteten Story-Points bzw. Testfälle von Release zu Release.“ (Sneed, 2013, p.47) (++)
Affected Interface: Customer | **MGMT** | **PL** | PO | RQE | SE | **DEV** | **TEST**
Classification: Artifact or **Activity** | Input or **Output**
Associated Hypothesis: Hypothesis 11 added by Literature
Solution or Solution Proposal: ▪ "Um sie wieder abzubauen, muss irgendwann ein Test-Release eingeschoben werden. Sonst wäre das "technical debt" zu hoch." (Sneed, 2013, p.47) ▪ "das Team ersuchen, das Release zu verkleinern in dem es einige Stories zurückstellt." (Sneed, 2013, p.47)