FAKULTÄT FÜR !NFORMATIK

A Replicated Experiment on the Effect of Team Size and Individual Experience in Software Architecture Evaluation

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

ausgeführt von

Martin Stefan Heinisch
Matrikelnummer 0626982

am:
Institut für Softwaretechnik und Interaktive Systeme

Betreuung:
Betreuer: Univ.-Prof. Dr. Stefan Biffl
Mitwirkung: Dipl.-Ing. Dietmar Winkler

Wien, 11.11.2009. _____ _____
(Unterschrift Verfasser) (Unterschrift Betreuer)

Erklärung zur Verfassung der Arbeit

Martin Heinisch,
Walserweg 47, 6700 Bludenz, Österreich

"Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wien, 11.11.2009

(Martin Stefan Heinisch)

Abstract

The design of the software architecture is a success-critical issue in software engineering. Changes of non-functional requirements (or quality attributes), for example maintainability and modifiability, can have a major impact on repair efforts, project duration and the total project costs. If changes occur late in the development process their impact can be fatal.

Thus, a major goal is to identify possible changes affecting the architecture in the early stages of software development. The identification of future goals and scenarios can support software architects in focusing on most likely changes within a defined time period. Software architecture reviews - embedded within evaluation processes, e.g., the architecture trade-off analysis method (ATAM) - are effective and efficient approaches to explore various software design options and to identify defects and weaknesses early.

Typically, conducting architecture reviews and identifying valuable scenarios depends strongly on the qualification of individual reviewers. To strengthen individual review results, i.e., focus on critical and most likely scenarios, teams can gain benefits through synergy effects during team meetings through discussion and interaction. A major challenge is to identify the effect of team sizes on the quality of scenarios. Based on previous studies in the area of architecture evaluation, the application of nominal teams, i.e., non-communicating teams, seems to achieve better results, i.e., find more valuable scenarios, than real team meetings. One important goal of this thesis is to replicate the previous study with respect to the impact of experience and team size in scenario brainstorming processes.

Replication is a common technique in empirical software engineering to strengthen previous findings with respect to a body of knowledge in architecture evaluation and scenario brainstorming. An important factor is that all parameters must be comparable in the original and replicated study. The results of this thesis (a) can support reviewers in conducting better scenario brainstorming activities, (b) help project and quality managers in assessing better the quality of software architecture according to future changes, and (c) provide a deeper insight in architecture evaluation processes to increase empirical evidence.

The first part of this thesis aims to explore whether the results of both studies are comparable with each other by considering them as a replication. Therefore, all analysis results are repeated using the data collection of the original study. The second, and more extensive part, discusses whether the two studies can be compared on a scientific level, or not. In order to get sufficient answers, three main problem areas are considered:

- The comparability of the focus groups (degree of equality)
- Analyzing the kind of replication used including error finding with the procedure
- Known problems with replication of software studies

The third part of this thesis summarizes the results according to both studies including aspects of replication with respect to gaining empirical evidence on architecture evaluation and scenario brainstorming processes. Regardless whether individuals or teams are considered it turned out that the approach (top-down or bottom-up) of an evaluation process has no impact. Teams lose scenarios but eliminate less-important scenarios. But experience of the individuals does help and teams with the size of three team members are most economic in terms o efficiency.

Kurzfassung

Das Design einer Softwarearchitektur ist ein erfolgskritischer Faktor. Änderungen von nichtfunktionalen Anforderungen (etwa Wartungsfreundlichkeit, Modifizierbarkeit) bzw. von Qualitätsattributen können sich stark auf die Projektdauer, etwaigen (späteren) Reparationsaufwand sowie auf die gesamten Projektkosten auswirken. Entstehen solche Änderungen erst gegen Ende des Entwicklungsprozesses, so sind deren Auswirkungen mitunter fatal.

Eines der Hauptziele von Softwaretechnik ist es daher, mögliche notwendige Änderungen betreffend die Architektur schon früh in der Softwareentwicklung zu identifizieren. Durch die Identifizierung von zukünftigen Zielen und Szenarios ist es leichter, den Fokus auf die wahrscheinlichsten Änderungen in einer bestimmten Zeitspanne zu legen. Softwarearchitekturreviews - eingebettet in den Evaluierungsprozess (zB ATAM) - sind effektive und effiziente Ansätze um verschiedene Designs zu durchleuchten und um Defekte und Schwächen früh zu erkennen. Beides, die Durchführung von Architekturreviews als auch die Identifizierung von gewichtigen Szenarios hängt stark von der Qualifikation der einzelnen Personen des Reviews ab. Um die Einzelresultate zu verbessern - i.e. Fokussierung auf kritische und wahrscheinliche Szenarien - können Kategorien für die Szenarios im Voraus definiert und während der Evaluierung verwendet werden. Teams können Synergieeffekte durch Diskussion und Interaktion nutzen. Eine große Herausforderung ist, den Effekt der Teamgröße auf die Qualität der Szenarien umzulegen. Basierend auf früheren Studien im Bereich Softwarearchitekturen scheinen nominelle Teams bessere Resultate zu erzielen als reale Teams.

Replikation ist eine bewährte Methode in empirischer Softwaretechnik um vorangegangene Resultate zu überprüfen. Die Vergleichbarkeit aller Parameter mit der Originalstudie ist dabei ein wichtiger Faktor. Das Ergebnis dieser Thesis kann (a) helfen, Aktivitäten wie Scenario-Brainstorming zu verbessern; (b) Qualitäts- und Projektmanagern helfen, zukünftige Veränderungen in die Softwarequalität besser einzubeziehen; (c) helfen ein tieferes Verständnis für Softwareevaluation zu gewinnen um die empirische Beweislast zu erhöhen.

Der erste Abschnitt dieser Thesis untersucht, ob die Ergebnisse beide Studien vergleichbar bzw. replizierbar sind. Zu diesem Zweck wurden alle Analysen und Berechnungen mit den neuen Datensätzen wiederholt. Der zweite Teil beschäftigt sich damit, ob sich beide Studien auch wissenschaftlich vergleichen lassen. Drei Hauptproblempunkte werden hierfür diskutiert, um eine aussagekräftige Antwort zu erhalten:

- Die Vergleichbarkeit der Fokusgruppen (Gleichheitsgrad/Vergleichbarkeitsgrad)
- Die Analyse der Replikation als solche inklusive möglicher Fehler während der Durchführung
- Bekannte Probleme von Replikationen von Softwarestudien

Der dritte Teil dieser Arbeit fasst die Resultate zusammen, auch im Bezug auf die Originalstudie. Der Aspekt dieser Replikation, die empirische Beweiskraft in Sachen Architekturevaluierung und Scenario-Brainstorming, zu erhöhen, wird ebenfalls diskutiert. Im Endeffekt stellte sich heraus, dass die Vorgehensweise (top-down oder bottom-up) keinen Einfluss auf die einzelnen Personen oder Teams hat. Zudem verlieren Teams Szenarien, eliminieren dafür Unwichtige. Zudem ist Erfahrung hilfreich und Teams mit drei Mitgliedern sind höchst ökonomisch.

Contents

List of Figures

List of Tables

1. Introduction

The design of software is the principal focus of software engineering.[9, p.1] Like in any other productive enterprise the design can be interpreted as the plan of the final product. No matter if this is a skyscraper, an automobile or any other manufactured item. Software design contains design related structural issues of the system. The aggregation of them leads to the software architecture. Hence, software architecture is a (specified) form of software design.[10, p.3] Any software architecture evaluation requires a (pre-) defined architecture which can be reviewed.

1.1. Critical impacts of software design

Despite technological improvements the challenges of software design today are quite the same as forty years ago: How to design a set of programs into a complete software system? How to design and build a system robust, tested and well-documented which contains the necessary artifacts to obtain (pre-) defined goals?[9, p.3] Therefore, the software design respectively the software architecture can be seen as success-critical for the whole software project. Three fundamental issues are relevant [10, p.11]:

Mutual communication between stakeholders

The software architecture representing the software design is a high-level abstraction of the software system. Most, if not all, system's stakeholders can communicate with each other, creating mutual understandings and forming consensus by using these design related documents.

Early design decisions

The software architecture represents early design decisions of the system. The impact of these bindings on the system's remaining development, its service in deployment and the maintenance until the end of the life cycle is very high.

Transferable abstraction of a system

The software architecture defines how the system is structured and how the involved components work together. However, this model is transferable across systems especially if they underlie similar requirements. Large scale reuses of proved software design are possible too.

Another way of representing the critical aspects of the software design is trying to express them quantitatively using the costs of a software project.

Impact of software design on software project costs

Due to the famous *Department of Commerce's National Institute of Standards and Technology (NIST)*, errors, bugs or similar software problems created damage costs for the U.S. economy of approximately $59,5 billion dollars in 2002.[11, 12] In other words, software failures add up to extraordinary 0,6 percent of the U.S. gross domestic product. Really astonishing is the fact that over a third of those errors, about $22,2 billion dollars worth, could be avoided through improved testing environments whereby software defects can be identified and removed earlier. A large percentage of those are produced within the development process but nearly half of all of them are not detected during development or beta testing.[11]

However exhaustive testing of all possible features, situations, or paths is impossible for most (large scale) software systems.[13, p.1] Moreover, if a certain level of software refinement has been achieved further efforts spend to increase reliability will increase exponentially. The right timing when to release a software system to the customers and stop testing it is important. Especially when the total software costs as well as all future warranty and risk costs shall be minimized.[13, p.1]

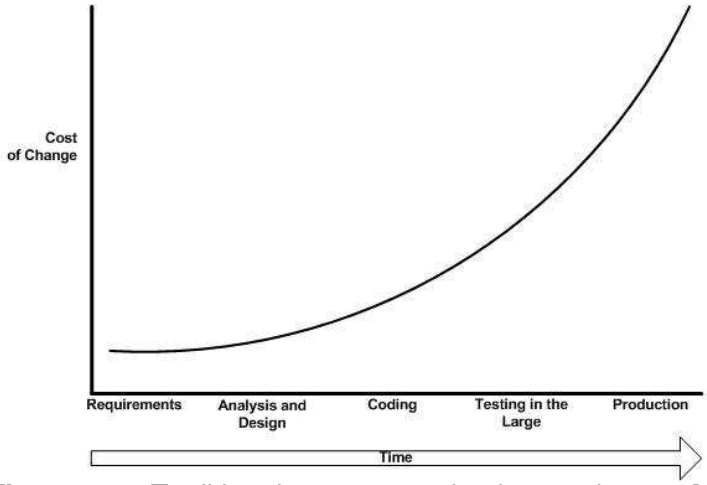


Figure 1.1.: Traditional cost curve of software changes[1]

Among others, decisions dealing with the architecture of a software system are the earliest ones to make. Corrections or error fixing because of wrong or bad decisions dealing with

this matter are most costly.[14, p.1] As figure *1.1* shows, changes and corrections of software systems are getting more and more expensive in the process of time. In many cases those late changes are necessary because of errors and bad decisions made within the early stages of a software development project.[12, p.1]

As more and more components are based on previous work more and more corrections have to be done if a problem made at the beginning of the project is covered up later on. In the run up of software development the major task is to define the software design respectively the software architecture and to ensure that as few mistakes as possible are made.

So, the focus of error avoiding is drawn to the design process. Software architecture evaluation provides effective methods to find architecture related errors at the design stage of a software development process.

1.2. Using scenarios in software architecture evaluation

Main goal is to avoid bad or buggy software architecture design. In order to do so, the involved persons must be able to assess the impact of their decisions on the desired quality objectives respectively the quality attributes of the software by the time they are made.[14, p.1] Examples of such attributes are the performance, the reliability or the functionality.

Software architectures are not intrinsically good or bad.[15, p.7] Instead the architecture has to be evaluated with respect to the requirements of the stakeholders of the software project. An effective way to achieve that is the usage of methods to evaluate the software architecture in terms of design errors and the stakeholder's requirements. Today a couple of such methods exist. The most important ones are SAAM and ATAM.[15, p.2] Most of them are more or less related to each other. Among many similarities they often use scenarios.

Software architecture analysis is a very cost-effective way to control risks and to maintain the system quality throughout the software life cycle.[16, p.1] Especially important analyzing issues are the software design, the software development and the maintenance work after the software roll-out. Main purpose for scenario usage is to predict the software's ability to fulfill certain quality objects, so called quality attributes, currently and in the future. They are derived from the requirements of the involved stakeholders.[14, p.2] If the software system does match with the defined quality attributes the risk of cost-intensive changes and/or high maintenance efforts is expected to be rather low. Thus, scenarios are considered as very important import to software architecture evaluation methods. The accuracy of the results is heavily related to the quality of the found scenarios.[16, 17]

1.3. IT-research gaps considering scenario based software evaluation

According to recent research work scenario-based approaches for software architecture evaluation are considered as useful and helpful.[18, p.1] Nevertheless there are still open questions which currently cannot be sufficiently answered. One example is the impact of using change scenario categories. Therefore further research work in IT-science dealing with this topic is necessary. Basically the goal is to find out how more and/or higher quality scenarios can be elicited.

Currently there is only little empirical evidence of the relative effectiveness of scenario elicitation techniques.[17, 19] The approach applied by the stakeholders to elicit these scenarios is also questioned. Either the whole process is done top-down or bottom-up. The first one uses change scenario categorization for guiding the scenario elicitation process whereas the latter approach does not.[17, 19] Another topic questioned in IT-science is the optimal team size for evaluation teams again in terms of effectiveness.[6]

One point of interest is the effectiveness of scenario development meetings in terms of the software architecture evaluation process.[20] This research question involves also the fact that individuals on their own as well as evaluation teams are conceivable for the evaluation process in the industrial world. Setting the focus on individuals and/or teams the impact of experience on the performance is also not quite clear.[6] Do well-experienced persons perform better compared to experienced or less-experienced ones? Or do different scoring systems (frequency-based, participants-based or expert scoring) rate scenarios differently considering whether they are critical or not? Again the impact of experience is expected to affect the people when they rate such scenarios.

1.3.1. Research study replication

During a controlled scientific study performed at the University of South Wales, Sydney, above research topics have been investigated.[20, 17, 19, 6] This study as well as its results are the initial position for this work.

As already mentioned, a lot of research still has to be done in this scientific area. However, IT-researchers have covered up the problem that there is a lack of scientific replication work too.[21] Basically, research results which cannot be proven are not scientifically valid.[22]

Therefore the original study was replicated at the Vienna University of Technology in June 2008. Main purpose was to re-investigate the scientific research questions and statements made there and to probably be able to strengthen them. Like in Sydney, the study replication

was performed at a university observing the performance of IT-students. Again both undergraduate (Bachelor) and graduate (Master) students were mixed. It was decided to repeat the study conditions as exactly as possible in order to achieve similar results.

1.3.2. Research goals

Accordingly this replication is the main objective of this work. The design and preparation work as well as the execution process is described in detail. Afterwards the results and findings are presented. Of course this work refers to the original study as both are connected through the same research purpose.

An important step is to compare and to evaluate the findings of the original with those of this replication. The fundamental question is: "Can the results of both studies be compared at a scientific level?" If the answer is yes, the statements derived from the original study can be evaluated. Depending on the findings of the study replication it is possible to affirm or reject those statements on a higher level. This is because more observed individuals and/or teams worked similar under the same conditions producing larger data sets which provide bigger confidence.

The second purpose of this work is to evaluate in terms of replication research whether both studies can be compared on a scientific level or not. Known replication problems as well as other possible affecting factors are considered. This shall either explain thinkable reasons for different results and/or if research errors have occurred respectively have been made.

1.4. Structure of the thesis

The first part of this work, apart from the introduction, describes and explains the theoretical background of this work. Considered is especially the current state of the art within the IT-science regarding software architecture, scenario-based software architecture evaluation and replication research, especially with reference to IT-studies. Additionally the topics software product life cycle and software quality attributes are covered.

The fundamental issue of this thesis is, of course, the workload produced by the performed study replication including the experiment preparation, the actual performance and data collection, the analysis and calculation of results as well as the findings and conclusions. The whole documentation of this is separated into three main parts.

Research approach and experiment description

Conducting a study or replication requires substantial efforts. The preparation, the execution and the postprocessing tasks create a huge workload. This part describes the research approach as well as the experiment proceedings in detail. The research approach involves the created research hypotheses behind this work. Further the topic of “validity” is discussed.

Within the experimental process the design of the study and the basic procedure are explained. Additionally study-related characteristics concerning the participants are presented in this chapter. Basically the team setting and study groups are terms of interest. Further the classification of the individuals regarding their experience and language skills are explained.

Research activities

The most extensive part deals with the actual research work. Creating the data set and performing the data mining have taken a serious amount of time. This was necessary to correct redundancy and eliminate not usable data. However, these activities do not belong to the study research work and are not described within this thesis.

As the replication can be categorized as an “exact replication”, all original calculations had to be repeated based on the new data sets. Additionally further calculations have been done. This resulted from the fact that the replication was executed twice using both times two different software systems - and their architectures - for evaluation. On finishing the calculations the results are summarized and compared with those of the original study.

Comparability

Unfortunately aggregating results of different studies or replications is not that easy to do. In particular the impossibility to achieve equal conditions and environment does provide a lot of problem fields like variability of the results or performance affecting factors. Issues like bias, a shortage of cooperation and communication often come along with them. So the last chapter compares the original study with this replication. The main goal was to find out if and how the results can be aggregated considering whether the findings of both studies are similar and the statements can be affirmed on a wider basis or not.

Finally this work discusses if known replication related mistakes, documented in IT-science, during this research activities have been made. The reason is that through such mistakes probably the results and findings cannot be compared and interpreted together with the original study.

2. Software quality attributes

In several fields of engineering (e.g. civil engineering), quality concerns have a long-term history.[23, p.1] For example, constructing high buildings always involves constant tests of the buildings materials (e.g. concrete, steel) used. Otherwise it would have been impossible to build constantly higher, larger and more complex houses, skyscrapers, bridges, tunnels or else. Safety measures, costs and time schedules forced the planners to develop mechanisms to ensure that the respective quality concerns are matched.

Yet, software development processes differ from manufacturing or civil engineering processes. For example, the daily improvements cannot be seen that clearly whereas when building a wall, the progress can be observed easily. Also defining, controlling and standardizing quality attributes are quite tricky compared to civil engineering. A bridge's quality attribute could be that it has to withstand weights of 16 metric tons, which is a clear definition and there are practical methods for checking this. By contrast, a software system might have the specification to provide a good handling user interface. Especially in advance, it is hard to make assumptions of what users prefer in terms of user interfaces. However, there are similarities on both sides.

Determining quality aspects, attributes and goals are depends largely on the stakeholders involved in the software development. Unfortunately, discrepancies frequently result from different opinions, wishes, ideas and technical knowhow. Limited resources make it impossible to fulfill all quality aspects. Time, costs and quality are interconnected and manipulating one of them affects the others - positively or negatively.

2.1. Classification of software quality attributes

Ultimately, inspection team meetings as a part of quality management are held to ensure the quality of the software product, whereby the system, respectively its architecture, is of particular importance. Consequently, (pre-) defined quality attributes must match common or additionally requested quality levels. In case of a software system, those attributes can be divided into three categories.[24, p.3]

1. Whether Quality attributes exist, can be observed through the output of an existing system given some test input. Usually they are described by names such as reliability, security, availability et cetera. There are time-dependent ones like performance or data throughput and others which are not.

2. Other qualities can be described by measuring the activities of a development maintenance team. These include maintainability, portability, adaptability, and scalability.

3. Further, the activities of a particular user (or in some cases of another system) in concert with the executing system can be measured and described. Usually, these include usability, predictability and the ability to learn.

This differentiation seems to be rather vague and in fact, it would be very difficult to analyze a software architecture using this categorization system. Yet Kazman states that such a classification turns out to be quite useful. Quality itself is occasionally tricky to measure.

Depending on a certain point of view a quality attribute can be important / unimportant, more or less fulfilled or has a higher or lower priority. For example, certain software is able to run on two different operating systems, but not on a third one. Is the program portable (enough) or not?

What do users on both supported systems think in contrast to the users working on the third operating system? At the present time and for the foreseeable future, no simple (e.g. scalar) universal measurement for attributes such as security or portability does exist. Valid attributes in the past have become obsolete now and today's state-of-the-art attributes will be unusable in the future.

Instead, only context-based or context-dependent measurements exist constantly. Additionally, they are only meaningful in the presence of specific circumstances of execution or development. Looking at the *ISO9126 Software Quality Attributes*, the following classification is presented:

Quality Attributes	
Functionality	*Suitability* *Accuracy* *Interoperability* *Compliance* *Security*
Reliability	*Maturity* *Recoverability* *Fault Tolerance*
Usability	*Learnability* *Understandability* *Operability*
Efficiency	*Time behavior* *Resource behavior*
Maintainability	*Stability* *Analyzable* *Changeability* *Testability*
Portability	*Installability* *Replaceability* *Adaptability* *Conformance*

Table 2.1.: Quality attributes according to ISO 9126

ISO itself is also subsuming attributes under convenient classifications, though its approach is less vague and abstract and therefore more precise. It should be pointed out that ISO9126 is a standard for software quality attributes and must be adapted to individual cases. But this list accommodates particularly requirements of software architecture quite well. Another example is the categorization of Lundberg et al. [7] which provides only a very limited taxonomy. The reason leading to this was that the authors created it during a specific experiment. Therefore the categorization represents the quality attributes defined for that. Combined with the knowledge about the topic, they were able to reduce the quality attributes needed to a minimum which can solve many discrepancies between the stakeholders.

Quality Attributes	
Performance	*Throughput* *Response time*
Modifiability	*Maintainability* *Configurability*

Table 2.2.: Quality attributes due to Lundberg et al.[7]

This subset was exhaustive enough to run the experiment. Although this classification is reasonable, it is not useful for this kind of study. Lundberg et al. dealt with specific programs to test exactly described architectural problems. Similarly, the procedure in our studies used

two software systems. But contrarily, the study related to this work uses scenarios which shall be valid for more or less every kind of software or software architecture respectively. The categorizations discussed so far were only used as examples. Dependent on the selected software system, environmental constraints and other project conditions there is no answer which software architecture classification or which quality attributes shall be defined or used. The decision must be made specifically every time. Due to the limited time period of this replication study, the participants had to learn and handle the software in order to develop scenarios for the predictable future. They had to define their own quality attributes which they wanted to consider.

During the replication study, nearly all possible quality attributes were permitted. Nevertheless, in order to get usable results, scenario categories were provided to the treatment group. So the range of the answer sets of these individuals or teams could be limited. Additionally, the participants needed less time to get into the whole topic of (change) scenario based software architecture evaluation.

2.2. Trade-offs of software quality attributes

Throughout the world computer systems use software applications or whole software systems. Many of them are critical, since as failures could have serious consequences involving, among others, danger to lives. Such critical software often has the following characteristics:[25, p.11]

1. The life cycle of these systems may last several years or even decades. Updates or upgrades are of evolutionary kind.

2. Operation of the application / software is required to be almost non-stop.

3. Quality attributes like timeliness, reliability, safety and interoperability are of paramount importance.

Crucial for the success of software (systems) is to identify the requirements correctly. Merely satisfying those to some degree can lead to fatal failures. The quality measure of a software is the degree to which quality attributes or a combination of them are covered by the software system.[25, p.13] Unfortunately, there exist trade-offs between quality attributes. Software developers have to balance them in order to get the best results. To achieve this goal they have to be evaluated. Accordingly, it is one of the ultimate goals to be able to quantitatively measure the trade-offs. The starting point to find quality attributes which can be evaluated this way is a description of the software architecture.[25, p.13]

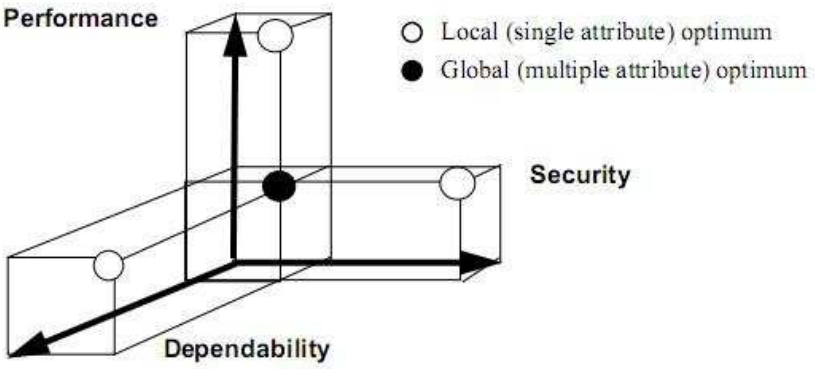


Figure 2.1.: Trade-offs of software quality attributes

As shown in the figure right above, quality attributes related to *performance*, *security* and *dependability* must be traded-off. The limiting factor is the software project budget. Therefore, the costs of all three must not exceed this certain amount which forces to balance them.

2.3. Impact of software quality attributes on software architecture evaluation

An important step is to implement the desired software quality attributes of a system into the system's architecture.[25, p.1] Further it is important that the architecture has to include and consider as many quality attributes required by the stakeholders as possible. The main purpose of the software architecture is to describe the system's components as well as their connections, their interactions and the interaction of the system and its environment.[14, p. 45] This enables software engineers to make objective decisions in terms of design trade-offs and accurate predictions about the system's attributes and their quality.

The specific interest of software developers in software architectures emphasizes the importance of the software architecture as a determining factor for the software quality.[14, p.1] Decisions about the software architecture have great impacts on the quality attributes implemented and as a consequence on the complete final software system. Main target of designing a specific software architecture is to be able to quantitatively evaluate and to balance the multiple software quality attributes in order to achieve a better system (quality) overall.[25, p.1]

This explains why bad decisions or errors respectively mistakes during the design of the software architecture can result in very costly fixing work later on. Therefore it is very meaningful to evaluate the software design/software quality during that period and before the system is actually built.

3. Software architecture

> *The design of application or system software that incorporates protocols and interfaces for interacting with other programs and for future flexibility and expandability. A self-contained, stand-alone program would have program logic, but not a software architecture.*[26]

Many definitions of software architecture exist. And actually there exist many different software architectures as well. Principally, the design of the software architecture is one of three major elements of the whole system design.[27, pp.187-188] Modelling interfaces and the data capturing are the two other elements specified. Undeniably, thoroughly planned software architectures are essential for building successful and high quality software systems. As the complexity and size of an application increase, design problems exceed solutions based on algorithms and data structures [28, p.1] and lead to the next level of problems, i.e. specifying an overall system structure. Main issues pertain to organizational arrangements, team coordination, scaling and performance, architectural design, et cetera.

Software architecture is high-level designing.[24, p.2], [29, p.23] Developers are concerned with the configuration and integration of components which create the final architecture of the system. Interfaces between those components are also part of the architecture. Though the word software is used, in the end often software *and* hardware components provide the functionality desired.

The software architecture defines software elements which are most common described as a set of components.[29, pp.20-23] Further, the software architecture can be explained as the structure of the system containing its elements, their properties and their relationships. Notice, that no structure can assert itself as being the usual architecture used. Too large is the range of problems given and the variety of solutions based on software. Consequently, several architectures and even more varieties exist.

In this context, the software architecture serves somehow as a design plan. From another point of view, a software architecture can be seen as an abstraction.[30, p.5] Handling the complexity of programs is getting far more easily for both the development and evaluation team using such abstractions. Lack of experience and missing guidelines are the main reasons, why wrong details are considered, false steps are taken and important matters are missed. In economy, for example for portfolio investments, managers use scenarios for decision making. Those scenarios contain (environment) conditions, constraints, requirements, etc. and are created

to ensure that all important aspects are considered. Scenario descriptions are abstractions of possible future events.

Consequently, it makes sense to develop scenarios for software architectures in order to deal with them in evaluation meetings. Time shortage and different levels of knowledge and experience demand procedures which simplify and push the whole process in order to achieve sufficient results.

3.1. Scientific and historical background of software architecture

In IT-science it is common knowledge, that early software development stages are of significant importance in order to produce successful software systems. The idea of *software engineering* first came up in 1968.[27, p.29] The tenor was that the development of software shows similar characteristics to other engineering disciplines (e.g. civil engineering). Therefore, software engineers adopted their proved methods and proceedings. As a result, construction plans and procedure models usable for software projects emerged. Well-known examples are the *V-Model* or the *Waterfall-model*. Furthermore, all phases before the actual coding have become more and more important, especially the design phase, which is therefore included as a major stage in most of the currently used procedure models.

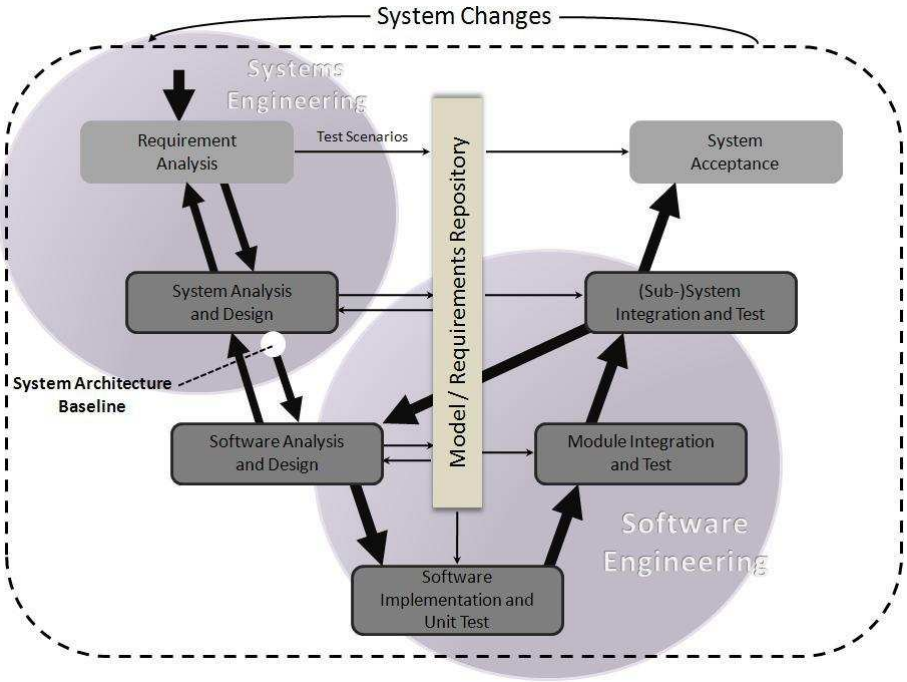


Figure 3.1.: The V-Modell used in software development projects[2]

Since the mid 1980s, IT-scientists began to create a new discipline, i.e. software architecture. The main impulse was that software exceeded levels of range, complexity and size, which could only be dealt with new data algorithms, right data structures, and so on. Consequently, software systems tended to fail more often over time. The approach of designing software architectures at the beginning of the development process is intended to counter such problems. Currently, such architecture is comparable to a construction plan of a building (or at least to a certain degree). And, as in civil engineering, these plans are created before starting the actual work. Logically, the decisions made within this period are of directive importance for the whole project and cannot be changed easily later on. The design phase of a software product strongly determines its success. The whole process is embedded into the software life cycle.

3.2. Software life cycle

> *Der* Software-Lebenszyklus *(software life cycle) ist der Prozess der Entwicklung von Software-Produkten und kennzeichnet alle Phasen und Stadien dieser Produkte von ihrer Entwicklung, Einführung und Wartung bis zu ihrer Ablösung oder Beseitigung.* [31, p.17]

Translation:

> *The* software life cycle *is the process of the development of software products and pertains to all phases and stages of these products from their development, implementation and maintenance to their removal or elimination.*

Every software product experiences several stages throughout its existence, similar to other industrial products. IT-systems are often combinations of their hard- and software. But this coexistence is not constant over time as hardware components might be replaced or substituted. Updates as well as upgrades of the software can occur. However, the average lifetime of hardware components is often limited to two or three years whereas software components, especially when application oriented, can be in use for more than ten years.[27, p.37]

Thus, software development has to be more conscious of future changes. Two main objectives of software architecture are to diminish concomitant effects due to certain impacts caused by them (like costs) and to decrease the likelihood of necessary adjustments by foreseeing developments. Important in terms of technical and development of aspects in the software life cycle is the introduction stage where software development processes are included.

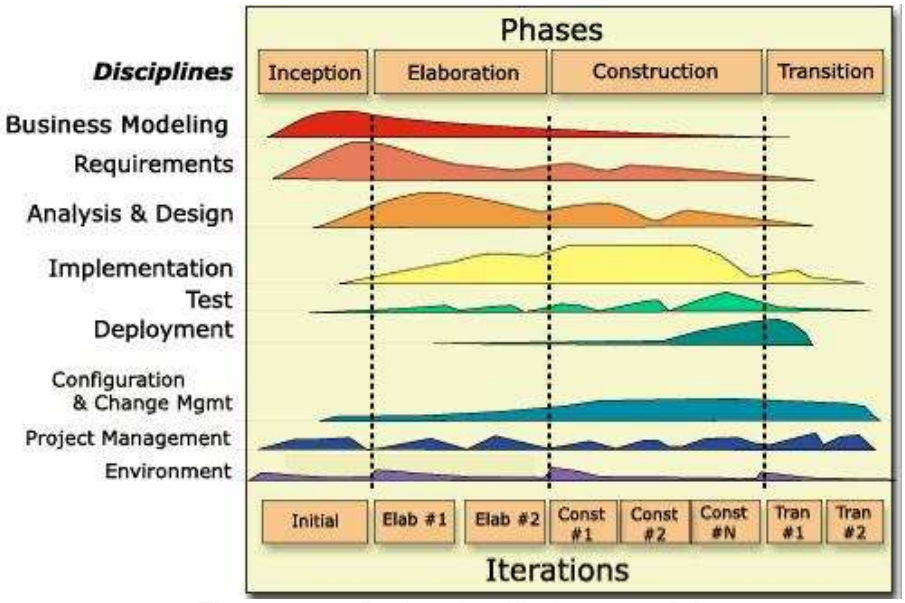


Figure 3.2.: Rational unified process[3]

Within the IT sector, the introduction and the development stage respectively, are structured according to a procedure model which is an abstract illustration of the software (development) process.[2, pp.56-57] Various models have been created to organize and to illustrate the development of software systems within their specific life cycle. Not surprisingly, they differ. However, science divides them into two categories: [31, pp.103-110]

- *sequential life cycle models* having a relatively strict sequence of phases
- *nonsequential life cycle models* using feedbacks to jump from on phase to another

One of the most commonly used sequential life cycle models is the waterfall model. Further, it is quite simple and therefore well suited as an example for demonstration. Even without a software engineering or at least technical background it is easy to understand its software development process. Several clearly defined levels exist.

When all the tasks related to one of them are finished, the next stage is entered. Of course, the boarders are not completely fixed. But it is not intended to give an exact presentation. Other representatives are the *V-Model* (see chapter introduction) or, to give an example of a non-sequential one, the *spiral model*. The latter uses prototypes for progress monitoring.

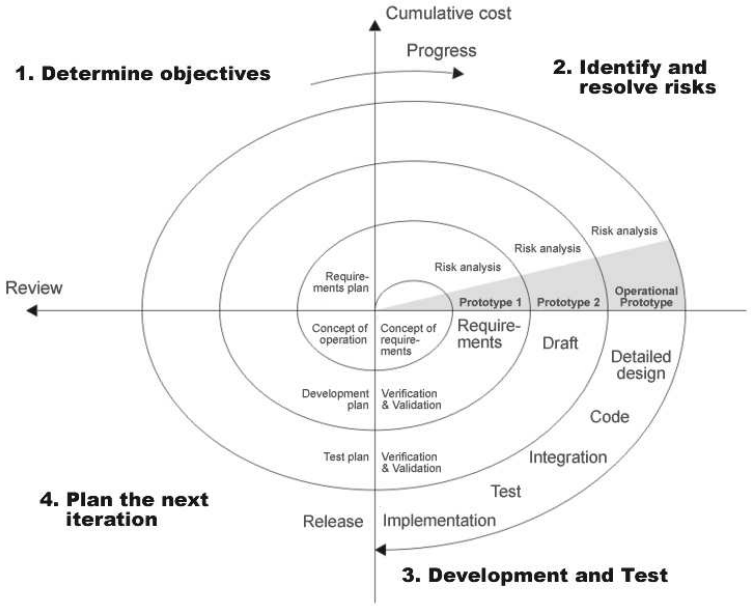


Figure 3.3.: Software development process - Spiral model

Nevertheless, all of these types of models specifically include a design phase. This indicates the importance of this part of software development. Decisions concerning the architectural design within a software development project have to be made very early. Therefore, they are very difficult to get correct. No one can foresee the future and creating applications can last even for years. Especially later on, wrong decisions in the design stage cannot be changed without great expense.

Thus, over time, software architecture design has become more and more important. As a consequence its own place was created within the life cycle process of a software product. Due to the gap between the lifetime of software and hardware, software systems must be designed to handle changes of even critical (hardware) components without larger efforts.

Software developers therefore have to consider factors like which programming language to choose, the implemented interfaces or which hardware standards to consider. In most cases the question is which of them will be used midterm.[27, p.37] Studies show that early efforts spent on system analysis and design are of great benefit later on.[32, p.151]

A graphical view will clarify this topic in a compact way. The B-curve represents old, antiquated project proceedings. Investments in system analysis, system design and coding are relatively small which cause, however, very high efforts with regard to testing and maintenance costs.

Contrarily, the A-expenditure-curve represents up-to-date software development approaches. Efforts spent on system analysis and designing the software system in the early phases of the project are higher, which leads to smaller efforts needed for testing and error fixing later on.

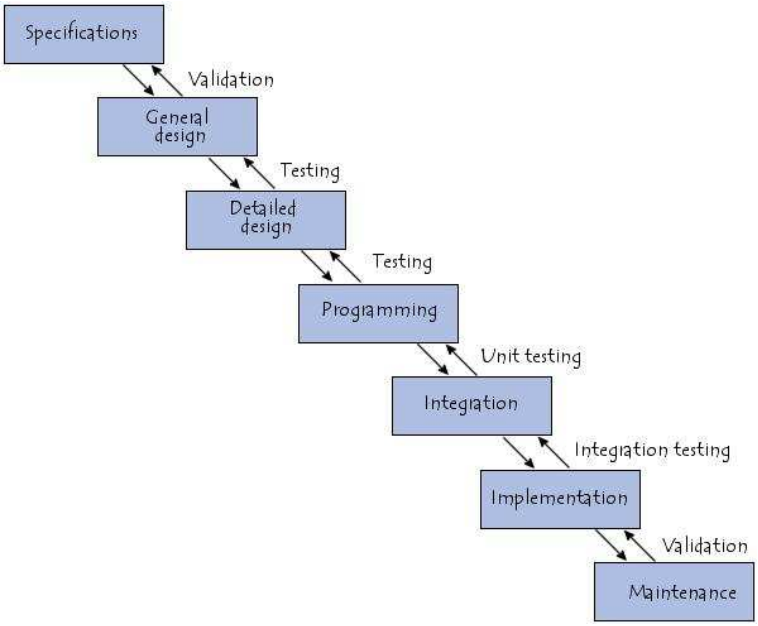


Figure 3.4.: Software development process - Waterfall model

Further, the software is finished earlier for roll-out. The main reasons for this are fewer errors during the project development. The benefits exceed the extra costs for system analysis and design by far. Extremely important is the step right after the design phase. Reviews, evaluation and checks of the system architecture are essential for ensuring the quality of the product and allow to reap the benefits of faster, cheaper testing and maintenance.[32, p.152]

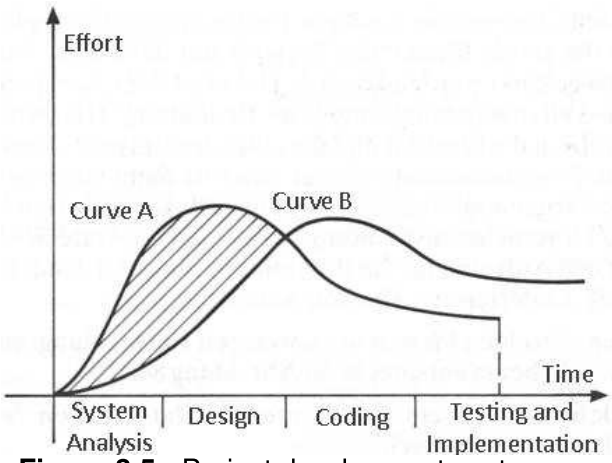


Figure 3.5.: Project development cost curves

3.3. Software design techniques

The next step is to ensure that the software architecture is created extremely carefully in order to avoid any bad decisions, errors and mistakes. This process is related to the overall software design approach.

Basically, there are two ways to design software, top-down or bottom-up. In the first approach, the architecture is created beginning from abstract ideas to concrete solutions. Bottom-up means exactly the converse. Concrete solutions are arranged together until an abstract overall design is built. In industrial practice, developers usually use and combine both methods. If prefabricated modules exist in old libraries, bottom-up is more likely to be chosen and vice versa.

An important aspect of software architecture is the future maintenance effort. Service efforts are the major cost factor within the life cycle of a software product.[33, p.2] Not surprisingly, the stakeholders are interested in designing an architecture allowing future implementation at low costs.

Over time, several examples of software architectures have been designed and have achieved broad acceptance. Primarily, evolutions and improvements in IT science and industrial pressure have led to more and more complex software systems, demanding new approaches of software development and, as a consequence, to enhancements of software architectures. Some are designed for special kinds of software others for specific customers and their requirements. The following list gives a short summary.

Software Architectures	
Blackboard	Client-Server
Database-centric architecture	Distributed computing
Event Driven Architecture	Peer-to-peer
Pipes and filters	Representational State Transfer
Service-oriented architecture	Three-tier model

Table 3.1.: Examples of software architectures[2, 8]

The problem is how to present and discuss software architecture if there is not a uniform set. Searching for common elements, the solution found was to classify them into views. Unfortunately, as there are many reasonable points of views to consider, many examples have been developed and documented. One of the first classifications created distinguished three main elements.[34, pp.40-52]

- (1) processing elements
- (2) data elements
- (3) connecting elements

The processing elements (1) are components which are used for transformation of the data elements (2). These data elements themself store the actual information. The connection elements (3) are considered to be a kind of "glue" holding the pieces of the software architecture together. Similar compact is the later on developed categorization of Hofmeister et al. which separated the architectures into four views.[30, pp.11-12]

Code View

Initially, the source code for the first software applications was stored in one single file. Today, this is simply not possible any more, even scripts often use more than one file. Structuring the code into libraries, artifacts or binary code has been heavily supporting the reusability of the code. Later on, the organization was enlarged with configuration management, version files, directories et cetera.

Module View

Due to the increase of complexity and size, software systems needed more and more human resources for development. In order to maximize effectiveness, interfaces were defined and code encapsulated into modules.

Execution View

Old programs were commonly coded as strictly sequential. In the course of time, software systems became more and more complex, were distributed but also more interconnected. Turning functional components into runtime entities became difficult. Interconnection languages arose in order to cope with this challenge. Soon those issues became strategic enough to deal with them in terms of software architectural components.

Conceptual View

Recently, software architectures have become so complex and extensive that system component itself reached a level comparable to software architectures of whole programs some years ago. Accordingly, this view considers major design elements of the software system.

This categorization is very common and is still taught at universities [35] although there exist several differing classifications of views. In terms of software architecture evaluation, probably the most convenient definition of software architecture is provided by the *IEEE Institute*.[36]

Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

Due to this definition, software architecture primarily deals with its own structure of software

components. It is important to understand and control the essential elements which are essential in terms of utility, cost and risk. The IEEE standard does not explicitly exclude other elements concerning those matters. Of course they can be created of physical components including their relationships.

3.3.1. Describing software architectures

One major problem field of software architecture evaluation (whether scenarios are used or not) is how to get an overview of the functionality of the specific software. Of course, the architecture level is indeed abstract and somewhat simplified. But on a large scale, dependencies in functions, linked components and so on may become quite complex to understand. One way, especially in architecture-based development processes, is to use architectural description languages like ADL or MIL.[37, p.2] Also possible is the use of view models. Most common are documents standardized in UML 2.0. Basic examples are:

- Activity Diagram
- Class Diagram
- Communication Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Interaction Overview Diagram
- Package Diagram
- Sequence Diagram
- State Machine Diagram
- Timing Diagram
- Use Case Diagram

Another possibility of getting an abstract visualization is the use of *Use Case Maps* which are described in more detail in the following section. The advantage of such maps is the level of abstraction they provide.

3.3.2. Use case maps

With UCM (Use Case Maps) there further exists a modeling technique to describe scenario-based software architectures. Originally, UCM was developed as a tool to help understand especially large software systems.[38, p.1] Graphical abstraction is much more effective than skimming thousands or ten thousand lines of code, probably written a long time ago by another programmer. Major application areas are [39, p.2]:

- Requirements Capture
- Architectural Evaluation
- Transformations to Designs and Tests

Basically, this technique closes the gap between requirements (engineering) and design. In the software development process, UCM is applied between defining the requirements and the (detail) design stage. Use cases link and help to visualize the behavior combined with the structure of a software system at the architecture level. Note that message exchange and component behavior are at a lower level. This allows drawing scenario paths within the system visualizing the behavior of the system to new scenarios. It is then possible to study the negative effects of the interaction of several scenarios. Understanding Software is well crucial.

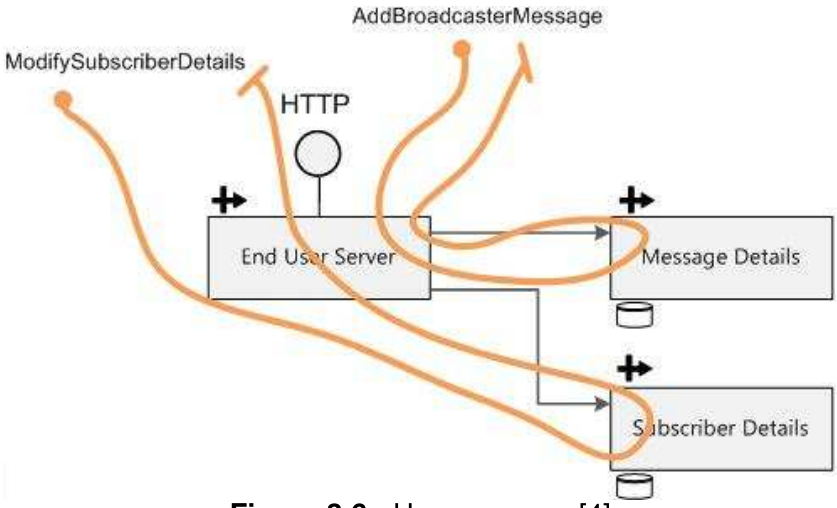


Figure 3.6.: Use case map[4]

Research in the 1980s showed nearly thirty years ago, that understanding a program just through its code represents between 50 to 90 percent of the maintenance work. Today's programs are at least at that level of complexity or most likely above. Exploring architectural problems requires the exact representation of source information. The more abstractly this information is presented, the more easily and faster it can normally be understood. This is the purpose of use case maps. Use case maps can be derived from use case diagrams or from informal defined requirements.[40, p.3]

The image above shows a small and simple UCM. The two lines represent scenarios which can happen to the system.

3.3.2.1. Use case maps - A short introduction

A UCM itself consists normally of path elements as well as of software components.[40, p. 4] In basic path notation, an operator casually links responsibilities in sequences.[40, p.4] Such

sequences can be functions, methods or something similar. Basically these responsibilities alternate, or occur in sequence or parallel fashion respectively, thus enabling pipelining procedures. More advanced operators are also able to structure UCMs hierarchically. Further, they can be used to represent exceptional scenarios and dynamic behavior. Components are used for the description of entities in a system. They can be of different types or nature.

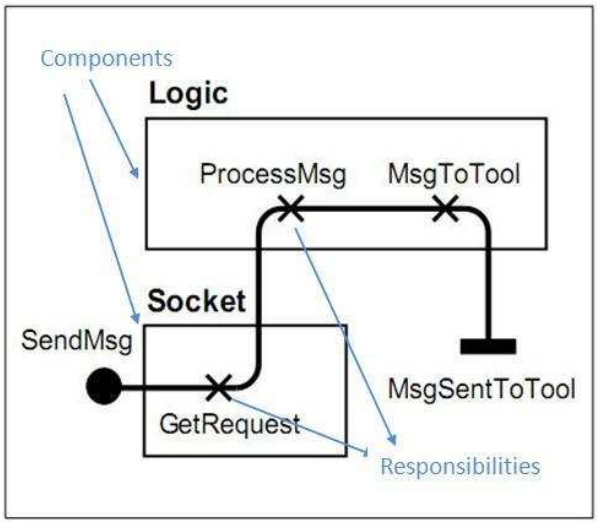


Figure 3.7.: Use case map - example of the notation

Figure *3.3* shows a UCM with two different paths drawn. Each of them represents a sequence of methods. The crosses indicate that the component is bound to the sequence at this point. This component is then in charge to carry out the function, task or action represented by the responsibility. Use case maps are neither a standard of software architecture evaluation nor of UML 2.0. Nevertheless, they provide a very good visualization of the software architecture concerning the functionality, respectively their requirements. This enables the evaluation team to gain better knowledge about the software architecture in less time and helps to come up with high-quality attributes perhaps within a broader range of viewpoints.

3.4. Quality evaluation of software design

However, the best way is to avoid mistakes and errors completely. Although this may be the best method, ironically it is quite impossible to do so. Therefore, tools and methods have been created to find errors as early as possible, which should be even before any coding begins. Hence, software reviews have been introduced.

3.4.1. Costs of error fixing

Basically the objective of software reviews is to avoid costs because of software problems. There are two different kinds of such problems, called bugs, can be addressed in software development. The first to be mentioned are code errors. Commands can be used wrongly, or the syntax of the code contains failures, similar to spelling mistakes in conventional typewriting. The second class of errors are more abstract, often much more complex or complicated and due to these characteristics costlier to repair. The reason is that these errors happen most often during the design and conception phases. A large impact on the future correcting effort needed has the point in time, the error is covered up. Barry W. Boehm, one of the pioneers of cost analyzing of large software projects, has been researching for years in IT-science. Due to his work, the cost of error fixing is rapidly increasing during the progress of development.[41, p.96] Notice that correcting errors right after they were made is not that problematic.

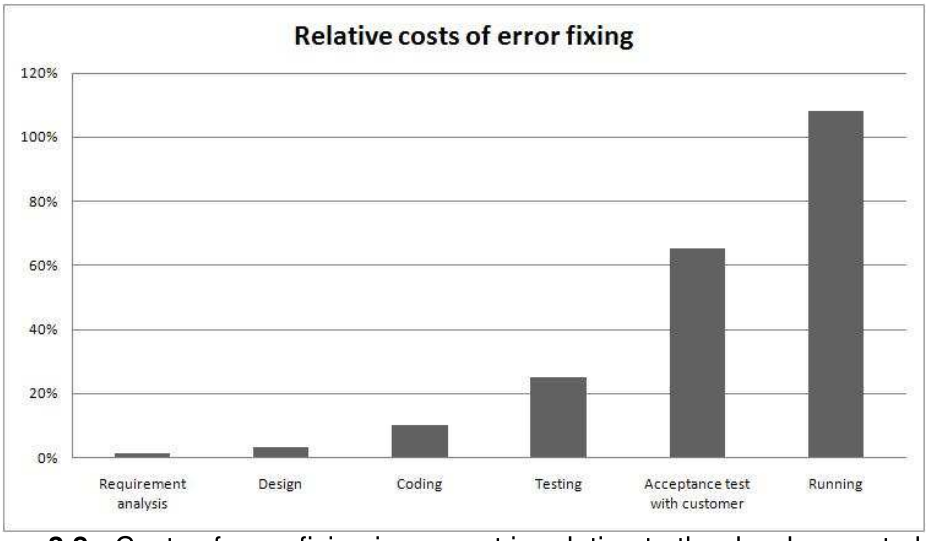


Figure 3.8.: Costs of error fixing in percent in relation to the development phase

The figures used have to be considered as factors. To repair an error at the testing phase is, relatively seen, 25 times higher than the normal working costs. If these factors are allocated a variable of €400, it is astonishing, how large sums become, if errors are discovered late in the process. Getting back to the testing phase, an error costs €10.000 accordingly. Other studies have come up with similar results, strongly affirming these findings.

3.4.2. Reviews

Not surprisingly, people in charge of software development soon realized that controlling this phase is a key issue of becoming much more efficient and sufficient. Scouting out for techniques in order to achieve those goals the software engineers and designers encountered the quality management. (Deming cycle, military, ...). Edward Deming was not the first one using

such working process but one of its industrial pioneers. Based upon those ideas, Michael Fagan, employed at IBM at that time, began in the 1970s to adopt them for software engineering, thereby becoming world-renowned with his article in the IBM Journal 1976.[5]

Basically his invention, the "Code inspection", is a structured workflow split into several sub-processes. The main focus is on finding any defects the development documents including specifications, design, actual code and other stages of the software creation process. Thereby, the inspection follows similar rules as for example quality ensuring methods within the TQM.

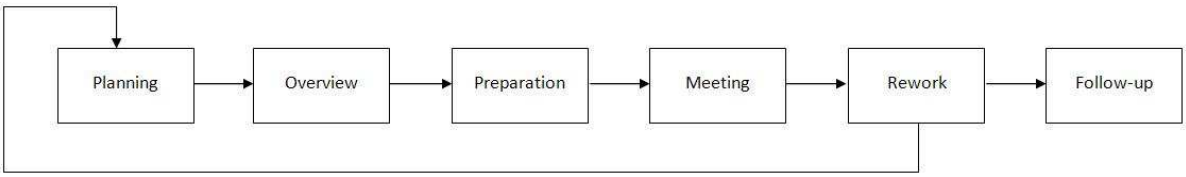


Figure 3.9.: Example of a quality process[5]

Through further research in the IT-science, the "Fagan inspection" was allocated to the taxonomy of review techniques.[42, p.11] Reviews are qualitative methods in order to improve the quality of both development processes and software products. Whereby the latter is the main target. However, there are many similarities between inspection and review methods. Using readable documents the product is checked against guidelines and given norms in order to find errors. Thereby not a single person but a review team is created to do the job. Due to further research and gained industrial experience review methods with or without the presence of the customer(s) have been developed.

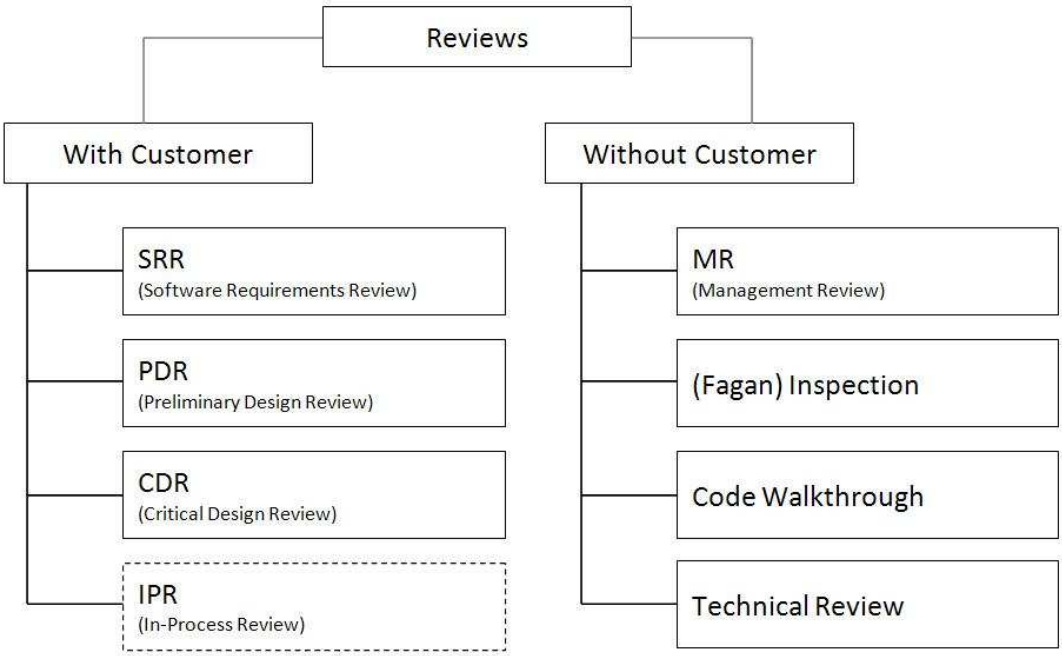


Figure 3.10.: Taxonomy of reviews in software development projects[6]

Throughout several modifications, research, and experience several evaluation techniques have been developed based on the "Fagan inspection". However, the basic idea has stayed the same. Software architecture evaluations are important examples. Somehow they can be seen as successors of Fagan's work. Within this work, special evaluation techniques using scenarios as a fundamental source are of interest.

3.4.3. Architecture reviews

Software architecture evaluation and software inspections are quite similar. Both of them serve to improve the quality of software artifacts.[6, pp.2-3] Some inspection techniques even use scenarios. In contrast to software evaluation, inspection-result reports uncover defects. They are primarily done to find (possible) errors, also in very early stages like the specification or design phase of software development.[41, p.97].

However, software architecture evaluation aims to find quality attribute (change) scenarios for evaluating the software architecture in terms of impacts of future events. Thus, inspections are rather concerned with current problems whereas evaluations try to predict future ones. Nevertheless, both methods are similar enough so that experiences gained during inspections can be transferred to study different aspects of the architecture evaluation process.

One representative of an inspection method is the "Fagan inspection", named after the inventor of software inspections. Applied equally to reviews and most evaluation techniques, this method is used in teams where a specific role is allocated to every member.[41, pp.97-99] It turned out, that this influences the results positively as long as everybody fully understands his role and knows how to act accordingly. Studies prove that "Fagan inspections" reduce costs and improve the quality of the final product. As both methods can be compared to a certain degree it is possible to assume that these results can be applied to software architecture evaluations.

Phase	Error per KLOC at the end of the phase, without Fagan Inspection	Error per KLOC at the end of the phase, with Fagan Inspection
Analysis of requirements	20	5
Design	40	10
Coding	100	15
Module testing	50	7
Integration test	20	3
System test	10	1

Table 3.2.: Average error reduction of the Fagan-Inspection[5]

The main difference between inspections and software architecture evaluation is their specific focus. Inspections are used to find errors in early stages before the actual coding starts or, if possible, to avoid making mistakes at all. So the target is the software system as it is designed. Evaluation of software architecture targets possible changes which then necessitate

adjustments or changes in the software system in the foreseeable future. The architecture of the software is not strictly tested against bugs. Contrarily, the architecture is tested against future events to increase the ability of adjusting the software, if necessary.

Software architectures inhibit or support/enable a system's quality attributes.[29, p.30] Thus bad design and/or lack of care about architecture planning diminish the quality of the software system. Throughout a large number of industrial software projects such negligent procedure is the main reason for cost explosions, which often results in their crashes.

4. Scenario-based software architecture evaluation

Software architecture and its quality management are classical problems of the software development process.[43] Scenario-based software architecture evaluation is a method to deal both of them. However, this technique focuses rather on quality aspects within the development process.[20, p.1] Nevertheless, when performing such an evaluation, feasible architectures together with their specific strong points and weaknesses are the main focuses during the scenario developing process.

The effectiveness of this method heavily depends on the scenarios developed by the stakeholders of the related software development project.[44, pp.59-78] Their ability to create high-quality scenarios is of crucial importance, since the adopted scenarios are the main input to the whole evaluation process. Scenario-based software architecture evaluation has recently become an important tool in software development. But controversial debates have arisen, questioning whether this method has a major influence on the software project. And if it does, it is not clear whether it contributes positive, negative or insignificant effects. Therefore, studies have been done which simulate such situations and their replications shall confirm the results.

In the last few years the main focus has rested on optimizing inspection team meetings facing the challenge that team meetings are expensive and must not be conducted ineffectively [20, p.1]. Unfortunately, only rare efforts have been made to research team meetings that develop quality attribute scenarios. The original study of this replication was, among others things, motivated to explore the effects of lost and gained scenarios within such a meeting. Based on this study, a blind study was performed a year later. The reason for this work was to find out whether the results of the original study can be reproduced and whether a common statement about the effectiveness of inspection team meetings to develop (change) scenarios is valid.

Underpinning the statements in this work so far, studies have shown, that currently 50 to 70 percent of the total life cycle costs of a software system are required for evolving the system.[33, p.1] As already mentioned, considering changes in advance is less expensive to solve than reacting ad hoc. Hence, it is important to consider the changes most likely to happen at the development stage within the software life cycle. The software engineers are probably able to come up with design solutions considering changes in the future. Unfortunately, hardly any techniques exist to find out the level of modifiability of any given software architecture.

Finding a technique for the comparison of software architecture designs with regard to the same view poses a similar problem (see chapter 3.2 “Software design”). The research field of *“Software architecture analysis”* deals with these issues. Nowadays, some methods for such an analysis do exist. The list below shows the currently existing methods [45, p.1]:

- ALMA, Architecture Level Modifiability Analysis
- ATAM, Architecture Trade-off Analysis Method
- CBAM, Cost Benefit Analysis Method
- FAAM, Family- Architecture Analysis Method
- SAAM, Software-Architecture Analysis Method

The list is far from being complete. There exists a pool of other representatives like “PASA”, “ARID” or “ABAS”. Comparable to other methods, the quality attributes of the future software product are evaluated. But in contrast to them, these predictions are made before an actual code is available. As the first efforts to achieve quality are made within the software architecture design, techniques must enable to predict the quality of the system at this important point of the development process. At the design stage, different stakeholders have different views about these matters. It is even more complicated to foresee possible environmental changes in the future concerning the software system.

To deal successfully with both problem fields, software architecture evaluation is based on scenarios.[46, p.2] Normally, these scenarios consider both, current requirements and future effects possibly affecting the software. Why is the use of scenarios meaningful? The best argument is that scenarios are concretized.[33, p.5] So, the impact of the scenario on the software system can be predicted very precisely. Unfortunately, this includes a potential threat to the whole technique: If the scenarios are not of high-quality, the results of the evaluation can be faulty or just wrong.

In software architecture analysis, two methods are used to select a set of scenarios in order to obtain only high-quality scenarios. In advance, equivalent classes are created for the scenarios and in a second step, a classification structure is used within these classes to search for scenarios.[44, p.5] This matches the procedure of this study. Generally, software architecture evaluation is a related technique of software architecture analysis:

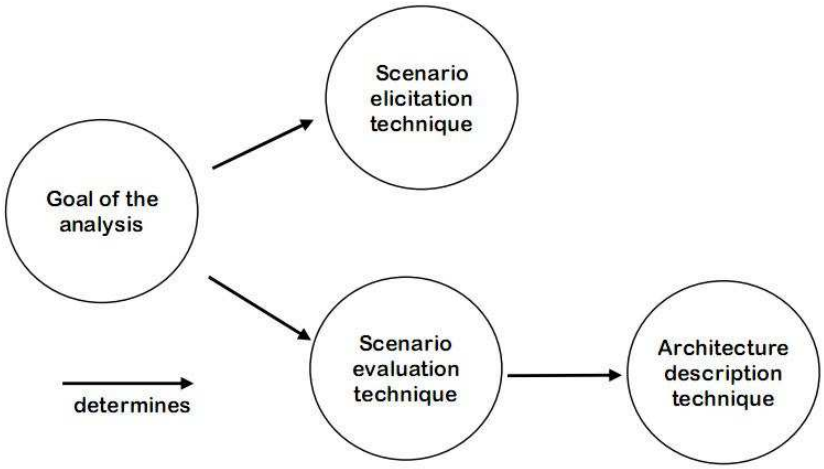


Figure 4.1.: Software architecture analysis related techniques

Most of the methods listed above are refinements of SAAM (Software Architecture Analysis Method), which was probably the first one, or ATAM (Architecture Tradeoff Analysis Method). Probably they are the best-known and most often used ones. Unfortunately, there has been only limited research done in the past on how effective these methods are or if they are even worth the effort.

4.1. Scenarios

Scenarios are used for estimating the probable effects of one or more variables, and from an integral part of situation analysis and long-range planning.[47]. When creating a scenario, one tries to predict certain consequences under given circumstances. As mentioned above, the definition of quality is often in the eye of the beholder. Using scenarios moves the quality definition onto a more objective basis.[29, p.69] Scenario building is heavily used in the area of risk management. During the phase of requirement finding, scenarios are well known and commonly used. Furthermore, scenarios help to compare design alternatives.[24, p.3]

Unfortunately, scenarios used for quality aspects are seldom applied. Even within software architecture evaluation this approach is often omitted. The scope of scenarios differs strongly in breadth and depth. In order to get sufficient results, you can either shorten the scope by explicitly given scenario descriptions or subsume them into categories, which is the approach used in this work. The quality of the found or given scenarios influences the effectiveness of evaluation meetings. But also the costs for meetings to develop such scenarios depend on the requested quality.[20, p.1]

The same problem concerning quality attributes is true for the scenario finding: subjectivity. Every person involved defines his own quality attributes or rates existing ones differently

according to his point of view. Hence, it is recommended to create and adjust scenarios for the different roles involved in a software (development) project. Primarily, these roles imply the stakeholders of a software project. Necessarily, every stakeholder must be represented through an appropriate role since their requirements have to be considered even in the early design stage of the life cycle process.

The use of scenarios forces especially the development team and the project management to estimate the foreseeable future. Which requirements will become obsolete, which will be more important and, of course, what kind of changes may have to be made within the software system? Besides, architectural analysis is not able to handle this matter sufficiently because it demands quality measures like “How scalable is the system?”.[24, p.4] Seldom can a future event be expressed like this enabling numeric answers. As mentioned, quality cannot be defined or measured exactly either. Contrarily, scenarios lead to open questions like "‘How can the software architecture handle these changes?"’.

To be more specific, the type of scenarios involved within a software architecture evaluation is called *“Change Scenarios”* which contains particular events within the life cycle of a software system causing the system to be modified according to these changes.[37, p.2]

4.1.1. Benefits of using scenarios in software architecture evaluation

There are several good reasons and benefits which justify scenario elicitation efforts. Some of them occur during the creation or collection others are covered up during the analysis of scenarios.[16, pp.4-5]

Better understanding of requirements: When scenarios are mapped onto an architecture conflicts and effects of requirements become clearer.

Stakeholder buy-in and shared understanding: Stakeholders can see how their scenarios are implemented into the software architecture. This increases their confidence that their requirements are considered. Further their understanding of the architecture itself increases.

Better documentation: A “walk through” or other documentation are helpful for software analysis. Lacks in the documentation are covered up when evaluation packages are created. Additionally, mapping scenarios on architecture requires an abstract image of it. If the involved people are not able to understand that the level of abstraction might be adequate.

Requirements traceability at the architectural level: The satisfaction of requirements is frequently not attributable to a single software component, or even a small set of components. The best way to understand requirements completely is to map them onto the whole soft-

ware architecture. Treating requirements like scenarios for mapping them is a meaningful solution.

Analysis of quality attributes: When mapping scenarios onto a software architecture helps to find out how easy or difficult it will be to achieve a scenario. Thus, it helps the stakeholders manage risk, plan resource allocation, determine an ordering of activities based upon their achievability, and reason about the trade-offs that inevitably occur when design decisions are being made. The most important impact of analyzing quality attributes is on cost; scenario-based architectural analysis has been proven to uncover architectural faults early in the software development life cycle.

Sure, some of these benefits can be achieved by using other analysis techniques (requirements engineering or domain analysis).[16, p.5] But scenario-based analysis are very cost-effective and help stakeholders to focus on areas where scenarios interact.

4.1.2. Direct and indirect scenarios

A scenario contains somehow an anticipated or desired use of the software. Let us assume any kind of scenario has been created, describing an anticipated change to the software in the future. Two possible situations can arise from of this. The first one is that the system is able to handle the challenges without any modification. The other possible situation is that the scenario problem cannot be solved without a modification which may concern one or more components, or their connections of them. Maybe a new component has to be developed and implemented or a complete substitute for all of them combined is necessary. The worst possible outcome would be the overall replacement of the application. The first case is called *direct scenario* while the second one is called *indirect scenario*.[24, p.4]

4.1.3. Development of change scenario categories

Finding or devising high-quality (change) scenarios is strongly correlated to the stakeholders who are actually developing them.[33, p.11] Experience, practice, know-how, invested time and further engagement of the participants of a scenario based software evaluation are essential factors which result in suitable benefits.

Additionally, the developed scenarios have to be properly documented. Not surprisingly, the amount of possible scenarios for a given software system architecture is nearly limitless. Hence, measurements for sorting, limiting and rating are urgently required. Basically, two techniques have been proven to be especially effective:[33, p.11]

- (1) equivalent classes
- (2) classification of change categories.

(1) can be compared to similar actions applied in software testing. By dividing the range of scenarios into such classes enables treating one single scenario as a representative of one class. This limits the number of scenarios which have to be considered. Unfortunately, this limitation does not render the selection of meaningful or relevant scenarios superfluous. One or more criteria for selection must be created, which is the objective of (2). Thus, attention is drawn to scenarios which will satisfy the selection criterion or criteria. Nevertheless, there can still be too many scenarios left. In addition a stopping criterion is therefore required, thus, the main goal is to come up with a set of scenarios which are both relevant and manageable.

Top-down

At the very beginning, predefined classifications for change categories are given to the evaluation team to help search for scenarios. These classifications can be created out of a domain of interest, from the knowledge of potential complex scenarios or from other external knowledge sources.[33, p.11-12]

Bottom-up

With this approach, the stakeholders have to come up with scenarios without guidance or structural help. It is assumed, that they are familiar with some categorization systems.

The question in both cases is at what point a limiting criterion or process comes into play. Which element will be chosen as a trigger to set the limiting process into motion, deciding that enough scenarios have been found? In fact, the cut-off process has to occur within the categorization system regardless of whether it is explicit (top-down) or implicit (bottom-up). The scenario finding process continues until a sufficient amount of scenarios coverage of the classification is available.

According to Bengtsson et al. both approaches are often combined in practice.[33, p.11] Change scenario classifications are derived from interviewing stakeholders and their resulting scenarios. Afterwards, the defined classifications are used in a second round to find further change scenarios. Enough scenarios are (normally) found if all the change categories are explicitly covered and if new change scenarios do not affect those categories any more.

4.2. Evaluation Techniques

Over time several variations of software architecture evaluations have been developed. Those methods using scenarios are of special interest for this work because such an evaluation was used for the study. Anyhow, an exact classification which method was exactly used is not possible. The experimental environment and constraints required many adaption and adjustments. Therefore the best description would be a composition of the methods SAAM, ATAM and ALMA. In order to get a better idea how those methods work all three of them are explained in the follow-up.

4.2.1. SAAM

Logically, before this method can be used, any architecture must have been designed in advance. In short, SAAM contains four basic steps.[33, p.6] Developing scenarios is the first part of the technique. To be more specific, scenarios which could probably affect the system are collected. Describing and analyzing potential software architecture(s) is the following step. Logically one type of architecture or variants of adequate architectures are envisioned. SAAM implies that all persons involved are capable of understanding the technique. In step three, those scenarios are being evaluated before the results lead into step number four. More specifically, possible effects on the architecture are analyzed. In the final step an overall analysis is made of both, the scenarios as well as the potential architecture is performed. All components of the candidate architecture which are affected by the scenarios are considered. SAAM is focused on scenarios on evaluating software architectures.[48, p.3]

4.2.2. ATAM

This evaluation technique can be seen as the successor to SAAM. ATAM uses quality attributes too, but in a different way. Only a certain number of qualities are considered are at the same time. ATAM works like a framework. Quality attributes are analyzed together in order to find tradeoff points between them. These areas are considered the highest risk factors in software architecture.[48, p.1] Every attribute is considered isolated. Only this ensures that the software architecture is analyzed with regard to components which are concerning and affecting multiple quality attributes. Based upon these findings, scenarios are developed. Afterwards, the architecture can be modified until the best solution is found. ATAM can be split into eight major steps, beginning with zero.[48, pp.2-3]

Step 0:	***Planning/Information exchange***	Meeting to explain ATAM and expectations to stakeholders; gathering their quality attributes and presenting initial architecture and scenarios
Step 1:	***Scenario brainstorming***	Actual start of ATAM; gathering all stakeholders; scenario brainstorming; analysts add scenarios based on their knowledge and experience plus upon quality attributes under review
Step 2:	***Architecture presentation***	Detail presentation of the architecture; normal usage scenarios are mapped for understanding
Step 3:	***Scenario coverage checking***	Selected quality attributes to check proper scenario covering (including boundary conditions)
Step 4:	***Scenario grouping and prioritization***	Stakeholders vote on scenarios; limitation to 10-15 scenarios
Step 5:	***Mapping of high priority scenarios onto architecture***	Architectures check scenario influence (e.g. modifiability) and the response of architecture design (e.g. for quality attributes)
Step 6:	***Performing quality attribute-specific analyses:***	Analysis of the architecture using models based on the architecture's information. By manipulating parameters, crucial points of the design are tested. Sensitive points are correlated with scenarios.
Step 7:	***Identifying trade-off points***	Exploring all important architectural elements with multiple sensitivities to find tradeoffs.
Step 8:	***Consolidating findings and developing an action plan:***	Development of a plan to improve the architecture based on recommendations resulting from the analysis. Probable need of further documentation

Figure 4.2.: The eight steps of the ATAM-Method

If in Step 8, the decision is made to modify architecture, the method returns back to step 1. In a certain this evaluation way reminds one of the principles of the "Deming Cycle" or management principles like "TQM". In literature, ATAM is described as a spiral model (see chapter "Life cycle" in this work) where both, the design and the analysis, are required together.[48, p.3] Without design no analysis is possible and vice versa. ATAM checks software architectures based on quality attributes but not whether the architecture itself is correct.

4.2.3. ALMA

Like ATAM, ALMA is based on SAAM and consequently based on (change) scenarios too. Unsurprisingly, ALMA then again is organized into steps.[37, p. 2] Altogether, there are five of them:

- Step 1: Goal defining: Setting the purpose of the analysis
- Step 1: Description of the (relevant parts) of the software architecture
- Step 1: Developing a range of (relevant) change scenarios
- Step 1: Evaluating the change scenarios; What are their effects?
- Step 1: Interpreting the findings of the analysis

Now, what are the differences compared to SAAM? Dealing with ALMA, the differences between the various techniques used for the analysis are very important. ATAM or SAAM does not necessitate the use of a certain one. The involved persons are more or less free to choose which technique they want to use. Additionally, each analysis instance in ALMA must have only one single goal which distinguishes itself from every other one.

To sum it up, there are some major similarities in all those models.[23, p.2] At least five steps or activities are part of most of them. (1): A planning phase at the beginning is included where the analysis or the evaluation is prepared. (2): The architecture or rather the different approaches are explained, usually to the stakeholders. (3): Quality scenarios are developed. (4): Analyses the design approaches of step 2. (5): The findings are presented, interpreted and evaluated.

4.3. Influencing factors on individual and team performance

There is a difference whether an architecture evaluation is performed as a individual brainstorming session or a team meeting. As individual experience may have an impact on both methods, teams probably can benefit from synergy effects.

4.3.1. Experience influencing software architecture evaluation

Basically the whole process of a (scenario-based) software evaluation, despite of the technique used, is performed by individuals.[49, p.132] In most cases the participants are persons involved within the software project. This includes the presence of the various stakeholders. It is recommended that all involved stakeholders should be fully integrated into the whole process. Consequently they have to join the evaluation sessions/reviews as well.[50]

The purpose of an evaluation session is to find as much errors in the design or architecture as possible. In order to support the reviewers, the review and the related documents have to be design to make it easy for them to do so. If an error escapes the attention of the reviewer, the review itself failed. The problem is that many of them are hard to find because they were made

earlier and it takes a lot of experience and knowledge to find them.[49, p.133] Therefore the evaluation method has to use the maximum of the skills and the know-how of the reviewers available.[49, p.132] Not surprisingly a main source of problems with software architecture evaluation or design reviews is the presence of unqualified or simply wrong people.

Biffl et al discovered that people with industrial experience seem on average perform better during exercises. Further they better understand the theoretical concepts behind architecture evaluations.[6, p.1] Hence, more-experienced stakeholders are expected to achieve better results during such reviews. Either they find more respectively more critical defects or more respectively more important scenarios within scenario-based software architectures. Unfortunately well-experienced people are costly and probably unavailable too.

Obviously project managers are forced to work with less experienced stakeholders. The challenge is to still achieve good results.[6, p.1] Two basic options are thinkable. Either the process of software architecture evaluation is supported actively or review teams are applied in order to get synergy effects.[51],[20, p.3] Probably these effects enable the stakeholders to find additional scenarios.

One possible active support is to guide the reviewers by giving them predefined change scenario categories (top-down approach).[33, p.11-12] Probably this leads to an increase in the number of found errors or scenarios. The same target is focused to achieve from team synergy effects. Unfortunately there is danger that because of rivalry or other factors the teams loose defects/scenarios.

4.3.2. Team size and team meeting benefits

Already within the Fagan's software inspection the team meetings are considered as a key process.[5] As mentioned before experienced and well skilled persons are expensive. Hence team meetings require larger efforts than individual inspections. The point of interest is therefore the benefit or the effectiveness of teams. Are team meetings worth the money charged for? Fagan himself stated that teams are very effective.[5]

Larger team sizes increase the heterogeneity which leads to diverse expertise in IT. Together with further knowledge in various areas this is desirable.[52, p.1] On the other hand the communication requirements increase exponentially with the increase of the time size. Contrarily several more recent research results showed different results.[53, p.1],[54, p.114] Researchers found following potential benefits of team meetings.

Synergy

Basic theory behind this issue is that team meeting dynamics lead to higher defect finding rates. Anyhow, there is only little evidence about this positive effect. Unfortunately recent research work does not affirm this. Changes in the team size seems not lead to higher defect detection rates.[53]. In some studies the teams found only few new scenarios (1 out of 10) compared to individual work or the losses of team meetings exceed the possible gains.[54, p.109]

Correction of False Positives

Recent research findings provide evidence that team meetings are effective in detecting true defects and eliminating false positives.[55, p.306] These findings are supported by Land et al. They report that team meetings are superior to individuals in distinguishing between defects and false positives[56].

Soft benefits

Besides above mentioned points teams allow the share of (review) experience of their members.[57] Further product/system related knowledge can be passed on to reviewers with a lack of it. Probably some kind of team spirit emerges which leads to a collective feeling of responsibility of the product/system and its quality.[54]

However, finding false positives and soft benefits seem not to justify the high costs of team reviews.[6, p.3] Nevertheless the fact that a software project involves many people and a lot of stakeholders do have certain interests and responsibilities related to it, teamwork is inevitable. This emphasizes the importance of exploring those aspects within the context of software architecture evaluation. This work deals like its original predecessor with the topics of team size effectiveness, participant experience and the usage of nominal teams.

5. Replication

The “basement” of any scientific work is (critical) scientific thinking which is based on three things.[58] The first one is to seek empirical evidence followed by logical rationalism and possessing a skeptical attitude. Without those principals, no scientific work would be possible. Empirical evidence can be experienced by others and is repeatable. It is the only evidence scientific decisions are based on. Empirical research studies are necessary to gain such evidence. A replication is basically a repeat of an already existing respectively performed scientific research study.[21, p.1] The main goal of such a repeated activity is to control the original work whereas the replications are considered successful if it was possible to reproduce the previous results. On the other hand, a reproduction can be seen as an expansion of a research study. Two possible ways can lead to such a reproduction:

- scientific improvement - using the same data but new methods
- generalization - using the same methods but new data

So, a reproduction itself requires a successful replication. Otherwise a reproduction would not make a lot of sense, as earlier results have not been proven. There exists also a taxonomy for replications.[59, p.3].

- statistical replication - different sample, but identical underlying model and population
- scientific replication - different sample, different population but similar model (perhaps advanced or adapted)

The second type, scientific replication, is used most commonly and comprises best what researchers normally choose for their work. Replication is one of the basic requirements of science.[21, p.1] McCullogh/Vinoud defined: *“Research that cannot be replicated is not science, and cannot be trusted either as part of the profession’s accumulated body of knowledge or as a basis for policy.”*[22] Hence, work which cannot be replicated (perhaps the author is covering data material) should be ignored! Unfortunately, replications are only practiced in a limited way.[59, p.3] In literature, two overall classifications exist.

Experiments and studies do have the same purposes in the IT-area as in other scientific disciplines. New technology, procedures, methods, etc. are tested and evaluated in industrial

(real) or scientific environments (e.g. laboratory). Regarding IT-science, they are used to improve the software development process (practically and theoretically). Replications are an important part of that work.

In particular both the internal and external validity of a research field are tested.[60, p.1] In short, this helps the research community to create and extend its knowledge about results and observations, and under which conditions they hold and vice-versa. However, replications can either successfully affirm or reject original results. Hence, a replication has to be judged on its own, apart from any previous study.

Exact replication

As the name already indicates, the researchers are conducting a replication by following the original experiment as closely as possible. The main question is whether the same results can be obtained or not.[60, p.2] Notice that the definition "exact" is used gradually. It is quite impossible to repeat the same experiment in exactly the same manner when human beings are involved or tested. Furthermore, this taxonomy contains those replications which modify the pool of subjects or experimental conditions (time, location ...) but keep the original procedures the same.

Regarding exact replications, two sub-categories are described in the literature. One where the researches are following the original experiment as closely as possible and another one where major conditions are changed deliberately in order to address a specific research question more specifically. Example: In an experiment was explored if kids like candy. Now a replication study shall find out if kids specifically like chocolate.

Conceptual replication

Similar to the above replications this variation tries to test the same hypothesis and research questions. The huge difference, however, is that totally different procedures are used. In addition the research variables are often changed completely (e.g. artifacts, population ...).

5.1. Replications in IT-Science

There are several reasons why a study or a replication is launched. Common topics are either the benefit or the effect of a process or a technique on a software product or cost prediction and environmental constraints. Yet, there is often not a big difference whether the study is an experiment or more of an observation.[61, p.1] In the end, in many cases the results will help to build better software in practice.

A controlled experiment in IT-science provides the same benefits as it does in other scientific areas. The proceeding can be well designed and focused delivering significant answers to clearly stated questions and hypotheses. Independent and dependent variables can be used and result in key variables which explain certain phenomena. Furthermore, the relationship of these variables can be investigated.[61, p.2]

However, one single experiment has only restricted significance. Studies must be replicated. Moreover, also changing the conditions (to observe effects under different circumstances), the design and following new or evolved questions from previous studies must be considered. Combining all the results is a reasonable way to build up knowledge of a specific topic. And also negative findings must be integrated as they can help to improve this pool of knowledge.[61, p.2]

The goals of replications are not strictly limited to hypotheses and statistic values. Often researchers try to gain more data to add to a given data set hoping to reach statistical significance so they can prove some effects. Interpreting the enlarged data set researchers can find, merely by chance, subsets which uncover some interesting effects. For this reason and from another point of view this approach is quite disliked. Software engineering practices must have a positive influence. Otherwise no one is willing to invest in them. To change an investor's mind, new techniques and practices must be significantly better in a large range of environments compared to previous ones in use. Searching just by chance, without a definite idea is therefore a questionable procedure. Concentrating on software engineering, two major research goals are of interest:[60, p.2]

1. It is possible to reproduce and/or to test results (implies that the results found in an experiment are true for industrial situations). These replications strengthen the confidence in the results of an experiment.

2. Discovering and knowing the reasons for the variability influencing the results. This is useful in order to find out which project types best fit which techniques. From this, the range of the results can be understood.

5.2. Classification of the replication study

In theory, there exist to different ways of statistical experiments, *"group case research design (or cluster analysis)"* and *"single-subject (or case) research design (or analysis)"*. But this categorization is mostly used in the fields of psychology, education or human behavior. In fact, this replication work demonstrates more typically characteristics of group analyses.[62, pp.5-6]

First, single individuals are observed in the first stage while in the second stage two relatively homogeneous groups are formed and analyzed. Furthermore, the sample sizes differ significantly, from n = 55 in the individual proceeding to n = 2 at the group level. The workload of getting sufficient data is above-average, because it is not that easy to simulate a group of software engineers. Therefore, validating results and statements (hypotheses) is quite a challenge.

Single-case analysis

Two possible variations are conceivable. If the single-case analysis is to only describe and only one single case is to be observed, then a single spot test is adequate. If the analysis is only interested in a few cases, most often the possibility of replicating the single spot test is chosen.

Due to the small size of sample(s), the measurements can be repeated arbitrarily as the technical and organizational aspects are easy to handle. This fact ensures a good error checking factor over time (as more and more single spot tests are performed). Further, a generalization of the results is possible because of this. But the more complex the specific issue gets the less possible are replications.

Group analysis

Contra to the above characteristics, group analysis with repeated measurements uses large samples considered at a few points in time (around 2 or 3). The main reasons for these few measurements are technical and organizational aspects. Further, this implies that the error checking factor over time is limited. Another consequence is that the "mean-reversion-effect" is very strong and cannot be corrected at all or can hardly be corrected.[63] The validity of the results must be tested using a cross-validation in order to be able to interpret them. Performing group analysis is often an elaborate task.

Hypothesis testing

In the original study and consequently in this replication, universal and more generous hypotheses are created and shall be tested. So the problem is how to manage this task with a group case research design.

This work tries to use replication to compare the two studies in an effort to validate the original findings. Cross-validating the questionnaires is quite hard to do as it is very difficult to find enough appropriate participants to check the questionnaires on the one hand, and to do a replication on the other hand. So, single-subject research design methods must be applied.

The problem is that in case of single-subject research designs, only singular or pseudo-singular hypotheses can be validated. Through a gradually performed replication, a single hypothesis can be generalized (to a certain degree);[62, p.12] or, opposite, a general hypothesis can be created and a single hypothesis can be conducted. However, also a group case analysis can be derived from a single-subject analysis.

5.3. Typical replication related problems

Conducting a replication to a previously performed study seems to be rather straight forward. Instead it is very difficult to avoid problems and errors which are coming along with replication work.

The interest factor

It does make a difference who conducts a study, experiment, survey or a replication. Basically, two possible situations can occur. Either the researcher has strong interests in the results or not. And if he or she does, the follow-up question is whether they are wanted to be affirmed or rejected. This decisions affect the creation of hypotheses as well as the whole study proceeding. [60, p.3]

There exists evidence that replications conducted by researchers without a vested interest have less bias and more potential value.[60, p.3] Unfortunately, they first have to be convinced through enough evidence to work on a replication that their engagement is worth it. In most cases, the initial evidence is presented to them by researchers with a vested interest in the resulting outcome.

Documentation

Nevertheless, whatever the goal of a replication may be, a comprehensive documentation of the previous study is quite helpful. Frequently, researchers claim that in papers relevant details and information are often missing or are excluded. So called “lab packages”, a collection of detailed reports, could be a solution.[60, p.4] Of course, only providing or using such a package does not result in good replications. Researchers working on a replication should not just follow the given procedure without examining it. Some mistakes may have been made or the replicator concludes that a dependent replication with minor changes is adequate. Or conversely, he decides to perform an independent replication with a completely different experiment design. However, the replicator might even consider to disregard the original study and to test the hypothesis completely differently. No matter which decision is chosen, such a “lab package” furnishes the replicator with valuable information.

Unfortunately, it is nearly impossible to prevent small errors or slight changes when performing replications. It is all too easy that some slight changes in the experiment design or within the performance itself are introduced. Often those happen without the awareness of the replicators. As a consequence, even solid results can be weakened, diminished or even nullified.

Training

If a process itself, like software inspection, is the investigated object, training of the subjects has an important impact on the results. Hence, even if the same materials are used it is very difficult that two independent instructors can deliver the same training without coordinating their work. But if the trainings cannot be compared then the results/the findings are expected to differ.[60, p.5] If the evaluated process is developed anew, then the "issue of training quality" is moved on to the next level. At this point, any given knowledge is most often not yet introduced to external researchers.

Language

As human knowledge increases more and more rapidly, the cultural aspect of language becomes an issue of growing interest. Especially languages that do not count as world languages have major problems to keep "alive" in modern science. But even in these countries, broken English has become the standard scientific language.[64]

The ability to speak English has become a factor in science to get the chance of doing research. Without any knowledge of this language, it is rather difficult to become a scientist in most countries. Non native-speakers have to deal with some disadvantages when they are forced to work in a foreign-language environment. This may lead to minor productivity, high translation efforts and a higher possibility of making mistakes.

Along with the stress factor, it can be assumed that the output of a non-native speaker in a certain language might be lower compared to a native-speaker. Especially for untrained individuals tasks could even get insoluble due to language problems.

Environment

Researchers might be forced to change the setup of an experiment when repeating it as a replication. Due to a different environment this happens in particular when the replication is performed at another location (another university, city, country, or something else). Not surprisingly, it can occur that guidelines are omitted, the length of activities is changed or artifacts are modified. However, it does not matter whether this happens accidently or not. The results are quite possibly affected.

To prevent these such sources for errors, scientists advise that only a strict, exact and complete documentation like a lab package or a technical report can help researchers to make a valid replications.[60, p. 6]

Of course, sticking to such guidelines only allows for more or less dependent replications. An independent study possesses a more confirming value. Lab packages might help to avoid bias, questionable protocols or procedures, or similar problems related to the replication. Dependent replications further help to reduce costs for re-designing a complete experiment about the same issue. And of course, they help to convince researchers without a vested interest in the topic to run a replication. Since these scientists are probably more aware of the required efforts, such a lab package can reduce those a lot.

After all, results and findings of such a dependent replication are often useful and should not be disregarded. But they have to be viewed with care and discussed, just because the replication was not independent.[60, p.6]

Doing a scientific work necessitates considering regulations and guidelines as well as working meticulously and accurately. Otherwise, the results contain errors or a failure in the design can occur. Consequently, that particular research cannot be used anymore. Replication and reproduction are methods to control and check scientific output.

Problem/Error	Verifying Method
Error in sampling	Reproduction
Errors made by scientists	Reproduction
• Faulty data collection also: interview faking	Reproduction
• Faulty data evaluation	Replication
Imitation	Replication
• Manipulating results	Replication
• Omission of differing data	Reproduction
• Changing of differing data	Reproduction
• Faking data	Reproduction

Table 5.1.: Common errors in scientific work

As displayed in table *5.1*, both methods are used to validate different aspects. Sometimes replication and reproduction are used together, as for example, to cover up errors caused by the authors or scientists. The number of wrong scientific statements proven to be false through those methods is high. Normally, a negligent or inaccurate work method is the main cause for publishing wrong results. Unfortunately, scientists are often put under pressure, particularly by their stakeholders. Sometimes yet often enough, incorrect results and statements are published on purpose. Honor, fame, the prospect of a lucrative job also affect and lure scientists. Especially this is the case, when the chances of being checked are minimal. To be fair, the more promising, interesting or important a research field is, the more likely and another one is doing research work too. Nevertheless, science requires replication and reproduction in order to explore new areas and to prove the validity of new findings in general.

In vitro to in vivo

Most problematic is the fact that throwbacks occur, thereby increasing the costs of scientific work. Hence, many studies tend to deal with very limited models relying on small sample sizes, few and small artifacts and other resource constraints (e.g. time).[61, p.3] This then poses one of the biggest threats to achieving validity of an experiment and is, at the same time the step from in vitro to in vivo. Although scientists and researchers try their best to achieve satisfying designs, it is all too easy to make a mistake or to miss an important point. Subjects might be "contaminated" and there are always some kinds of learning effects on both sides: researchers and subjects. Running an experiment twice can produce different results just because of the people involved.[65, p.1] A key element in a well-executed experiment is efficient teamwork between the researchers - these should be several -and the people (as many as possible) who review the design.[61, p.3]

Communication of information

To run a replication in software engineering is rather difficult. One of the main problems is shared with similar set-ups in other science areas. Quite often it is impossible to get an equal setting like at the original experiment/study.[66, p.1] Thus, changes and modifications are necessary. A second important issue is the transfer of knowledge by the original researchers to the replicators. Information in the form of "lab package" or "replication package" is important but also the direct communication between researcher and replicator is of value.

Fundamentally, a replication package is vital for a successful replication (knowledge).[66, p.3] On the other hand, occasional communication is insufficient.[66, p.4] For a proper communication a couple of meetings is necessary.[66, p.10]

Lack of replications

Important for an experiment or a study is the fact that the results must be measurable somehow. So measurements have to be created in advance. Additionally, the findings have to be interpreted afterwards. This requires a kind of "framework" which allows an interpretation of the data.[65, p.1] Today researchers involved in software engineering claim that too few empirical evaluations are carried out. Researchers investigating this topic agree with them.[66, p.1],[67, p.1] There is a huge amount of studies which is isolated and not replicated. Moreover, Zannier et al. found a remarkable absence of replications of studies and experiments within the IT-science.[65, p.1]

Throughout this study, some astonishing facts appeared. Around a third of the studies or experiments have not been replicated at all. Self-evaluation is quite a rare practice in IT-science.[65, p.2;5] Furthermore, the findings show nearly a complete absence of negative results of studies.

These data lead to the question whether those findings are always realistic and how efficient the applied review processes really are.

Compilation of results

After reviewing the results of several experiments and studies, the successful aggregation of the results of scientific studies with their replications is quite impractical and disappointing.[66, p.1] Only if the experimenters combine the results with own replications significant results could be found. So one key element of a successful replication is to design the setting as closely as possible compared to the original one.[66, p.1]

Unfortunately, many conditions are nearly impossible to copy. For an example, it is very unlikely that identical subjects can be found or that the same resources are available (e.g. time, subjects, environment). Concerning the subjects again, their knowledge about the technology or technique evaluated is most likely not of the same level and also some other qualifications may be different- if not worse in some cases.[66, p.1]

Human subjects

Several issues have to be taken into account regarding human subjects in scientific research. Basically, people can take over two fundamental roles. Either they are the researchers conducting the experiment respectively the study or they are participants or subjects of it. Following threats to validity have to be taken into account. The first on is the variability of human behavior. The difficulty of isolate confounding factors is also a major threat. People can be disturbed by many things like things are the environment, the working conditions or other participants. The bias of researchers is a problem to validity too.[67, p.1]

Experiments with human subjects can lead to highly variable results. Reasons for these different results are the attention of the participants, their skills and knowledge, their prior experience or their motivation and expectations. Because of that, replications are urgently needed. Unfortunately, as already mentioned, exact replications are nearly impossible to achieve. One solution is to conduct a theoretical study or replication with adoption to the population and their characteristics.[67, p.2] In terms of human subjects factors like education, time, nationality and cultural background are inevitable issue leading to replication problems. Furthermore it is very difficult to find skilled subjects.[67, p.2] In the end it is more fruitful to perform a theoretical than a literal replication as there are too much variables which are incalculable.[67, p.10]

6. Research approach

This chapter deals with the design, planning and execution of the study replication. The focus is on the guidelines of the original study. As it is the case with many replications, it was impossible to carry out an exact copy. Making adaptations and minor modifications was inevitable. Environmental conditions and restrictive circumstances forced the replicators to do so [see 7.1 “Experiment description”].

Different environments as well as different lecturers, another university in another country, et cetera affected the planning and design. But this had not only negative effects. For example, it was possible to recruit more individual participants of two separate modules for the replication. Further, two software systems were used enabling two complete study rounds.

6.1. Empirical study and replication

The basis of the whole work is an empirical study performed at the “Vienna University of Technology”. Main goal was to gain and to confirm knowledge of scenario-based software architecture evaluation. Participants of the experiment were students visiting two different subjects of computer science. They were randomly assigned to two study groups. One was using a top-down approach using predefined scenario categories. The other group performed a bottom-up approach which means that they had to find scenarios without any help. In general, most of the design of the study was adopted from its predecessor. Further details are described in section.

Due to the fact that a previous study had been performed this one served as a replication of that. Replication of research work is essential in science. Every result, finding and statement in the scientific environment must be repeatable in order to prove them. This puts scientists in a position to evaluate scientific work in order to avoid that falsity is spread misleading or diminishing further research work.

6.2. Research hypotheses

Being the replication of an already existing study, it is the purpose of this study to verify the results of the original one. Accordingly, its main objective is to find out whether the originally proposed hypotheses hold up to the findings of the replication. Therefore the hypotheses are used again in this work[17, 20, 19].

Informally, the big picture is about scenarios and the effects of the support of (change) scenario categories:[17, p.2] *"The usage of domain specific categories of software changes can help stakeholders generate better quality scenarios that characterize the future changes in a system."* However, there are further factors investigated. Experience may have an impact on the performance of an individual. Probably the size of the team as well has positive or negative effects too.[6, 19] Out of this all, three basic research questions/topics have been created.

6.2.1. Impact of change scenario categories on the scenario quality

Do change scenarios categories help find more or better scenarios? This question involves two possible configurations. Either an individual person can be supported by such categories or whole teams. According to this, following two different hypotheses have been created.

H1.1: *The provision of software change categories will not improve the quality of scenario profiles developed for software architecture evaluation at an individual level.*

H1.2: *Teams who are given the software change categories for use in scenario elicitation perform similar to teams who are not given the software change categories.*

It is expected, that both hypotheses can be rejected. Scenario categories should help especially less experienced individuals and teams. At the beginning they probably give them some ideas to start with and avoid any late beginning. Further they support a broad range of scenarios as categories might help the individuals or teams to think about many aspects of the evaluated software architecture.

Possibly, the availability of scenario categories does help persons or to come up faster with scenarios and/or enables those to produce better ones. The study design includes a two-stage individual scenario development process to get two different result-sets of individual created scenarios or team created ones. Based on these findings, the hypotheses are checked. The process is explained in detail later in this chapter.

Anyhow, referring to the teams there it is possible to evaluate the calculated results with nominal teams of the same size. As only few (real) teams existed, it is meaningful to use a larger data set, even though the additional teams and their results are notional.

H1.3: *Nominal teams who are given the software change categories for use in scenario elicitation perform similar to nominal teams who are not given the software change categories.*

The same expectations made for the individual and real team are valid for the nominal teams. This is because they are completely based on the individual results. Again the process of creating these nominal teams is explained later in this chapter.

6.2.2. Impact of experience on the scenario quality

Does experience help to find more or better scenarios? Contrarily to above research question the factor experience is only be considered at an individual level. But there are too few real teams to get sufficient results in order to make predictions. Further the comparison with nominal teams is not meaningful out of the same reason.

H2.1: *Extensive and less experienced participants will identify a similar number of scenarios.*

As software architectures can be quite complex and complicated it requires a lot of know-how and experience to be able to evaluate one professionally and to achieve good results. Out of that reasons it is expected that extensive experienced individuals perform better than less experienced ones. However, according to the original study it is also tested whether average individuals or teams perform similar as experts. Underlying of course this involves that experts do have significantly more experience.

H2.2: *The number of identified critical scenarios is similar for expert ranked and frequency related scores.*

It is quite difficult to make any assumptions here. Therefore the original study is the basis for the prediction of the results. The number is expected to be similar and that the hypothesis cannot be rejected.

6.2.3. Impact of team size on the effectiveness of scenario development

Does increasing the team size help find more or better scenarios? Important in terms of review cost (an evaluation still is a kind of review) is the number of persons involved. The smaller the team is the cheaper the whole procedure. Nevertheless, on single individual will surely find fewer scenarios than a group. Target is to find that specific team size where the gain of efficiency is the biggest compared to the smaller team size before. Unfortunately, no data sets of comparable teams with various sizes are available. Actually it would have been impossible with the given resources to do so. To overcome this problem again nominal teams have been created based on the individual data sets. A detailed description of this proceeding is given later in this chapter.

H3.1: *The number of identified scenarios is similar for all nominal teams independent of team size.*

As more team members will most probably lead to more scenarios found it is expected that above hypothesis can be rejected based on the calculated results.

6.3. Variables

These hypotheses were tested by applying several statistical calculations. As in every regression, one or more independent and one or more dependent variables are needed. The independent variables in this study were:

- The domain specific change categories
- The prior qualification of the individuals
- The size of the nominal teams

The domain specific change categories, acting as one of the independent variables, were provided to the treatment group whereas the control group did not receive them. The qualification of the participants was based on a calculated index which regarded all individual qualifications. Now the frequency of each scenario found served as the dependent variable. The frequency of a scenario is determined through:

- *Individual brainstorming*: Amount of scenarios per individual and, in the case of the treatment group, also per scenario category recorded on an individual scenario list.
- *Team meetings*: Amount of scenarios per team (three-persons) and, in case of the treatment group, also per scenario category. Findings are recorded on a combined team list.
- *Simulated team meetings*: Amount of scenarios per nominal team (three-persons). These teams are formed with the help of a random generator, but the persons do not meet. The scenarios of each team member are pooled with the ones of the other individuals in the group. This way, a scenario list is created by each team.

Comparing individual brainstorming and team meetings (no matter whether real or nominal) with regard to scenarios, there are two possible options: A scenario can be created within a team meeting or it can be lost. In other words, within the team meeting a new one can be developed which did not exist on any individual scenario list. Or, the other way round, a scenario which showed up on one of the individual's lists is not selected for the combined team scenario list. It is important to point out that two different systems allowed two different runs of the study replication. Within those rounds the above mentioned variables existed. Furthermore, the systems themselves act as an additional variable when comparing the results of both runs.

7. Experimental Process

Apart from the above mentioned problems, the main target was the replication itself. It was decided to design, plan and execute the whole procedure as exactly as possibly compared to the original study. This involved an introductory lecture, an individual brainstorming session and team meetings. However, several constraints (e.g. environment, timetable) as well as some adjustments, details follow later on within this work, were inevitable.

Due to these modifications the individual experience, the scenario ratings and other factors concerning the evaluation had to be treated slightly differently. Explanations about details concerning these aspects are also provided in this chapter. Given the guidelines of replication, of course the hypotheses are similar to the ones of the original study. This is absolutely necessary to be able to compare the results. What is the aim of such a replication? The basic target question and follow-up questions pursued are:

Was it possible to reproduce the original results using the data set of the replication? If they are similar, how strongly do they affirm the statements made? If they are not similar, what factors may contribute to the differences? In order to answer both secondary questions, the thematic replication in IT-science is explored if and what kinds of problem fields exist in terms of general and specific IT viewpoints.

7.1. Experiment design

Basically, the whole study was designed as a two two-stage scenario process according to the guidelines of.[20, 44] At the beginning, each participant had to construct a scenario profile (a set of scenarios) on his/her own. In a second step, individuals were selected into teams in order to put the findings together to a group scenario profile.

In order to achieve two separate but comparable data sets, the whole proceeding had to be done two times, each time on a different platform - Wiki and LiveNet. So everybody had to take part in two individual brainstorming sessions and two team meetings. By providing flexible schedules it was hoped to get as many participants as possible.

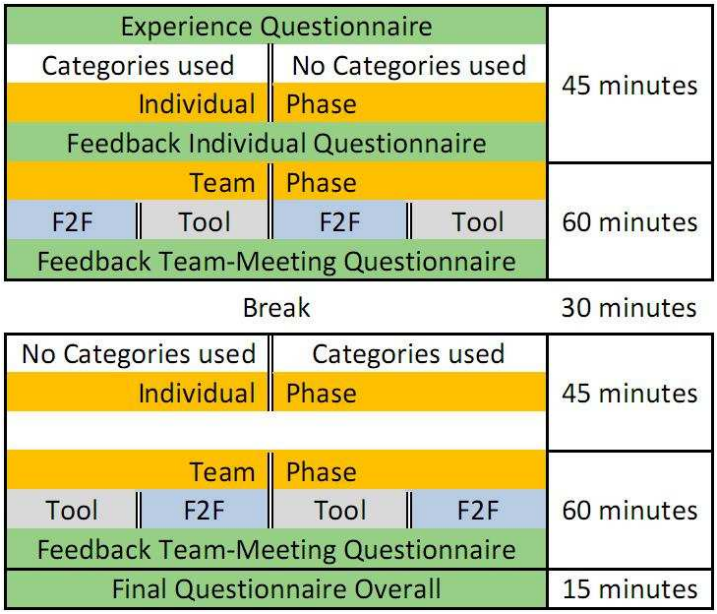


Figure 7.1.: Basic study setup

Due to timetable and participant schedule problems, the experiment was run on two days. The participants were split up between the two experiment runs. All other experiment conditions stayed the same. These measures were taken to provide the same circumstances as exactly similar as possible. Unfortunately this design involved one serious problem. As participants switch the study groups, learning effects probably occurred. The reasons for this as well as further details about that topic are described later on.

7.1.1. Study procedure

In the run-up of the actual experiment performance, everybody received the same initial lecture introducing the topic of software architecture evaluation meetings using scenarios, some examples as well as explanation about the procedure of the study. The lecturer was Professor Muhammad Ali Barbar from the University of Limerick, Ireland who does research in this area and was responsible for the initial study at the University of New South Wales, Sidney.

Then, the study materials were handed out (see appendix). These materials differed in terms of group affiliation. The participants were randomly assigned to the two study groups - the treatment group or the control group. To prevent group building because of friendships, the selection was done in the initial phase. One group, the treatment group, received additional predefined scenario categories. This was to help the participant to find scenarios which he then had to classify into the given categories. The control group did not receive this helping categorization.

So, the participant had to find all the scenarios without any hint from given categories. Furthermore, he or she had to specify the categories on their own and assign his or her found scenarios to them. After a short break, the participants had to group themselves according to their group assignment and hold a team meeting. To prevent falsification, no mixing of groups was allowed. Teams were composed only of individuals who had either been supported by (change) scenario categories or not. Each team consisted of three team members.

Afterwards the teams had to create a team scenario list. They could use the scenarios they had found or develop new ones. Two different meeting styles had been organized. One half of the teams met face-to-face and could discuss their findings verbally. The other half met virtually, using a software system for group discussions.

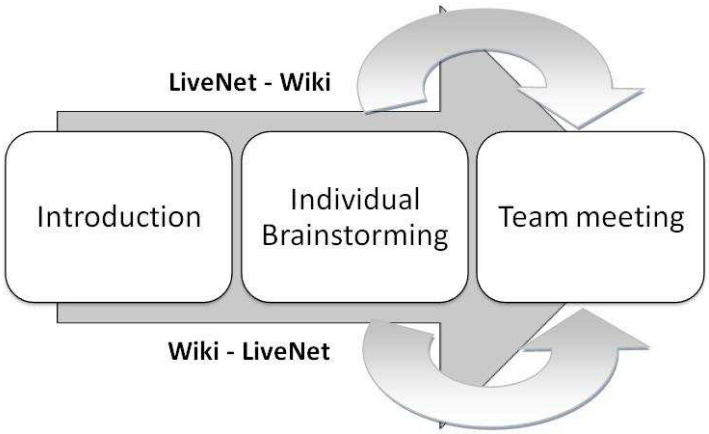


Figure 7.2.: Basic study procedure

Now, in the first run, these two stages were performed at one of the two given software systems. Both, the first and the second stage occurred on the same system. As already mentioned, the first system used was a Wiki-System and the second one the LiveNet-System. Both systems will be explained later on.

A balanced design was of utmost importance. Both groups were supported to be of the same size in order to be comparable in terms of the results. In the second run, every single participant changed to the other software system, performing both the individual brainstorming and the team meeting again. Also, the groups switched. If the student had been given scenario categories (treatment group) in the first run, he now had to work without them and vice versa.

As mentioned above, learning effects have been worried. The participants allocated to the treatment group in the first run worked with scenario categories. The problem was that when the study groups were switched that these individuals would remember the scenario categories they had used before in the second run. That would negatively affect the results of the replication. Nevertheless was decided to keep this study design.

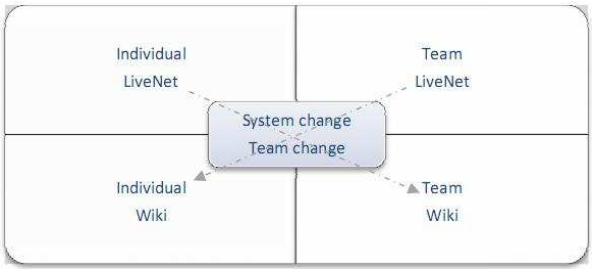


Figure 7.3.: Software and team setup change of the participants

At the end of each task, the participants had to hand in their results. For the individual brainstorming, this comprised the individual scenarios and for the team meetings, the group had to deliver the merged team scenario list.

7.1.2. Study schedule

The timetable as well as the locations were the same in both sessions. This ensured that both sessions provided the same environment and the same conditions for all the participants. In order to combine all the data for the evaluation of the study, this was necessary.

Study Schedule	
Duration	**Activity**
30 min Lecture Room	▪ Introduction / Repetition of the practical exercise ▪ Distribution of the study material ▪ Collecting of the individual background and experience
30 min + 15 min extra time Lecture Room	▪ Individual Scenario Brainstorming following provided guidelines ▪ Collecting of individual feedback on scenario brainstorming
60 min Lecture Room / Information laboratory	▪ Team Meeting (two different meeting styles) o Group 1: Face-to-Face meeting at the lecture room o Group 2: Virtual team meeting at the information laboratory (Tool) ▪ Collecting of team meeting feedback
30 min	BREAK
30 min Lecture room	▪ Individual Scenario Brainstorming following provided guidelines ▪ Collecting of individual feedback on scenario brainstorming
60 min Lecture Room / Information laboratory	▪ Team Meeting (two different meeting styles) o Group 1: Face-to-Face meeting at the lecture room o Group 2: Virtual team meeting at the information laboratory (Tool) ▪ Collecting of team meeting feedback

Table 7.1.: Study schedule

The only difference in the two sessions was the introductory lectures. The speaker at the second session was different because of time and geographical constraints. But as the complete presentation material as well as the complete study materials and the presentation itself were exactly the same, this should not have had any positive or negative effects. Furthermore, the primary speaker, Muhammed Ali Babar, did not join the replication proceedings regardless of the session.

The entire individual brainstorming sessions actually lasted 15 minutes longer than planned. More time was needed to answer all the questionnaires and due to minor language problems some individuals required more time to read through the study materials.

7.2. IT-Environment and study materials

Like in most studies a study package was required too by this specific one. It was necessary to give the participants enough information to fulfill the tasks, to gain information about the participants and feedback from the participants and to get the results of the study.

The first part contained several questionnaires which were necessary for capturing important data about the participants. Within the following part an introduction, theoretical background, the study guidelines and the study tasks were included. The last sheets handed out contained the team allocation, the data capturing sheets and the feedback questionnaires.

Additional to this package two collaboration tools were needed out of two reasons. The first one is necessary to provide the students a platform to use during the study. The second reason is to provide examples of software architectures to evaluate.

7.2.1. Software systems

Using two different and separate systems enables to get two comparable result-sets. Further, a combined result-set can be created. Of course, the validity is lower because the software systems are not completely comparable and in the second run, the control group may remember the given categories and have an advantage, respectively is not free of minor influences.

LiveNet

Principally, the program can be described as a web-based collaborative tool. It is very simple, helping the users to discuss topics via several communication platforms like chat, forum, or email. Small files can also be up- and downloaded. A detailed description can be found in.[68]

Supported are synchronous (chat) and asynchronous (forum, ...) ways of communication between one or more discussion participants. The user is allocated a specific role involving different access rights. Accordingly, it is possible to create, schedule and assign tasks to other users. Furthermore, documents can be uploaded, read, changed and downloaded according to the user's rights.

The participants of the study replication were introduced to the system as an industrially used tool. They were registered according to their role within the team. Based on this, the task was to communicate with one of the given communication tools, creating documents and to try as many functionalities as possible. Drawing on that experience, the teams were told to create team scenario lists (using their individual or new change scenarios).

Figure 7.4.: LiveNet starting login screen

Wiki

In this study, a copied version of the program *Wikipedia* was used. Its modus operandi is exactly the same like the original one known from the "world wide web". The program simply can be described as an encyclopedia. People can place their contribution in the form of texts or graphics or other multimedia forms which then can be discussed in communication functions like the chat room, a forum or similar modes. A description of the system can be found on the homepage of the original portal: *www.wikipedia.org*.

7.2.2. Study materials

Two different and separate packages of study materials existed. Both contained basic introductory information and general guidelines. Further, a description of the study, its schedule and tasks were given to the participants. Primarily, the packages differed in terms of the software systems. One package included the description and information concerning the LiveNet-System, the other one for the Wiki-System.

Then, these two main packages were separated a second time depending whether the participant initially worked with given categories (treatment group) or without them (control group). Since individuals changed their group within the study proceedings, they had to hand in their initial, personalized package and received another one, depending on their following tasks (group selection and system).

7.2.3. Questionnaires

Several questionnaires and a data collection sheets were the main source for gathering all the required information of the study replication (including the individual participants and the created teams).

Experience Questionnaire

It was handed out at the beginning of the whole study replication whose basic goal was it to get an overall picture of the students experience. Based upon the given answers the, individuals were classified into the categories A (well experienced), B (experienced) and C (less-experienced). Due to this classification, it was possible to get more detailed information about the individual performances by using the evaluation of the results.

Feedback questionnaire - individual brainstorming

After the individual session this follow-up questionnaire was handed out to get the opinion of the participants about the organization and the design of this part of the study replication. The questions pertained to the timing, the study material package, what was considered good and what could be improved, and so on.

Feedback questionnaire - team meeting

Since two different meeting styles existed - face2face and tool supported -, also two slightly different questionnaires were developed. Hence, the focus of the questions was basically put on the quality of the communication. On the one hand, the topic was direct communication between the team members and on the other hand, the communication supported by collaboration software.

Final questionnaire

On finishing the complete procedure, the participants were given this last questionnaire. They had to answer the question in which meeting style they had probably performed better. Additionally, the group performance and group cooperation had to be ranked. This helped to compare the actual results with the self-estimation of the participants.

Data capturing sheet

On this form the participants (individual and team) had to write down the found scenarios together with a short description. Further, they had to rank the scenarios in terms of their importance and their likelihood of happening. The ranking ranged from A to C, i.e. from critical/very likely to less-important /unlikely.

7.2.4. Change categories

Due to the fact that this work is a replication and therefore the environment, the study design, the parameters, etc. are the same (at least to the most part) the change categories have to be the same as well. In the original study, the categories were established by the researchers and the developers of the software system LiveNet together.[17, p.4] As the same software systems were used for this research work the categories stayed the same too:

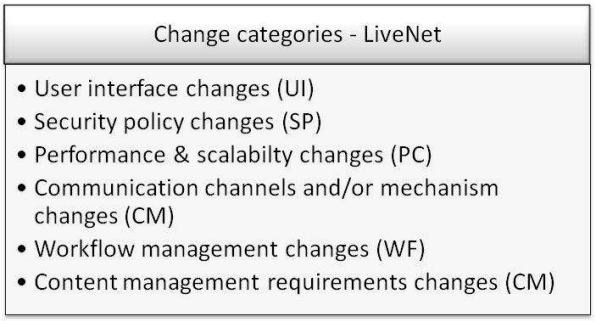


Figure 7.5.: The change categories provided to the treatment group - LiveNet

The user interface (UI) comprises changes concerning the applications used for the user interface. Necessary changes in the security policy category (SP) pertain to all the applications involved as well as to the content of the system. The category performance and scalability deals with changes required to improve the performance of the system(s) (PS) for a faster response time or to handle more users without a drop in performance. Workflow management (WM) supports the various business processes in a company and includes all the changes concerning them. Last, Content management (CM) covers all the aspects dealing with content management itself.

7.3. Scenario rating

Basically, scenario prioritization can be done either by experts or is based on the frequency of a given scenario. If scenarios are rated by (external) experts, higher efforts for categorization and classification have to be considered mainly because they are costlier than internal

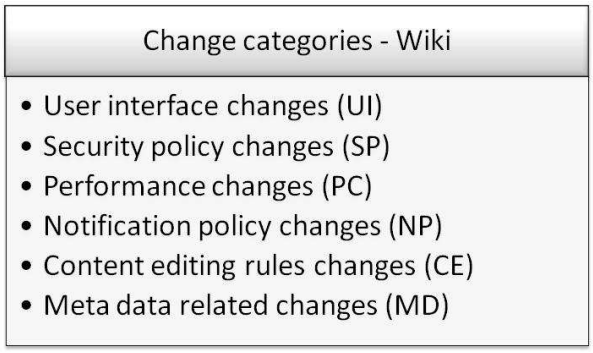


Figure 7.6.: The change categories provided to the treatment group - Wiki

employees. In accordance with the original study, the scenarios are prioritized in both ways. Moreover, a third scenario prioritization was performed by the individuals themselves. This was not proceeded within the original study. The reason for this was to find out whether the individuals rate scenarios similar to the frequency they found them overall.

7.3.1. Scenario classification

To actually be able to work with the individual scenarios, they had to be made somehow "comparable". Some scenarios are more important or critical than others. The problem is how to define which one is more or less important than another one. In accordance with the original study, scenario marking profiles were created. Furthermore, the performance of an individual or a team (both real and nominal) can be measured as well using such profiles.

These profiles act like a rating scheme/system and are based on the frequency of the scenarios. If a scenario is developed, found and frequently chosen, its importance is rated higher. So, often found scenarios are rated higher than the ones picked less often. All in all, a classification which uses three graduations was created: A, B and C. Thus, depending on their frequency and in accordance with the original study, class A implies roughly the top 20 percent of found scenarios. Class B contains the next 40 percent and the last 40 percent are classified as C. Logically, the order within every classification descends from the most to the least often found scenario.

Individual classification

Diverging from to the original study, this one offered the participants the possibility to rate their scenarios on their own (at an individual and a team basis). Two different criteria, *importance* and *likelihood* were used. For both, the ratings (A), (B) and (C) were available. For example, a scenario importance rated (A) expresses the scenario to be critical or very important. A classification going from (B) to (C) represents minor importance. The same ap-

plies to likelihood: A (C)-rating means the scenario is expected to occur seldom in the future whereas (B) and (A) express a higher probability that these change scenarios might happen in the future.

7.3.2. Scenario rating based on scenario frequency

The importance of a scenario is rated according to its frequency. The overall number of scenarios is taken into account. Analysis and calculation regarded all unique scenarios found. As described in this chapter, section 7.5.1 “Scenario Classification”, all of them are divided into three main categories (A, B, C). The input data contain both individual and team found scenarios. Further, the achievements of individuals, teams as well as of the nominal teams are measured. It should be remembered, that the real teams and the nominal teams always consisted of three team members.

Totally, 367 unique scenarios were developed, respectively were found. 174 of them are associated with the LiveNet-System and 193 belong to the Wiki-System. Not surprisingly, most of them were found during the individual brainstorming. During the team meetings (F2F and tool supported), only few scenarios were additionally created. In the original study, they were also classified according to their frequency. The top 20 percent fall into class A, the following 40 percent into B and the last 40 percent into class C. Unfortunately, this approach is not applicable with the data here. There are too many scenarios which show the same frequency. If the original approach were applied, the scenario range for Class A would reach scenarios with a frequency of two. As a consequence, no meaningful differentiation would be left. Instead, both scenario lists (LiveNet and Wiki) were compared with the frequencies of their scenarios.

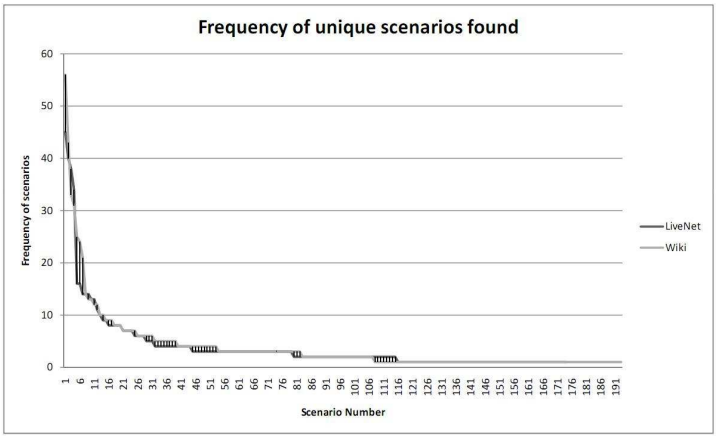


Figure 7.7.: Frequency of unique scenario found

It was decided not to just simply use the frequency for scenario ranking. Instead the scenarios developed during the team meeting have been valorized. If a scenario is created by a team, minimum two team members have to agree to put it on the team scenario list whereas during

the individual brainstorming only the opinion of a single person is arbitrative. Therefore following “formula” for calculating the frequency of a scenario was used:

- The frequency of the scenario of the individual brainstorming plus two times the frequency of the scenario of the team meeting yields the overall scenario frequency.

Interestingly, both distributions of frequency of both systems are very similar. This is also true for the values of the means (LiveNet: 4,14 / Wiki: 4,15) and for the medians (LiveNet: 2 / Wiki: 1,5). However, it was decided to set the range of frequencies for class A from five to the maximum (both systems). Class B contains all scenarios with a frequency including two, three and four for the LiveNet-System, and also two, three and four for the Wiki-System. All scenarios which were found only once were selected for class C. Overall, 19 percent fall into class A, another 41 percent into class B and the remaining 40 percent into class C. As can be seen from figure *7.11* as well as from table *7.5* in this section, the participants found an equal number of critical scenarios as well as more important scenarios within the Wiki-System.

Class A		Class B		Class C		Total	
LiveNet							
No.	%	No.	%	No.	%	No.	%
31	18	84	48	59	34	174	100
Wiki							
No.	%	No.	%	No.	%	No.	%
39	20	68	35	86	45	193	100
Total							
No.	%	No.	%	No.	%	No.	%
70	19	152	41	145	40	367	100

Table 7.2.: Total scenario classification

Not surprisingly, the amount of scenarios found exceeds the amount of the original study by far as the number of participants was much larger. For the Wiki-System, about 6 percent more unique scenarios were found than for the LiveNet-System. The spread along the various classes is quite different too. Concerning the LiveNet-System, the data set of 18|48|34 is very similar to the original study of 20|40|40. But the Wiki-system shows some kind of “dent” turning the classification more into a 20|35|45 distribution. According to the conclusion of the original study and the conclusion of this replication, the results are better in this case. Under industrial circumstances mostly class A scenarios would be selected for any system adoption. However, in many cases, time and money will most likely not be sufficient for solving class C scenarios.

7.3.3. Scenario rating based on experts scoring

Expert rating requires knowledge in terms of the domain and the impact on business values in order to find the most important scenarios. Unlike the original study, two members of the experiment team conducted the classification process. Based on an unsorted scenario

list, separate for both systems, they simply ranked the scenarios in accordance with their opinions.

7.4. Individual setting

Overall, 59 participants took part in the study. All of them were students of the *University of Technology of Vienna*. They were recruited from two different courses. 19 students attended the course *Software Testing*, which is a master course and therefore these participants were master students. The remaining 36 students took part in an undergraduate course in *Software Quality Assurance*. For the quality assurance course, the study was actually part of the program, whereas the master students could optionally take part in the study for extra points for their course. It was not possible to motivate all the students to take part in the study.

7.4.1. Study participants and study groups

Altogether 59 participants were recruited from a software testing course (Master students) and from a software quality assurance course (Bachelor students). Unfortunately, only 54 data sheets (software testing 19, software quality assurance 35) were of good enough quality to be considered. The remaining examples were either incomplete or of poor quality. Participants leaving before finishing their tasks happened too. However, in the end only 47 participants handed in all given data sheets and questionnaires. Nevertheless the missing data sheets only contained the feedback ones. So despite of this matter still 54 participant data sets have been usable for evaluation.

Treatments / Material	Treatment group (change categories provided)	Control group (change categories not provided)
System Livenet	27	27
System Wiki	27	25

Figure 7.8.: Sizes of the two study groups

Above table shows the number of participants of each group. In the first run, the treatment group was consisted of 27 participants and the control group of 27 individuals. Note, that in the second run, the participants switched their groups. So, the 27 individuals went to the control group whereby one student had got lost (possibly left). Accordingly, the 27 students of the control group also switched, whereby two students got lost here too.

7.4.2. Individual experience

Rating the experience of the individuals is based on the experience questionnaire handed to them in the run-up of the study. Unlike in the original study, it was decided to ascertain the experience of the students not only by their years as students or as professionals but to consider more aspects of the retrieved data set. Concretely, all the questions except the first four are used for calculation.

Since all the individuals were students, age was not a significant factor and did not provide any meaningful information about their experience. Neither is their ability to understand English as a foreign language an indicator of their knowledge of computer science. Nevertheless, weak language skills might cause significant performance decreases. Therefore, that topic is discussed separately in this work.

Surely, the amount of attended programming courses is a criterion and increases experience. Unfortunately, the received answers were largely disparate. A certain amount of students did not give any feedback at all and others stated a large number of attended programming lessons which are very unlikely for a normal computer science student. The problem is probably estimated with the definition of programming course. For some, such a course is strictly about programming and others take courses into account like dealing with UML or something else. Anyhow, it was decided to disregard this question because of the data set given.

The fourth and last question not considered is the one about a self-assessment of the individual about his or her software engineering experience. As these answers tended to be very subjective and are hard to prove, they cannot be taken into account.

Ordinal data

Except for two questions (E_FB09 and E_FB10), all the ones considered deliver ordinal data. They were collected and summed up for each student.

Nominal data

The two questions left are concerned with the experience in terms of working as a professional. More in detail, the individual had to state how long he or she had worked within real software projects and how large the biggest project was in which he or she had ever participated. Then the students were rated as experienced or less experienced by giving them a score of 4 or only 1 point. The cut-off point for this classification was in both cases a score of 5. Above that we classified the individual as experienced and below as less experienced. The mean of both scores was calculated and used.

Calculation of the Experience

Now, the sum of all ordinal data questions was divided by the mean experience drawn from the questions about professional work. Unfortunately, the results produced a data cloud which did not provide any hint for a meaningful categorization of the individuals in terms of experience.

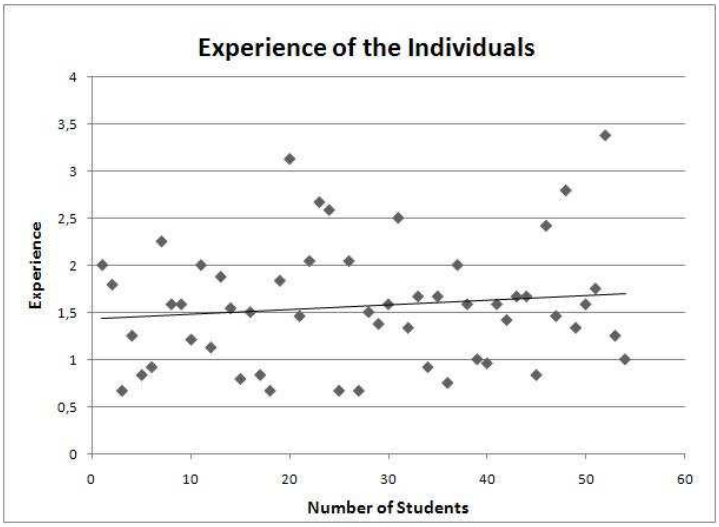


Figure 7.9.: Calculated experience of the individuals

Therefore, a classification of the experience was introduced. To be more specific, the classification was extended to three variables, well experienced (A), experienced (B) and less experienced (C), similar to the classes of the scenarios. The original study used only two classes, individuals with more than 3 years of IT experience and individuals with less than 3 years of IT experience. Taking a normal distribution as a point of reference, the ranges of the variables are:

Experience Class	Range of Experience Score	# of Individuals per Class
A	> 2,24	8
B	< 2,25	35
C	< 0,96	11

Table 7.3.: Range of experience classes

Figure *7.8* shows that the experience of the individuals is broadly based. A situation where half of the participants are well experienced and another one is less-experienced was not the case. Additionally, it was investigated whether the experience scores of the individuals were evenly distributed. As (nearly) all participants worked with both systems and were once part of the treatment group and the control group, there should be no difference.

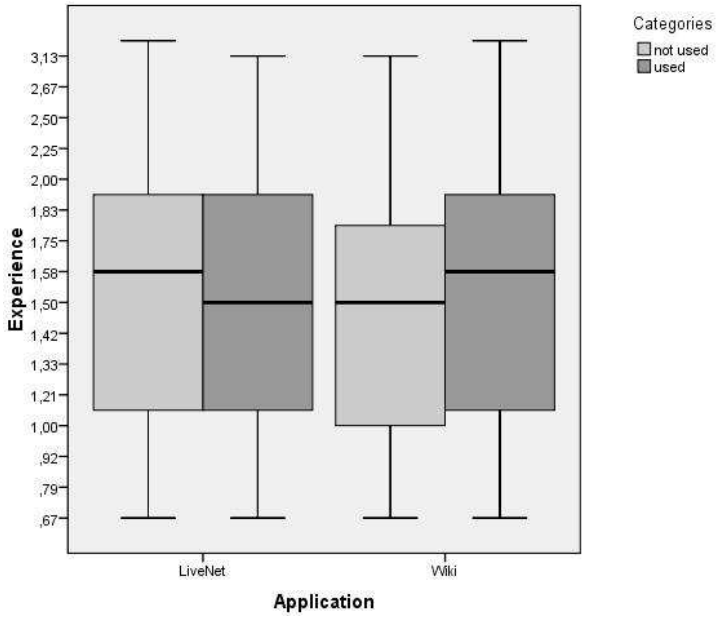


Figure 7.10.: Average experience score of the participants per system

As can be seen from the box plot, the expected results appeared. The individuals of both groups received almost the same experience score regardless of the system used. The mean for the control group is 1,54 and for the treatment group 1,56. The same findings can be seen in the medians which are 1,58 for the control group and 1,50 for the treatment group. The standard deviation within the systems are negligibly small (LiveNet: control group 0,65 - treatment group 0,64; Wiki: control group 0,64 - treatment group 0,65). Furthermore the difference between the means of the two systems is too tiny to be of any significance.

	Control Group - LiveNet		Control Group - Wiki		Control Group - Total	
Experience	No	Share	No	Share	No	Share
Class A	3	11%	4	16%	7	13%
Class B	18	67%	16	64%	34	65%
Class C	6	22%	5	20%	11	21%
Total	27	100%	25	100%	52	100%

	Treatment Group - LiveNet		Treatment Group - Wiki		Treatment Group - Total	
Experience	No	Share	No	Share	No	Share
Class A	5	19%	3	11%	8	15%
Class B	17	63%	18	67%	35	65%
Class C	5	19%	6	22%	11	20%
Total	27	100%	27	100%	54	100%

Table 7.4.: Distribution of the experience of the groups per system

The p-value of 0,979 indicates that there is no difference between the systems. The box plot *7.9* shows that one study group was diminished in terms of experience. The reason is that, as pointed out under 7.3.1 “Study grouping”, two students left the experiment between the first and the second run. To summarize, table *7.3* shows a table of both groups but separate

for each system. Again, the overall distribution is almost the same for both system and for both groups. Following table presents the distribution of the experience classes for both used software systems.

7.4.3. Language skills

The complete study replication was executed in English. Examples are the introduction lecture, all materials, the systems, the tasks, the questionnaires and other parts used within the study procedure. As German is the by far most-widely spoken language in Austria, English skills are expected to be lower compared to their German *and* to the English skills of the participants of the original study.

In the experience questionnaire, the individuals were also asked about their overall English skills. Of course the answers to this question were based on self-estimation. As nearly no one likes to rate himself lower than he actually is, there is a certain probability that the scores are higher than the "realistic" actual level. The possible answer set was ranked from zero (no experience) to five (excellent skills).

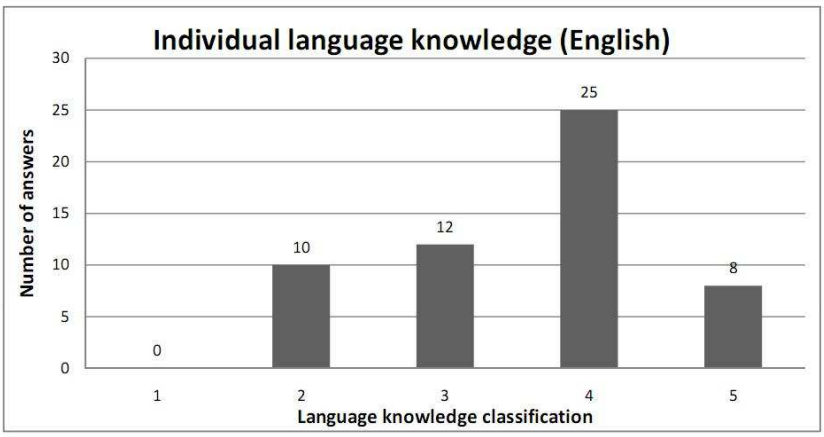


Figure 7.11.: Distribution of English language skills

As expected, no individual marked zero. Using the mean value of 2,54 at a standard deviation of 0,99 the predicted average skill level is rather good. But about 20 percent of the people questioned stated that they only had minor language experience. Taking into account that quite a number of special expressions and vocabulary were used in the whole study replication there exist major disadvantages for those individuals.

7.5. Team setting

The created teams consisted of three team members each. Conform to the team roles defined for the "Fagan inspection" each of the team members had to take on a certain role.[5] Every role implied several responsibilities. In addition, to fit a role, the person was supposed to have some specific characteristics. Logically, with only three members randomly assigned to a specific team, there was not much choice. Nevertheless, the participants were encouraged to keep to their roles and to treat them with care.

Role	Responsibilities	Desirable Characteristics/Skills
Moderator	Facilitates the generating of scenarios; Responsible to keep team focus on the task; Ensures that team follow the guidelines/process	Good facilitation skills; Careful observation skills – able to intervene when discussions stuck; social competences required
Scenario Writer	Responsible for actual scenario documentation; Captures exact agreed wording; Moving on to next scenario not allowed until exact agreed wording is written down	Pedantic, exact; good typing skills required
Timekeeper	Helps the team managing the allocated time. Is warning if the discussion on one single scenario goes on too long (normally 3 minutes).	Pedantic; good sense of time; dutiful

Table 7.5.: Team roles of a software evaluation team

7.6. Nominal teams

In order to classify the team performances, they have to be compared to each other and/or to some kind of index. The latter is useful, as it provides a less subjective measurement. Another possibility is the comparison with the nominal teams in addition to the comparison with the results of the original study

A nominal team is a fictitious team but consists of real individuals. For comparison with real teams there are again three members to a team. In order to test the effectiveness of several team sizes, nominal teams were build up to seven team members. The difference to the real teams is that the team members never met in reality. To achieve that goal a simple computer script was written (in Java) operating like a random generator. The input consisted in a list of the individuals together with their individual scenario lists. From that list, all the possible combinations of teams with sizes of three, four... seven and eight were created. It was important to avoid redundancy. The nominal team with the individuals 9-12-27 for example was not to exist again as 12-27-9.

Of course, random means that the actual real team comparisons exist as nominal team too. But the scenario list of the nominal team was not the same since the nominal team scenario lists were taken from the individually found ones of each member. Every unique scenario found by the team members are collected and put in the team scenario list. Again, all redundancies were eliminated. So the nominal teams represent somehow the optimal solution for a team

with regard to all their created scenarios. Nominal teams with more than three team members were used to measure possible gains or losses in efficiency within team meetings of larger team sizes.

Unfortunately, the calculations of nominal teams faced some restrictions. A java program was written to match all possible team combinations. Further, all scenarios found by the individuals of a specific team are put into a specific team list for that team. Due to IT infrastructure limitations, only teams up to 7 team members where calculated. Calculating 6 team members produced results of around 16.000.000 combinations and text files of nearly one gigabyte in size. This resulted in Database problems. The limit for a database is two gigabyte of the file size which is nearly exceeded with this amount of data. Therefore nominal teams with 6 and 7 team members were calculated by sampling.

7.7. Validity

Every replication has to deal with at least similar validity problems like their predecessor study/studies. This is especially true if the replication method is an exact replication.

7.7.1. Internal validity

Internal validity or validity of (causal) interferences in scientific studies is mostly threatened by causal relations between variables. This can lead to following problems:[69]

1. The “cause” precedes the “effect” in time (temporal precedence)

2. The “cause” and the “effect” are related (covariation)

3. There are no plausible alternative explanations for the observed covariation

However, to counter bias problems when attributing the values of dependent variables to the experimental variables, the assignment of the participants to the various groups [LiveNet (treatment and control) and Wiki (treatment and control)] was done randomly. In the run-up, all the participants were put on a list and then randomly selected for groups before the experiment started. The participants did not have any information on their group affiliation and had no influence on it afterwards.

Unfortunately the threat of learning effects probably existed. As described in chapter 7.1 “Experiment design” of this work the individuals working in the first run with the scenario categories would have been able to remember them in the second run. As the groups had to

switch they had to simulate persons who have not been given these scenario categories too. It was intensively debated whether to change the study setup to avoid this problem, or not. However, the replicators favored changes but the researchers of the original study asserted to keep the design. The probable learning effects have been considered less problematic.

The scoring system respectively the appropriateness of the approach to classify the scenarios based on their frequency represents another threat to the internal validity. However, this approach has been used in the original study as well as in several other empirical studies.[44, 19] The various threats regarding this procedure have been discussed and addressed during that studies.

Along with this approach another problem exits: The skill, bias and knowledge of the person in charge recording the scenarios, eliminating duplicates and joining scenarios which are semantically the same before counting their frequency can be a threat to the internal validity. Therefore, three researchers performed these tasks independently. Any disagreements were resolved before the scenarios were counted.

The second system used a categorization developed by two experts. All of them were familiar with the two systems (LiveNet and Wiki) as well as with the software architecture, software requirements and scenario finding. However in this context these threats have only little influence on the overall results of the experiments.

7.7.2. External validity

External validity defines to which degree the results of scientific research work can be generalized, or compared with respectively transferred with similar situations. In particular the question is: Are the participants representative enough for stakeholders performing architecture evaluation within an industrial context and are the experimental materials and processes representative enough for materials and processes used within industrial software architecture evaluations? Compared with the original study, the participants do not completely match the industrial stakeholder's background (e.g. marketing, sales, controlling, computer science, and so on).

All the students have a variety of backgrounds (educational and professional) in computer science and software engineering. But the differences of experiences may not be large enough for a significant impact. Therefore, it is more appropriate to generalize the results of stakeholders with a technical background. The same restrictions existed using the LiveNet and the Wiki system. In industry, the stakeholders would have more knowledge and experience with the systems than the people in this study. The participants had only limited knowledge, especially with the LiveNet-System. Hence, they can more likely be compared with stakeholders who have also limited knowledge about the current system that is being evaluated.

But there is evidence in IT-science that IT-students can replace professionals, working in IT-industry, within empirical studies.[70] The participating individuals within this study replication were all computer science students and attended several computer science lectures. Further most of them had some experience working in the IT-industry and therefore had the necessary technical skills and knowledge.

Another problem area arises within the methods of software architecture evaluation and scenario finding. The participants did not have any or only little experience with those. According to [17, p.5], companies nowadays do not provide extensive training to their employees considering software architecture evaluation or methods to find quality-sensitive scenarios. Thus, the experience of the participants in this study can be compared with the experience of stakeholders involved in software evaluation processes in industry.

The software requirement specifications used in this replication (and in the original study) are relatively short and simple. An industrial sample would be much more detailed and complex. But the stakeholders would also have much more time for the software evaluation and the scenario finding. Finally, there is a threat to the external validity if the scenario development process in this replication is not representative with industrial practices of scenario-based software architecture evaluation. However, the scenario development process in this replication was comparable to the one used for most of the scenario-based architecture evaluation methods creating scenarios for quality requirements characterization. These requirements are to be fulfilled by proposed software architecture through brainstorming workshops like QAW. [17, p. 5]

8. Findings of the replication study

For the statistical analyses, common and proven methods of descriptive statistics were used: Mean, standard deviation and median. Diagrams and box plots represent findings in visualized form. The results are tested using p-tests with a 95% confidence interval whereas the null hypothesis always states that there is no significant difference between the samples. To provide a good clarity, the order of the research questions discussed in section 6.1 “Research hypotheses” is kept the same for the presentation of the results.

8.1. Do change scenario categories help to find more or better scenarios?

This research question contains individual results, team results and nominal team results. All of them are viewed separate and compared to each other. Additionally findings are discussed showing scenario losses and scenario gains of real teams compared to nominal teams.

8.1.1. Scenarios found by individuals

As mentioned in the chapter 7.1 “Experiment design”, only 47 participants handed in the complete data set. In most cases, the participants left the study between the individual brainstorming session and the team meetings. Hence, nearly all of the other participants carried out the individual brainstorming. Due to the fact that only minor (change) scenarios were developed during the team meetings, it is acceptable to assume that each of the 55 participants was average in his performance. The various groups differed only slightly in size. For the LiveNet-System, the control group consisted of 27 participants whereas the treatment group comprised 27 students. Within the Wiki-System, 25 people made up the control group and 27 the treatment group.

The average number of scenarios found per student is around 6-7, which is not really that close to the 9 scenarios found per individual in the original study. On average the individuals found fewer scenarios within this study replication. Regarding the overall performance, the

individuals working with the LiveNet-System found around 20 % less scenarios compared to the ones using the Wiki-System.

Scenarios found by individuals

System	Categories	Mean	N	Std. Deviation	Median
LiveNet	not used	6,04	27	3,611	6,00
	used	6,19	27	2,869	6,00
	Total	6,11	54	3,231	6,00
Wiki	not used	7,64	25	3,277	7,00
	used	7,07	27	2,541	6,00
	Total	7,35	52	2,903	7,00
Total	not used	6,81	52	3,515	7,00
	used	6,63	54	2,722	6,00
	Total	6,72	106	3,122	6,00

Table 8.1.: Average scenarios found per individual

Extraordinarily, the individuals performed that much better with the Wiki-System, regardless of whether the treatment groups or the control groups are compared with each other.

Instead, between the treatment and the control group, no statistically significant difference could be found. A t-test on a 95 percent confidence interval scored a p-value of 0,287, which is above 0,05 and therefore affirms the null hypothesis that providing change categories does not produce or produces hardly differences in results.

So the differences in the data set may have been arrived at by chance. However, working either with the Wiki-System or the LiveNet-System has no effect on the results as indicated by a p-value of 1,000. Thus, an individual working on the Wiki-System does not find significantly more or fewer scenarios.

Without further parameters, no good explanation can be given for these findings. Unfortunately, it seems that providing (change)scenario categories in advance does not help finding more scenarios on an individual level. But this does not imply that the support is meaningless in general.

Perhaps the treatment group found more scenarios of higher quality which make them more useful. Or vice-versa, the supported individuals found fewer unimportant ones. Following are more diversified calculations covering these questions.

Number of scenarios found by individuals

System	Categories	Class	Mean	N	Std. Deviation	Median
LiveNet	not used	A	3,30	27	2,035	4,00
		B	1,74	27	1,457	2,00
		C	1,00	27	1,209	1,00
		Total	2,01	81	1,854	2,00
	used	A	3,30	27	1,660	3,00
		B	1,70	27	1,295	2,00
		C	1,19	27	1,711	1,00
		Total	2,06	81	1,791	2,00
	Total	A	3,30	54	1,839	3,00
		B	1,72	54	1,366	2,00
		C	1,09	54	1,470	1,00
		Total	2,04	162	1,817	2,00
Wiki	not used	A	4,80	25	2,363	5,00
		B	1,36	25	,952	1,00
		C	1,48	25	1,503	1,00
		Total	2,55	75	2,327	2,00
	used	A	4,04	27	2,009	3,00
		B	1,22	27	1,783	1,00
		C	1,81	27	1,111	2,00
		Total	2,36	81	2,057	2,00
	Total	A	4,40	52	2,199	4,00
		B	1,29	52	1,433	1,00
		C	1,65	52	1,312	1,50
		Total	2,45	156	2,186	2,00
Total	not used	A	4,02	52	2,305	4,00
		B	1,56	52	1,243	1,00
		C	1,23	52	1,366	1,00
		Total	2,27	156	2,105	2,00
	used	A	3,67	54	1,863	3,00
		B	1,46	54	1,563	1,00
		C	1,50	54	1,463	1,00
		Total	2,21	162	1,929	2,00
	Total	A	3,84	106	2,089	4,00
		B	1,51	106	1,409	1,00
		C	1,37	106	1,416	1,00
		Total	2,24	318	2,014	2,00

Table 8.2.: Average scenario per class found by individuals

Now, pertaining to scenario classification as a further differentiation, the students working with the Wiki-System performed better concerning especially critical scenarios. The overall total amount of scenarios found per individual serves as the used index. Let us focus on the LiveNet-System first. Looking at table *8.2* more in specific, the treatment group found a little bit more scenarios than the control group regarding important (class B) scenarios. However, both found almost an equivalent number.

Considering critical and less important scenarios, again the treatment group performed very similar. Overall, class A scenarios were found most often in both groups in spite of the fact that there are small differences in the performance. The situation is distinct regarding the Wiki-

System. The control group performed significantly better regarding class A and B scenarios. Only in the case of less-important scenarios the treatment group was able to find slightly more scenarios. Further, the difference between the two groups regarding critical scenarios within the Wiki-System is rather large. Similarly to the LiveNet-System, most of all class A scenarios were found.

Overall, the control group found slightly more class A and class B scenarios as well as marginally fewer class C scenarios. A t-test with a 95 percent confidence interval investigating the mean values of the two systems resulted in a p-value of 0,066. This indicates that there is no or only a slight statistical difference between the means.

Notable is the fact that within a 10 % interval the null hypothesis would have been rejected. Hence, differences observed between the systems are probably minor due to chance. But nevertheless, it does not make any difference for an individual performance whether he or she works with the LiveNet- or the Wiki-System.

Now, the same test considering the control and the treatment group results in a p-value of 0,320. This means that there is no significant statistical difference between them. Providing change categories as a support did not help the individuals to find more (or better) scenarios. However, the statistical evaluation test did not produce a stable finding.

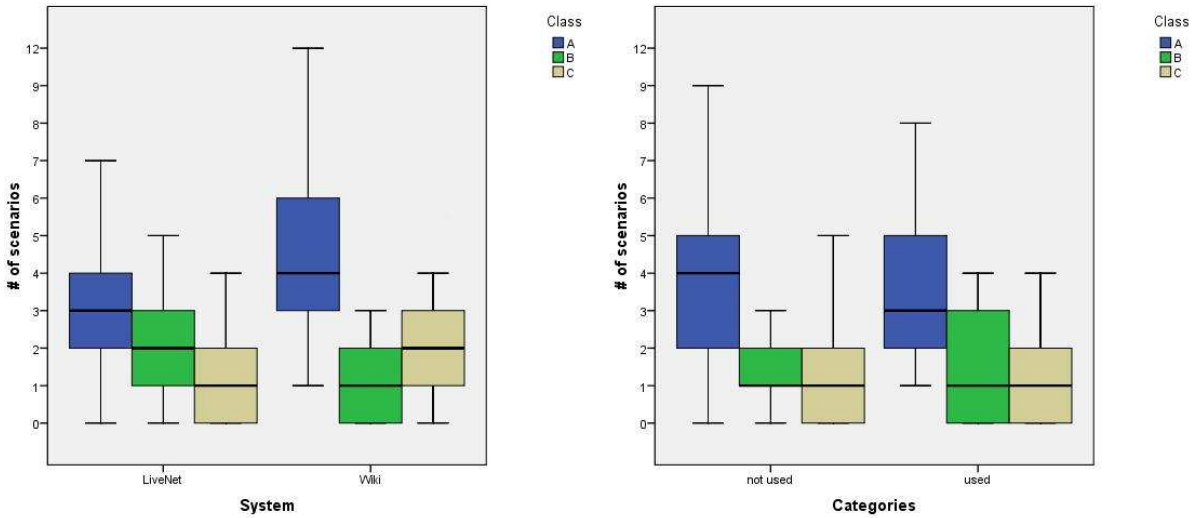


Figure 8.1.: Comparison of the individual groups

The box plot above visualizes the differences between the two systems and the two groups. With regard to the systems, the individuals working with the Wiki-System performed better concerning critical scenarios and slightly better in terms of less-important ones. Only for class B scenarios, the individuals working with the LiveNet-System found a few more scenarios. Comparing the two study groups the differences in the performances are smaller.

Interestingly, the range of scores for critical scenarios stays the same; just the median indicates the better performance of the control group. The boxes for class B and class C scenarios show similar results. The performance of both groups is more or less the same.

Number of scenarios found by individuals (LiveNet)

System	Categories	Class	Mean	N	Std. Deviation	Median
LiveNet	not used	A	3,30	27	2,035	4,00
		B	1,74	27	1,457	2,00
		C	1,00	27	1,209	1,00
		Total	2,01	81	1,854	2,00
	used	A	3,30	27	1,660	3,00
		B	1,70	27	1,295	2,00
		C	1,19	27	1,711	1,00
		Total	2,06	81	1,791	2,00
	Total	A	3,30	54	1,839	3,00
		B	1,72	54	1,366	2,00
		C	1,09	54	1,470	1,00
		Total	2,04	162	1,817	2,00
Total	not used	A	3,30	27	2,035	4,00
		B	1,74	27	1,457	2,00
		C	1,00	27	1,209	1,00
		Total	2,01	81	1,854	2,00
	used	A	3,30	27	1,660	3,00
		B	1,70	27	1,295	2,00
		C	1,19	27	1,711	1,00
		Total	2,06	81	1,791	2,00
	Total	A	3,30	54	1,839	3,00
		B	1,72	54	1,366	2,00
		C	1,09	54	1,470	1,00
		Total	2,04	162	1,817	2,00

Table 8.3.: Average scenario per class found by individuals (LiveNet)

In order to get more detailed answers, both systems are reviewed on their own. If the LiveNet-System is looked at by itself, a test of the means of the two study-groups results in a p-value of 0,633 whereas the same procedure for the Wiki-System produces a p-value of 0,397.

The following box plots (figure *8.2*) present an overview of those more specific data sets. Regarding the LiveNet-System the control group performed a little bit better in the case of critical scenarios. But all in all, both groups produced nearly equal results. Looking at the Wiki-System presents quite a similar picture. Again, the control group found more class A scenarios. Compared to the box plot above, the general statement there is confirmed.

Number of scenarios found by individuals (Wiki)

System	Categories	Class	Mean	N	Std. Deviation	Median
Wiki	not used	A	4,80	25	2,363	5,00
		B	1,36	25	,952	1,00
		C	1,48	25	1,503	1,00
		Total	2,55	75	2,327	2,00
	used	A	4,04	27	2,009	3,00
		B	1,22	27	1,783	1,00
		C	1,81	27	1,111	2,00
		Total	2,36	81	2,057	2,00
	Total	A	4,40	52	2,199	4,00
		B	1,29	52	1,433	1,00
		C	1,65	52	1,312	1,50
		Total	2,45	156	2,186	2,00
Total	not used	A	4,80	25	2,363	5,00
		B	1,36	25	,952	1,00
		C	1,48	25	1,503	1,00
		Total	2,55	75	2,327	2,00
	used	A	4,04	27	2,009	3,00
		B	1,22	27	1,783	1,00
		C	1,81	27	1,111	2,00
		Total	2,36	81	2,057	2,00
	Total	A	4,40	52	2,199	4,00
		B	1,29	52	1,433	1,00
		C	1,65	52	1,312	1,50
		Total	2,45	156	2,186	2,00

Table 8.4.: Average scenario per class found by individuals (Wiki)

In brief, no evidence has been found that scenario change categories helped the individuals to find more scenarios. Neither did they help to find scenarios of higher quality. Unfortunately, these findings do not confirm the results of the original study.

Comparison to the original study

The individuals found fewer scenarios this time. This is true for both systems and for both study groups. Then the treatment group found 10 and the control group 9 scenarios. Here the participants developed only 6 to 7 scenarios. Additionally the influence of scenario categories measured in the original study cannot be found.

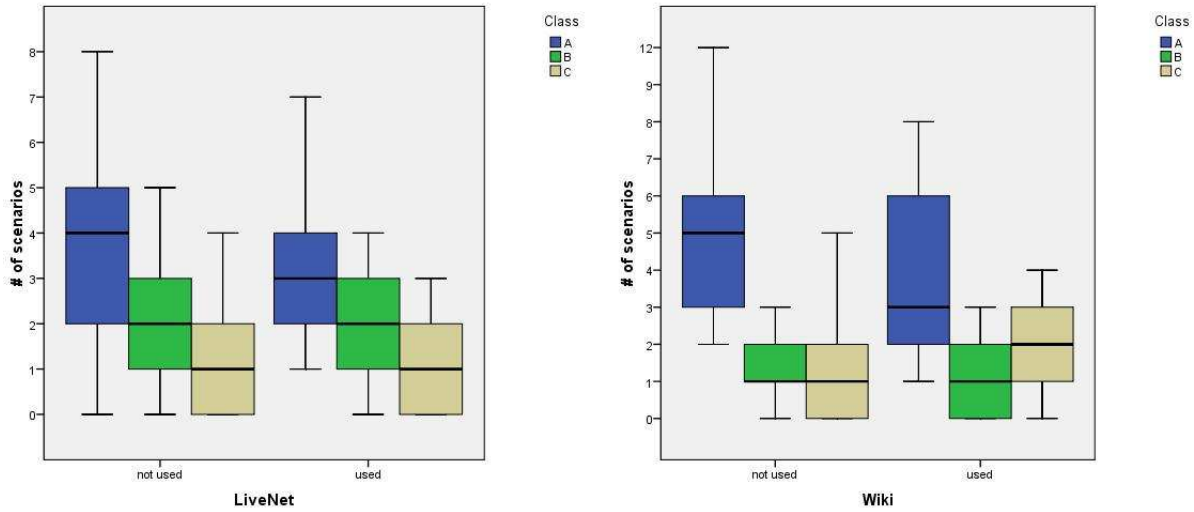


Figure 8.2.: Comparison of the individual groups specifically for each system

8.1.2. Scenarios found per real team

Analyzing both groups (control, treatment) and both systems (LiveNet, Wiki) combined, every team found around 10 scenarios on average. Comparing the overall individual performance with the the overall team one, the teams developed 30 to 40 % more scenarios. It is important to remember that the individual brainstorming session lasted only 45 minutes whereas the team session took 60 minutes.

Remarkable is the fact that the difference between the two systems is smaller compared with those of the individuals. There, about 20 % more scenarios were developed for the Wiki-System than for the LiveNet. At the team level, however, this gap shrank to half of this amount, which is to 10 %. To answer the question why is quite difficult. At this point no sufficient answer can be given using the available data.

Probably, the possibility to discuss problems within a team helps to overcome insecurities generated by a foreign environment, in this case represented by the LiveNet-System. Due to the spread of “Wikipedia” it is most likely that the Wiki-System is quite familiar to the individuals, more so than the LiveNet-System. The latter was created for university research purposes and is virtually unknown beyond that realm.

According to a p-value of 0,116, working with different systems seems to have no influence on the performance. The same assumption can be made for the treatment and the control group. A t-test resulted in a p-value of 0,174.

Both indicate, like for the individuals, that neither the allocation to a certain system nor to a specific replication group affected the performance of the teams. But the values are not really very high (either), which would make them more convincing for supporting predictions more strongly.

Scenarios found by teams

System	Categories	Mean	N	Std. Deviation	Median
LiveNet	not used	9,00	9	3,000	9,00
	used	9,90	10	2,514	10,50
	Total	9,47	19	2,716	10,00
Wiki	not used	10,89	9	3,855	11,00
	used	10,00	9	3,162	9,00
	Total	10,44	18	3,451	10,00
Total	not used	9,94	18	3,489	9,00
	used	9,95	19	2,758	10,00
	Total	9,95	37	3,091	10,00

Table 8.5.: Average scenarios found per team

Referring to the original study, the difference which was calculated there between the teams and the individuals was 90 %. Now within this study, the teams performed still better. But the difference is much smaller. There are no reasonable facts or data sets which could explain this big gap. Possibly, the students in the original study knew each other well as there were only 24 participants.

Therefore, they may not have had to introduce them to such a setting. Looking at the current study again it is conspicuous that the teams working with the Wiki-System achieved better results. But it cannot be stated that the treatment or the control group did make a better job.

Taking more variables into account presents a more detailed picture. More specifically, the amount of scenarios found per team can be split up into sub-variables. An average (real) team found 3 (change) scenarios per class. In general, the treatment groups performed not even 1 % better, which is quite negligible.

Concerning only on class A scenarios, real teams developed more than 6 scenarios on average. But the gap between the study groups is again only around 1 %. The treatment groups even found slightly fewer class B scenarios.

Viewing the systems separately, the big picture remains more or less the same. The control groups working on the Wiki-System found most critical and more important scenarios than the treatment groups. However, the treatment groups using the LiveNet-System found, by contrast, more class A and B scenarios. Regarded combined, both groups produced the same results.

Scenarios found by teams

Application	Categories	Class	Mean	N	Std. Deviation	Median
LiveNet	not used	A	5,78	9	1,922	5,00
		B	3,22	9	1,641	3,00
		C	,00	9	,000	,00
		Total	3,00	27	2,787	3,00
	used	A	6,00	10	2,160	5,50
		B	3,90	10	1,595	4,50
		C	,00	10	,000	,00
		Total	3,30	30	2,938	3,50
	Total	A	5,89	19	1,997	5,00
		B	3,58	19	1,610	4,00
		C	,00	19	,000	,00
		Total	3,16	57	2,846	3,00
Wiki	not used	A	7,00	9	2,179	6,00
		B	3,89	9	2,522	3,00
		C	,00	9	,000	,00
		Total	3,63	27	3,455	3,00
	used	A	6,89	9	2,147	6,00
		B	3,11	9	1,965	3,00
		C	,00	9	,000	,00
		Total	3,33	27	3,293	3,00
	Total	A	6,94	18	2,100	6,00
		B	3,50	18	2,229	3,00
		C	,00	18	,000	,00
		Total	3,48	54	3,346	3,00
Total	not used	A	6,39	18	2,090	6,00
		B	3,56	18	2,093	3,00
		C	,00	18	,000	,00
		Total	3,31	54	3,125	3,00
	used	A	6,42	19	2,143	6,00
		B	3,53	19	1,775	3,00
		C	,00	19	,000	,00
		Total	3,32	57	3,083	3,00
	Total	A	6,41	37	2,088	6,00
		B	3,54	37	1,909	3,00
		C	,00	37	,000	,00
		Total	3,32	111	3,090	3,00

Table 8.6.: Average scenario per class found by team

Intended to validate the findings, t-tests were applied focusing on the systems and the study groups. For the systems the resulting p-value of 0,156 indicates that the system has no effect on the team performance. Much more sound and clear is the p-value of 0,972, testing the group scores. Hence, support in the form of scenario categories does not have any impact on the performance of the team. As mentioned, the same is true for the systems but may not be that strong.

However, all the teams together did not have any class C scenarios on their result list. It seems that teams are better than individuals in terms of the quantity and even better in terms of the quality of found (change) scenarios! The box plots (figures *8.3* and *8.4*) visualize the above mentioned findings. It can be seen that the results are quite closely-set.

When both systems were separately analyzed, the following p-values comparing the means were calculated for each respective control group/treatment group. For the LiveNet-System the result was 0,479 and for the Wiki-System 0,549. These figures somewhat weaken the results of the above mentioned values. Any possible impact of the system seems to be lower and the one of the group allocation higher. But still they are within similar range, which leads

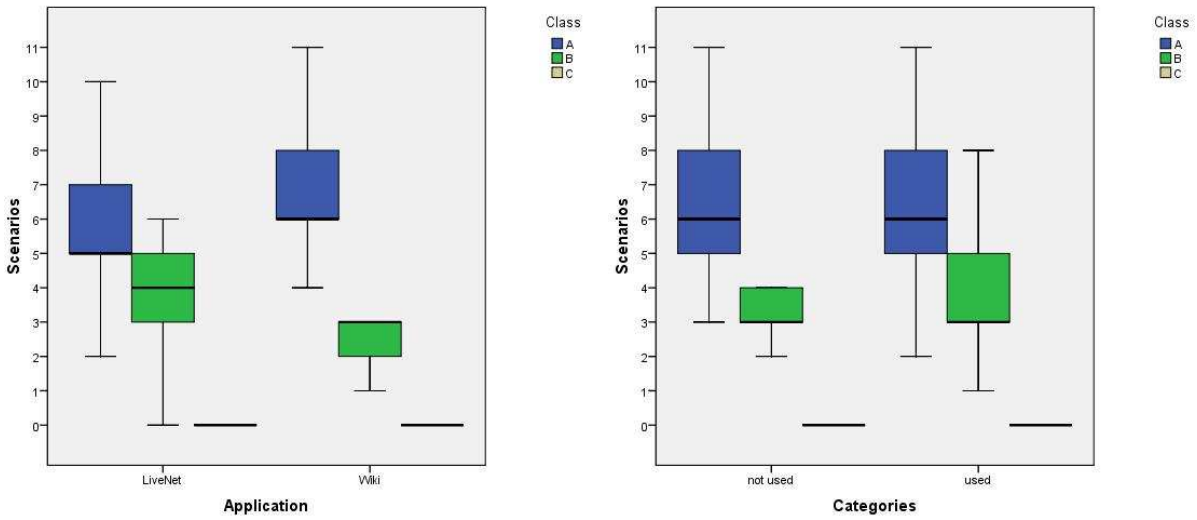


Figure 8.3.: Comparison of the real teams

to the same conclusions. Neither the systems nor the scenario categories influenced the teams respectively their results.

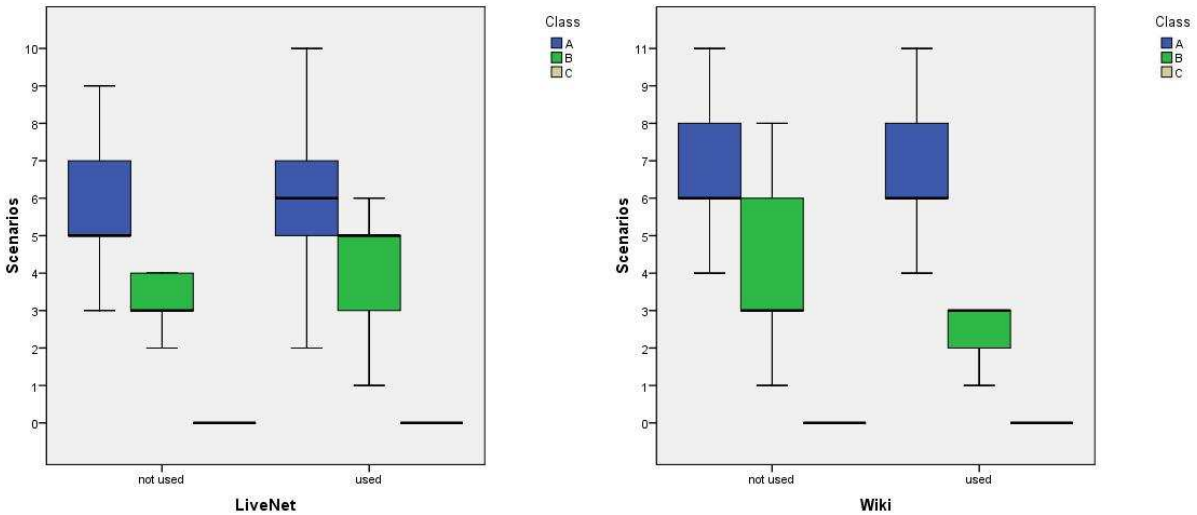


Figure 8.4.: Comparison of the real teams specifically for each system

More specific box plots, figure *8.4*, provide an overview of the data sets. Surely, differences can be identified but in the end the tendency towards the figure *8.3* is generally observed again.

Comparison to the original study

Interpreting the results for the real teams, they recall the statements made for the individuals. No evidence was found that the support of (change) scenarios has an influence (neither positive nor negative) on the results of the team session. Also the different systems did not affect the performance either. This is contrarily to the original study which summarized that, for real teams, scenario categories do help teams to perform better. The teams at the original study found on average 15 scenarios whereas during the study replication the average was 10 scenarios.

8.1.3. Scenarios found per nominal team

Notice that these results are abstract. As described in section 7.6 “Nominal teams”, the members of this teams never met each other. Logically, as all possible combinations were matched, the real teams also appear. But also this time, the nominal teams were randomly picked. Yet, several factors were involved which definitively have to be considered. Real individuals need some time to introduce themselves to each other. They need some time to feel comfortable, to deal with each other.

Furthermore, if the real teams work tool supported, they even have to get used to the system first. In addition, real team members discuss all their results and argue pro or contra scenarios before they probably put one on their list. Nominal teams, however, do not simulate human behavior. Instead, the drawing up scenario lists are “ideal”. So the comparison of real and nominal team performance is more a comparison of real results against optimal theoretical results.

Generally, a nominal team found some 18 (change) scenarios. Accordingly, this is an increase of approximately 80 % over the real teams with around 10 scenarios per team. Focusing on critical scenarios (class A), nominal teams found on average 9,5 scenarios whereas real teams only found 6,4. This time the difference was about 50 %.

For class B scenarios, the compared samples differed not that strong. The nominal teams found about 25 % more scenarios than the real teams. For less important scenarios, the nominal teams found on average about 4 scenarios which is not comparable as real teams did not find any class C scenarios at all.

Scenarios found by nominal teams

System	Categories	Class	Mean	N	Std. Deviation	Median
LiveNet	not used	A	8,44	2925	2,517	8,00
		B	5,15	2925	2,320	5,00
		C	3,00	2925	1,975	3,00
		Total	5,53	8775	3,195	5,00
	used	A	8,05	2925	2,254	8,00
		B	5,07	2925	2,091	5,00
		C	3,56	2925	2,794	3,00
		Total	5,56	8775	3,041	5,00
	Total	A	8,25	5850	2,397	8,00
		B	5,11	5850	2,208	5,00
		C	3,28	5850	2,435	3,00
		Total	5,54	17550	3,119	5,00
Wiki	not used	A	11,76	2300	2,818	11,00
		B	3,94	2300	1,478	4,00
		C	4,44	2300	2,443	4,00
		Total	6,72	6900	4,260	5,00
	used	A	10,06	2925	2,593	10,00
		B	3,61	2925	2,860	3,00
		C	5,44	2925	1,814	5,00
		Total	6,37	8775	3,664	6,00
	Total	A	10,81	5225	2,824	11,00
		B	3,75	5225	2,360	3,00
		C	5,00	5225	2,172	5,00
		Total	6,52	15675	3,941	6,00
Total	not used	A	9,90	5225	3,125	10,00
		B	4,62	5225	2,081	4,00
		C	3,63	5225	2,307	3,00
		Total	6,05	15675	3,748	5,00
	used	A	9,06	5850	2,628	9,00
		B	4,34	5850	2,609	4,00
		C	4,50	5850	2,538	4,00
		Total	5,96	17550	3,391	6,00
	Total	A	9,46	11075	2,904	9,00
		B	4,47	11075	2,379	4,00
		C	4,09	11075	2,469	4,00
		Total	6,01	33225	3,564	5,00

Table 8.7.: Average scenario per class found by nominal teams

Comparing the nominal teams among each other, the overall trend continued that the Wiki-System gained more scenarios. Especially for critical scenarios (A), a lot scenarios more were developed than for the LiveNet-System. However, reviewing class B scenarios, the situation is vice-versa. Within the LiveNet-System, the control group performed slightly better in terms of critical scenarios as well as in terms of class B scenarios. The findings are about the same with the Wiki-System.

A 2-tailed t-test considering the two groups resulted in a p-value of 0,028. Hence, there is some statistical difference between them. Concentrating on the LiveNet-System the same test scored a p-value of 0,503. On the other hand, testing the study groups the Wiki-System, the score is < 0,001. According to these figures providing (change) scenario categories had an impact on the individuals using the LiveNet-System and had no impact on the individuals working with the Wiki-System. Taking both systems into account the data states that giving (change) scenario categories to (nominal) teams has a statistical influence.

On reviewing the figures *8.5* and *8.6*, [] the following conclusions can be drawn: The control group (not supported by change scenario categories) and the treatment group found nearly

equally as many scenarios whereas there are slight indications that the control group developed a few more scenarios in terms of critical and important scenarios. However there is some difference between the systems. The nominal teams working with the LiveNet-System found less critical but more important scenarios. Unfortunately a 2-tailed t-test resulted < 0,001. The allocation to a system has an impact too.

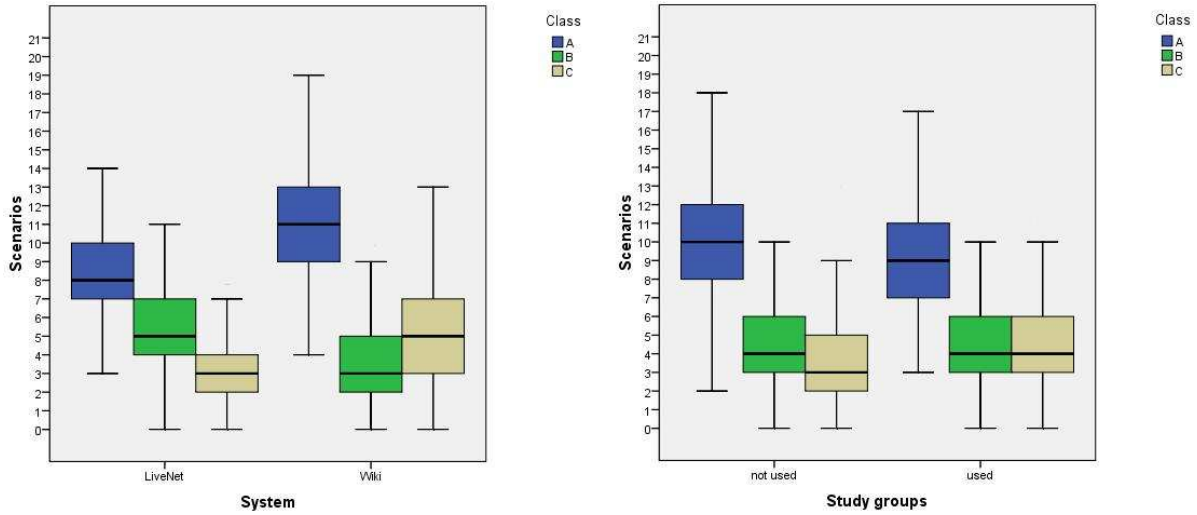


Figure 8.5.: Number of scenarios found by nominal teams

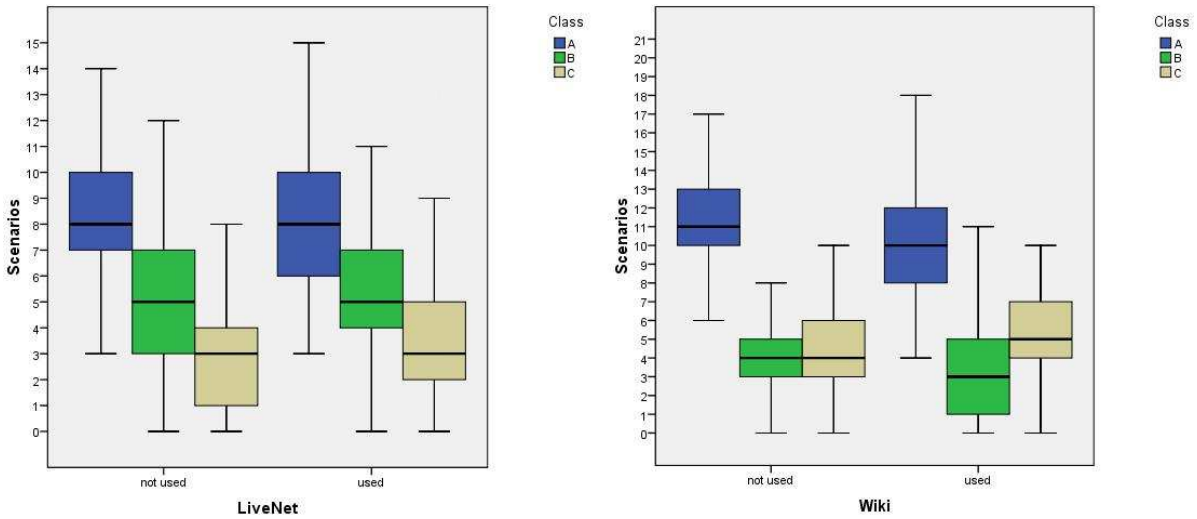


Figure 8.6.: Number of scenarios found by nominal teams for each system in specific

Comparing both real and nominal teams as visualized in figure *8.7*, shows considerable contrasts. The nominal teams performed much better in terms of the amount and quality of their scenarios. But real teams avoided less-important scenarios all together. Nevertheless, nominal teams do perform better than real teams. Even if the above factors and differences between nominal and real teams are taken into account, the difference is quite big.

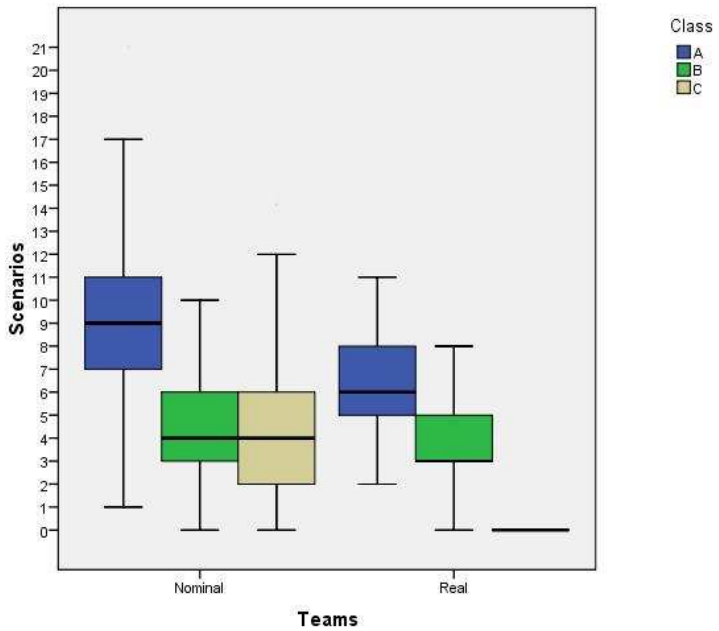


Figure 8.7.: Number of scenarios found by nominal teams compared to real teams

As already pointed out, real teams must be compared carefully to nominal teams. Time consuming actions like discussions and interaction have to be considered. But the difference in the performance leads to the conclusion that rivalry is going on within real teams, thereby reducing the output.

To find out how many scenarios are lost and gained, the output of the real teams is compared with the mean output of the nominal teams. If both systems are counted there are 37 real teams and 11075 computed nominal teams, so this approach was considered meaningful.

The figures *8.8* and *8.9* are representing the box plot diagram (figure *8.7*) in terms of scenario losses and gains of real teams compared to nominal teams (respectively their means). Real teams gain nearly no scenarios compared to the nominal teams.

This is true regardless which system was used or whether (change) scenario categories were provided or not. The biggest gap occurs at class C scenarios which is not surprising, as no real team had any class C scenario at all.

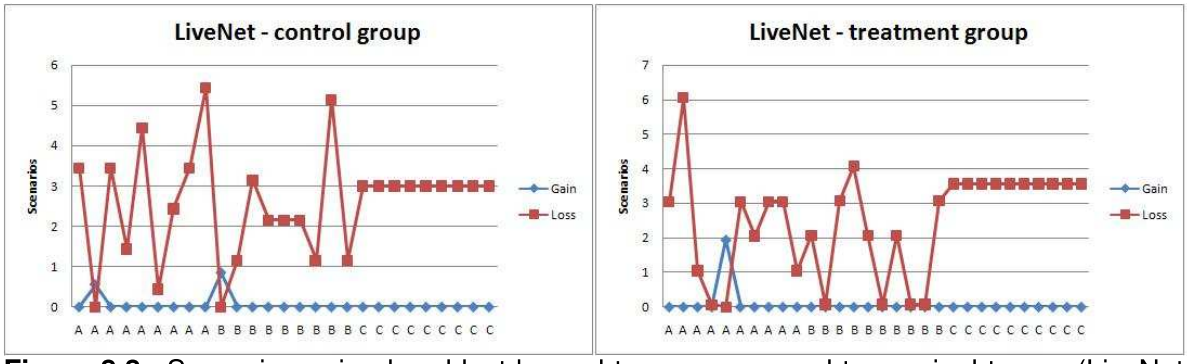


Figure 8.8.: Scenarios gained and lost by real teams compared to nominal teams (LiveNet-System)

However, overall these results are quite logical since the nominal teams represent somewhat the optimal solution as every scenario found by the team members was automatically put on the team scenario list. Nevertheless, could some real teams possibly have found more scenarios of one category or avoided less important scenarios?

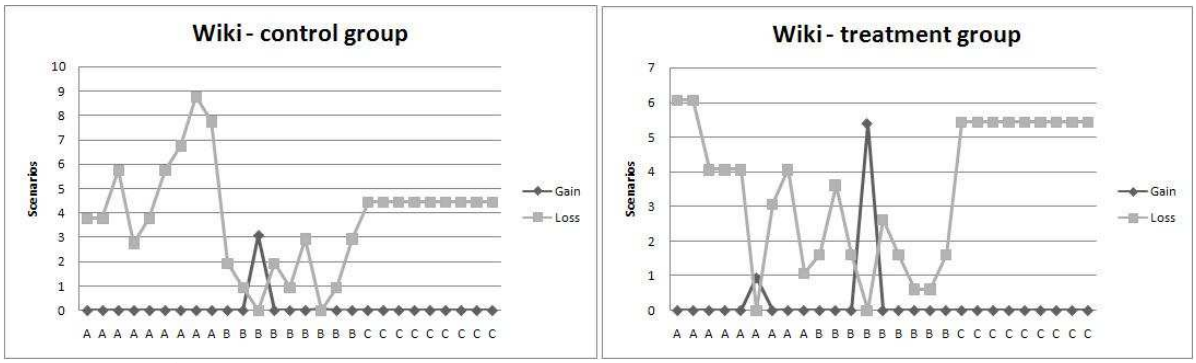


Figure 8.9.: Scenarios gained and lost by real teams compared to nominal teams (Wiki-System)

Judged from the viewpoint of the real teams, the best ratio between gains and losses occurred for class B scenarios. For every scenario gained a real team lost around 7 others. Looking at critical scenarios, the ratio shrank significantly. The real teams lost on average 34 scenarios when gaining one single scenario. By going deeper into the data pool, some surprising figures appeared.

System	Group	Critical Scenarios		Important Scenarios	
		Gains	Losses	Gains	Losses
LiveNet	CG	0,0622	2,7244	0,0944	2,0222
	TG	0,195	2,245	0,093	1,263
Wiki	CG	0	4,76	1,02	1,0711
	TG	0,1044	3,2756	0,4878	0,9867

Table 8.8.: Scenario gain and loss ratios of real teams

Teams belonging to control groups tend to have fewer losses than treatment group teams. Again, the ratios are much better regarding class B scenarios. Unfortunately, this indicates that real teams lose particularly critical scenarios in comparison to the nominal teams.

According to these figures, in particular less-important scenarios were eliminated. Real teams did not put any class C scenario on their lists. Unfortunately, this is the only advantage measured. As mentioned, within real team meetings a lot of scenarios got lost. Testing the influence of (change) scenario categories produces p-values of 0,455 (gains) and 0,243 (losses). It seems that there is also no influence by them. To sum it up, the support of (change) scenario categories had no influence on whether the real teams lost or gained scenarios.

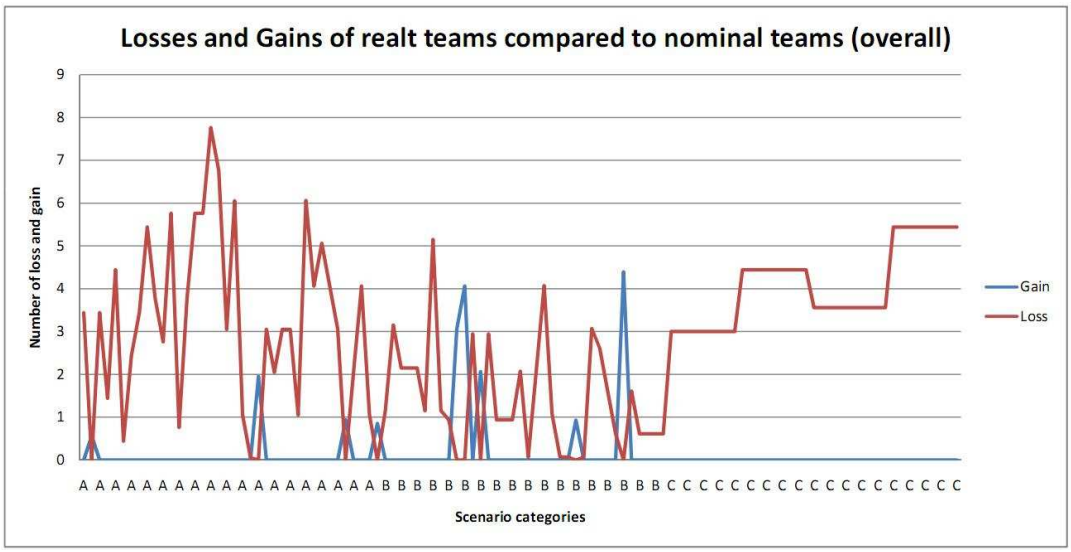


Figure 8.10.: Scenarios gained and lost by real teams compared to nominal teams

Comparison to the original study

The results are, in terms of the nominal teams differ compared to the original study but not the same. Nominal teams found much more scenarios compared to the individuals and the real teams. Also there is evidence that (change) scenario categories do help to find more and/or better scenarios. But the control group (no scenario categories given) achieved better results.

However, terms of losses and gains between the real and the nominal teams the findings are similar to the original study. The real teams found more scenarios than the individuals and they lost more A and B scenarios than they gained, compared to nominal teams. But this time the real teams eliminated all class C scenarios, giving them an advantage compared to the nominal teams.

8.2. Does experience help to find more or better scenarios?

Experience is an important factor in computer science. It is expected that individuals who have more experience achieve better results compared to less experienced ones. Within the section 7.3 “Individual setting” is explained how the individuals are categorized in terms of their experience.

8.2.1. Reference profiles

In order to evaluate whether experience improves the effectiveness or not, the quality of the scenarios found by each individual must somehow be determined. To be able to compare, the approach of the original study was applied again. Through this approach the scenario quality was scored by combining the scenario lists of the individuals with an overall top scenario list. The basic idea behind that list is similar to the classification of Class A, B and C scenarios. The more often a scenario was found, the more important it should be.

Reference Scenario Profile LiveNet		
Score	F	Scenario Description
15	45	Integration of external communications (e.g. video conference, VoIP, ...)
14	40	Management of access rights
13	38	Versioning (undo function); data storage + old version viewing; viewing of changes
12	34	Scalability/Portability (e.g. distributed systems)
11	16	Converting to other file formats (pdf, ...);
10	16	Livenet portable for mobile devices (small screens)
9	14	Multiple access of many users to documents
8	14	Automated notification about new/changed artifacts in group
7	14	Save chat history for later reviews
6	13	New roles defined and implemented
5	13	Database enlargement (for big documents, forum, users, files)
4	11	Offline Mode;- local saving/sync function
3	10	Support of various (new) evaluation techniques
2	9	Support different browser (same display)
2	9	Ajax or similar technology for better performance / efficiency

Table 8.9.: Reference profile list LiveNet

Again, all unique found scenarios (see classification in [] Experiment_Description) were put into a pool. Afterwards, the frequency of each scenario was counted. Then, the scenarios obtaining the highest frequency were collected and sorted into the reference profile list. In the original study, all in all 10 scenarios found their way into the profile list. There, 108 unique scenarios have been found. In this study 161 and 172 (LiveNet and Wiki) scenarios, found at

individual brainstorming and team sessions, had to be considered. To accommodate with this increase of 60 percent, the scenario profile list was increased by 5 additional scenarios.

Figures *8.10* and *8.11* show the two reference profile lists for both systems, LiveNet and Wiki. "F" stands for the frequency of the respective scenario. Only these scenarios are scored. Logically, the scenario with the highest frequency gets the highest score. As there are 15 items, the maximum amount of points for the top scenario is 15 dropping down to 1 point for the lowest scenario.

Reference Scenario Profile Wiki		
Score	F	Scenario Description
15	56	Supporting different kinds of media files, e.g. pictures, videos, sounds (within text)
14	43	Online editor for graphics, formulas, texts, ... (tinyMCE (WYSIWYG), possibility to pimp content)
13	33	Management of access rights
12	31	Versioning (undo function); data storage + old version viewing; viewing of changes
11	25	Make Application/Layout customizable to adapt to corporate identity; General layout-management possible;
10	24	Subscribing to change history through RSS feed;
9	21	Possibility to rate articles + average rating of all the articles of one author
8	14	Imply search function based on meta-data (xml)
7	13	Multilingual
6	13	Converting to other file formats (pdf, ...);
5	12	Scalability/Portability (e.g. distributed systems)
4	12	Ban user
3	10	Detailed settings for notification (prompt, periodical, subscribing only for certain categories only, ...)
2	10	Using HTML and CSS tags for writing articles, also Latex;
1	9	Ajax or similar technology for better performance / efficiency
1	9	Support for adding attachments to pages
1	9	Make pages non editable / lock them for specific users and/or IP-addresses

Table 8.10.: Reference profile list Wiki

This approach is different to the one applied in the original study. The reason is that if the frequency would result the scenario score in the study replication the values would be very large and huge gaps would occur. For example, the highest scenario score for the LiveNet system would have been 45, the second highest 40. And the fifteenth score in the reference list would only be 9. So the above method was chosen to get more conservative scores and introduce a linear scoring system which is more comparable and repeatable.

Reference profile score of the individuals

System	Categories	Mean	N	Std. Deviation	Median
LiveNet	not used	24,11	27	15,863	24,00
	used	26,30	27	12,477	29,00
	Total	25,20	54	14,179	27,00
Wiki	not used	32,28	25	16,157	29,00
	used	29,33	27	15,974	26,00
	Total	30,75	52	15,973	28,00
Total	not used	28,04	52	16,374	27,00
	used	27,81	54	14,279	28,00
	Total	27,92	106	15,269	27,50

Figure 8.11.: Individual scores concerning the reference profile

The data-results of table *8.11* show the remarkable fact that the individuals produced higher scores using the Wiki-system regardless to which study group they belonged. Interestingly, the control group performed slightly better than the treatment group. For the LiveNet-System the opposite is true. The treatment group produced better results despite a worse score overall.

Since more scenarios were also found for the Wiki-system, it seems that people performed better on well-known or easy to handle software architectures/programs. Further, providing them with predefined scenario categories does not have a positive impact on the results. But this thought cannot be proven with the available data set.

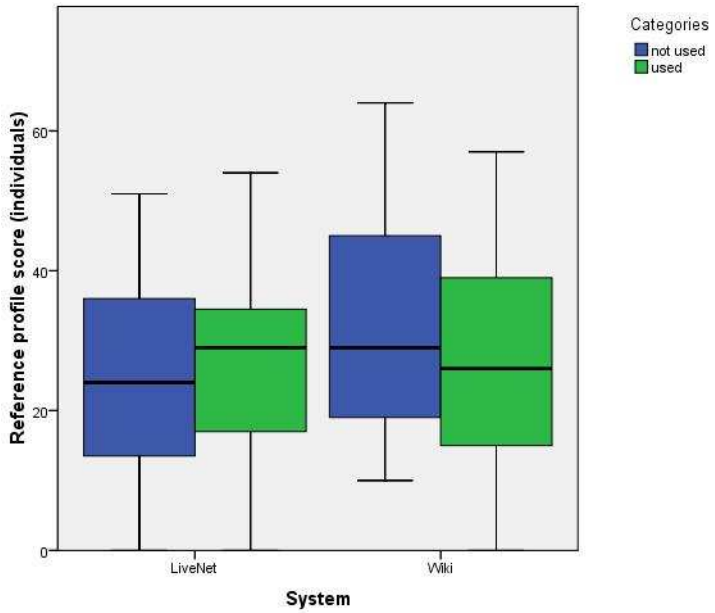


Figure 8.12.: Box plot of individual scores concerning the reference profile

There is no statistically significant difference between the two systems means referring to a p-value of 0,206 at a 95% confidence interval. Looking at the box plot it is easy to see that providing scenario categories does not help to produce better results. Just like with the system, there is a statistical difference between the control group and the treatment group resulting in a p-value of 0,184 which rejects the null hypothesis that there exists a statistical difference between the groups.

Comparison to the original study

Reviewing the data and the figures the findings of the original study were not replicable. Regarding the LiveNet-System the control group did find more scenarios but this is not true considering the Wiki-System. Overall providing change scenarios does not help the individuals to perform better. This matches the findings under section 8.1.1 “Scenarios found by individuals” above.

8.2.2. Experience and scenario quality

It is important to explore whether possible differences in the results could be produced or influenced by the respective experience of the treatment and the control group. As figure *8.13* shows, it is expected that no divergences should occur due to differing experience.

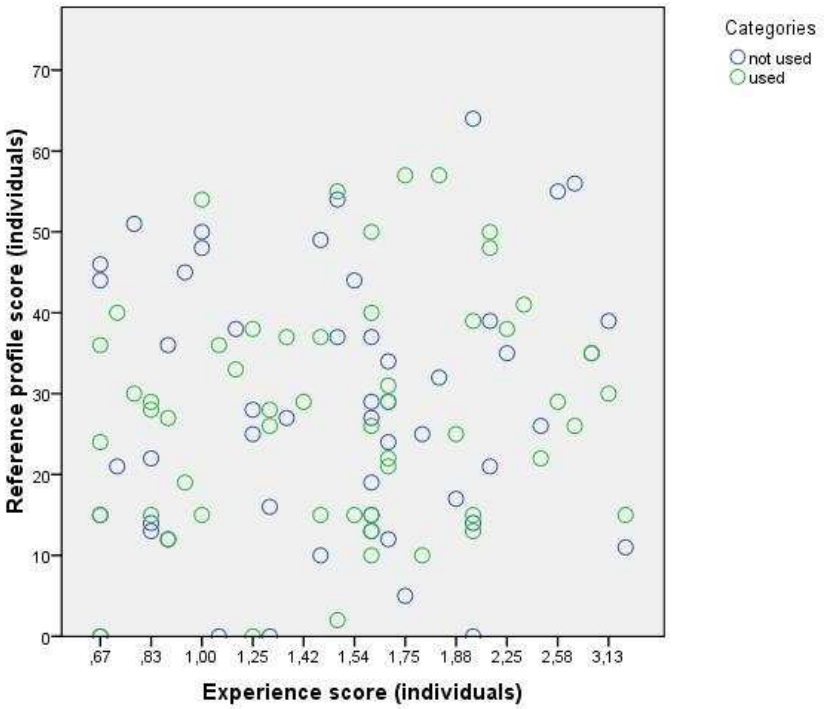


Figure 8.13.: Experience against scenario score

The correlation “r” between experience and scenario score is 0,052 indicating that there is no (linear) correlation. This is supported by a p-value of 0,866. Surprisingly, this could state that IT experience does not have any influence on the quality of the found scenarios. But this figure is only true for all individuals overall, regardless of any system or study group. The next step is to investigate if the situation is the same compared to all three experience classes or if within them there are significant correlations. Further, there two systems could produce correlations too.

LiveNet

Analyzing the box plot for the LiveNet-System gives rise to the speculation that experience does have some impact on individuals. Further this seems to be true regardless if they get support through provided change categories, or not. Interestingly, the performance of the individuals allocated to the treatment group with regular experience is nearly the same. Contrarily, in the control group well experienced participants reached significantly better results. On the other hand, within the treatment group both extreme cases (Class A and C experience) were conspicuously moderate. The well experienced individuals had scored lower and the less experienced ones reached much better scores on average compared to the control group. A p-value testing the impact of scenario categories scored 0,085. Considering a 95 % confidence interval this states that there is no statistical influence by them. But on a 10 % interval this null hypothesis had to be rejected.

Now testing the impact of experience on the scenario score (both study groups regarded) results in a p-value of 0,683. Only viewing the control group the same test achieves a score of p = 0,883 whereas for the treatment group the results is p = 0,807. The figures show no impact of experience on scenario scores. This is astonishing as well experienced individuals achieved the highest scores. But as experienced participants scored lower than less experienced ones the overall pictures seems to be differentiate.

Reference profile score in relation to experience (LiveNet)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	27,00	3	13,856	35,00
	B	23,50	18	15,621	25,50
	C	24,50	6	19,887	18,00
	Total	24,11	27	15,863	24,00
used	A	29,60	5	7,092	29,00
	B	25,29	17	14,717	29,00
	C	26,40	5	8,792	28,00
	Total	26,30	27	12,477	29,00
Total	A	28,63	8	9,242	29,50
	B	24,37	35	14,992	27,00
	C	25,36	11	15,154	27,00
	Total	25,20	54	14,179	27,00

Table 8.11.: Reference profile score of individuals in relation to their experience (LiveNet)

Wiki

The above results are somewhat confirmed as shown in the box plot for the Wiki-System. Again in class A the experienced individuals reached a higher score in the control group. And although the picture is not the same, it is clearly similar. But this time the p-value testing the impact of the scenario categories reached 0,850. Within the Wiki-System no significant influence was measured.

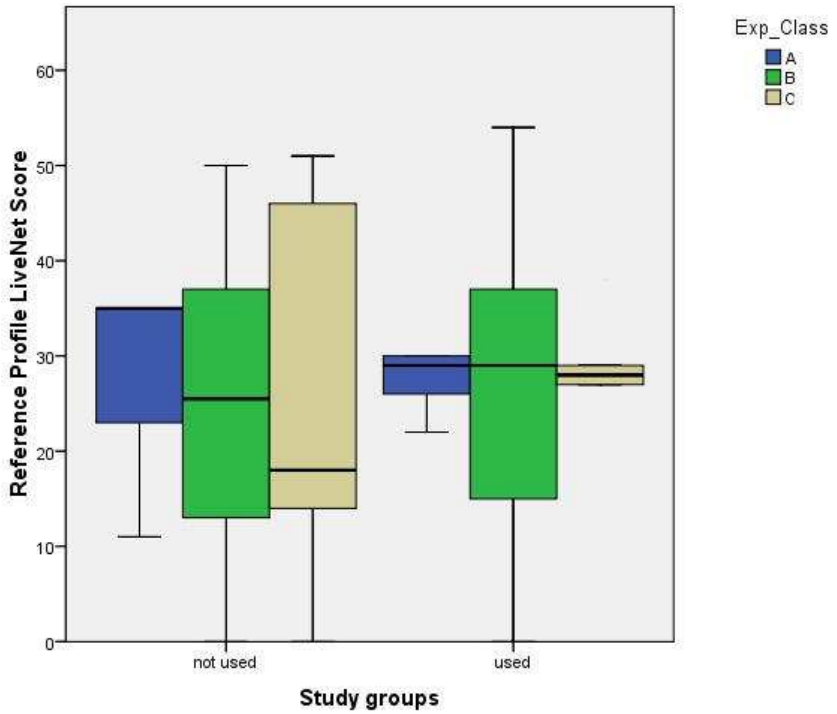


Figure 8.14.: Experience classes in relation to scenario score of the LiveNet-System

Switching to the experience a p-value of 0,121 indicates no influence on the scenario score. Isolating the control group the p-value is 0,224 and vice-versa (treatment group) the result of the t-test is p = 0,320. The figures of the LiveNet-System are calculated again which leads to the same statements for the Wiki-System. The experience of the individuals does influence the scenario score.

Reference profile score in relation to experience (Wiki)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	44,00	4	14,306	47,00
	B	31,50	16	16,436	28,50
	C	25,40	5	14,170	22,00
	Total	32,28	25	16,157	29,00
used	A	29,33	3	12,503	35,00
	B	32,22	18	16,728	27,00
	C	20,67	6	13,880	19,50
	Total	29,33	27	15,974	26,00
Total	A	37,71	7	14,694	38,00
	B	31,88	34	16,342	28,00
	C	22,82	11	13,519	22,00
	Total	30,75	52	15,973	28,00

Table 8.12.: Reference profile score of individuals in relation to their experience (Wiki)

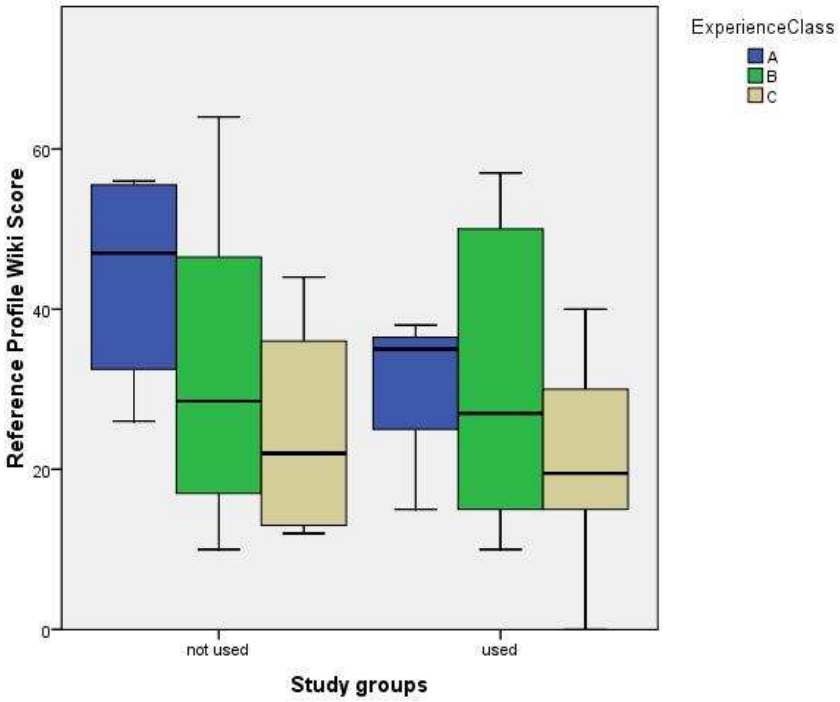


Figure 8.15.: Experience classes in relation to scenario score of the Wiki-System

Both systems

Putting the systems aside and looking at the descriptive data overall, the well experienced individuals performed better in terms of critical scenarios within the control groups. Comparing important scenarios the results are vice-versa. A p-value of 0,206 at a 95 % confidence interval indicates that the systems do not have a significant impact on the individual sores.

Hence, it is rather by chance that the individuals of one system have higher or lower scores compared to others working with another system. Testing interdependencies between the scenario score and the type of group (control or treatment) the p-value is 0,185. Like the system, giving the support of (change) scenario categories does not to have any influence on the quality score.

These findings lead to the conclusion that a framework similar to scenario categories is not of any help if it is the aim to find more critical scenarios or if less important scenarios should be avoided. In terms of the impact of experience the statistical figures show no influence on the scenario score.

As already mentioned, one possible factor could be the system itself. Possibly, if an individual is very familiar with one system he achieves a higher quality score and maybe more scenarios. There is no data available to confirm that the individuals knew the Wiki-System better than the LiveNet-System. But since Wikipedia is a very popular software used world-wide and LiveNet is only a small system built for research purposes, it is most likely that the individuals

Reference profile score of individuals (both systems combined)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	36,71	7	15,777	35,00
	B	27,26	34	16,277	27,00
	C	24,91	11	16,682	21,00
	Total	28,04	52	16,374	27,00
used	A	29,50	8	8,569	29,50
	B	28,86	35	15,943	28,00
	C	23,27	11	11,671	27,00
	Total	27,81	54	14,279	28,00
Total	A	32,87	15	12,541	35,00
	B	28,07	69	16,009	27,00
	C	24,09	22	14,074	23,00
	Total	27,92	106	15,269	27,50

Table 8.13.: Reference profile score of individuals both systems and study groups combined

had much more knowledge about the Wiki-System.

Comparison to the original study

The results of the study replication do not affirm those of the original study. The treatment group had not achieved significantly better scenario scores. The same is true for the impact of experience. But the descriptive data shows that well experienced individuals are achieving the best scenario scores. Perhaps further research work can solve this strange picture.

8.2.3. Impact of participant experience on individual scenario brainstorming effectiveness

As above applied reference profile showed no significant impact of experience, the effectiveness of the individuals was measured in terms of individual experience. Therefore, the primary scenario scoring technique (frequency-based) was used again.

Additionally it was confronted with a scenario ranking made by experts. Possibly the well-experienced participants show better results when critical scenarios, ranked by experts, are used. Because when applying a frequency-based system, actually minor important scenarios can get high ranks if many (minor experienced) students found them.

The two categorization techniques are discussed in section 7.5 “Scenario rating”. The number of assigned scenarios to each classification is shown below in table *8.14*. Astonishingly, the system of classification of the two techniques derivates from the set 20:40:40 to 20:60:20.

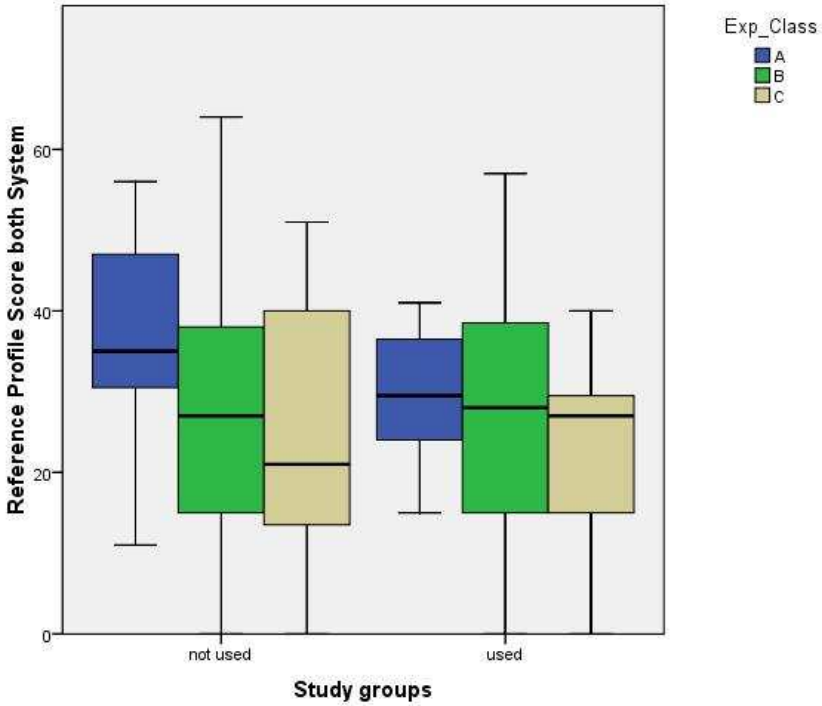


Figure 8.16.: Experience classes in relation to scenario score of both systems

	SC-Frequency				SC-Expert			
	LiveNet		Wiki		LiveNet		Wiki	
Scenarios	No.	Share	No.	Share	No.	Share	No.	Share
Class A	31	18 %	39	20 %	36	21 %	35	19 %
Class B	84	48 %	68	35 %	96	57 %	100	56 %
Class C	59	34 %	86	45 %	37	22 %	45	25 %
Total	174	100 %	193	100 %	169	100 %	180	100 %

Table 8.14.: Scenarios assigned to classification sorted by classification technique

Further, the variation concerning the systems within one technique is minor for the expert based one. Note that this study, like the original one, focuses more on Class A, B & C scenarios for the following calculations for evaluation purposes. As already mentioned in this work, due to external limitations (time & money), mainly class A scenarios and some class B scenarios will most likely be considered. Nevertheless this study replication considered all three categories for the sake of completeness.

Generally, effectiveness is calculated by looking at the number of scenarios found by an individual in relation to the scenarios developed overall and results in a percentage point. Of course, the scenario category and the experience category are considered.

Comparing the two systems (looking at both systems as a whole) which each other, the descriptive data clearly shows that well experienced individuals are most effective. Surprisingly, the effectiveness of a group with experienced individuals is very similar compared to groups

with less experienced people. If the groups are divided and looked at separately this result only holds true for the control groups.

LiveNet

Concentrating on the LiveNet-System and applying the frequency-based scenarios ranking system the descriptive data shows that, regardless of the scenario importance and the study groups, well experienced individuals were most efficient.

A t-test checking the influence of the scenario categories scored 0,800 which indicate that giving such categories has no impact on the effectiveness. Testing experience score against effectiveness (both SC-F and SC-E) achieved a p-value of < 0,001 stating that experience has a huge impact on the individual performance.

Effectiveness of the individuals in relation to scenario class (LiveNet/SC-F)

Scen_Class	Exp_Class	Mean	N	Std. Deviation	Median
A	A	12,5000	8	4,02072	12,9032
	B	10,5991	35	5,83230	12,9032
	C	9,3842	11	7,41999	6,4516
	Total	10,6332	54	5,93255	9,6774
B	A	2,8274	8	1,90237	2,3810
	B	1,9937	35	1,91589	1,1905
	C	2,1645	11	2,11745	2,3810
	Total	2,1520	54	1,94014	2,3810
C	A	3,8136	8	4,41515	2,5424
	B	1,4820	35	1,81540	1,4706
	C	1,5408	11	2,06954	1,6949
	Total	1,8394	54	2,48759	1,6949
Total	A	6,3803	24	5,62699	4,9233
	B	4,6916	105	5,57306	2,3810
	C	4,3632	33	5,74549	2,3810
	Total	4,8749	162	5,61790	3,0835

Effectiveness of the individuals in relation to the study groups (LiveNet/SC-F)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	7,3566	9	6,11633	5,0847
	B	4,3391	54	5,58553	1,6949
	C	4,9050	18	6,24970	3,4806
	Total	4,8001	81	5,79577	2,3810
used	A	5,7946	15	5,44615	3,5714
	B	5,0648	51	5,59074	3,3898
	C	3,7129	15	5,21477	1,6949
	Total	4,9496	81	5,46936	3,2258
Total	A	6,3803	24	5,62699	4,9233
	B	4,6916	105	5,57306	2,3810
	C	4,3632	33	5,74549	2,3810
	Total	4,8749	162	5,61790	3,0835

Table 8.15.: Individual effectiveness - LiveNet/SC-F

On viewing the box plots it can be clearly seen that experience increases the effectiveness. Unfortunately the allocation to a certain study group involving the help of scenario categories did not produce better results.

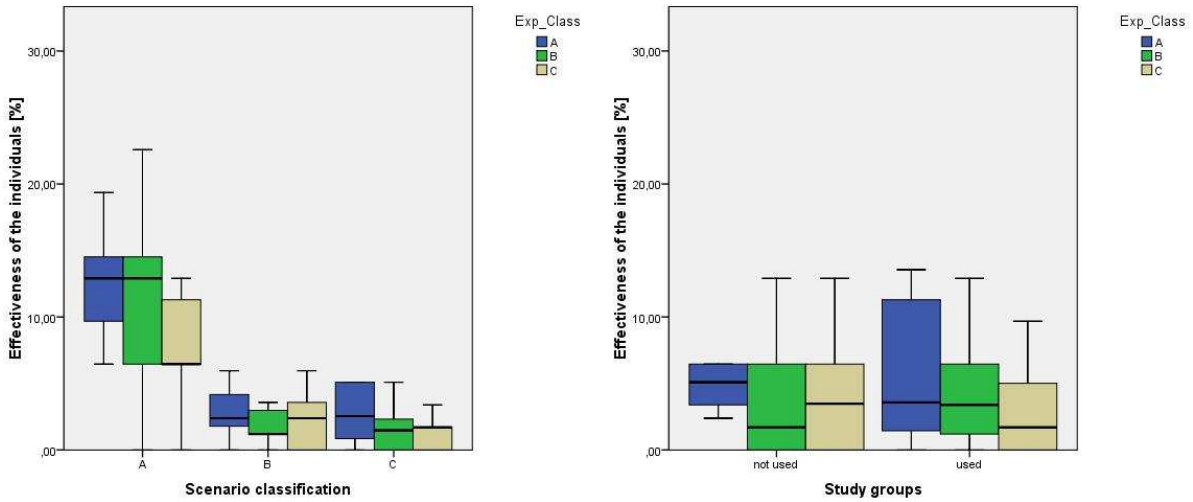


Figure 8.17.: Impact of experience on the effectiveness for the LiveNet-System/SC-F (Individual level)

Moving on to the expert-based scoring system shows more or less the same picture. Still well experienced participants achieved the best results. And again providing (change) scenario categories has no impact on the effectiveness (p-value 0,615). The box plots provide a better oversight.

As expected, the well experienced individuals achieved a higher effectiveness on applying an expert ranking system. However, somehow all participants gained more efficiency. Perhaps this is a result of the fact that the experts had ignored some scenarios because they had been not usable.

Wiki

Regarding the Wiki-System, while using the frequency-based ranking system, the descriptive data is similar to the LiveNet-System. Again the more experienced an individual was the more effective he worked.

In terms of class A scenarios surprisingly experienced participants achieved nearly the same results than well experienced ones. Regarding the median, they are even slightly better. But overall the picture is the same as for the LiveNet-System. Unfortunately the scenario categories still did not influence the effectiveness. A p-value of 0,189 affirms this statement.

Again testing experience score in relation to effectiveness scored in both cases (SC-F and SC-E) < 0,001. Like for the LiveNet-System experience does influence the effectiveness of the

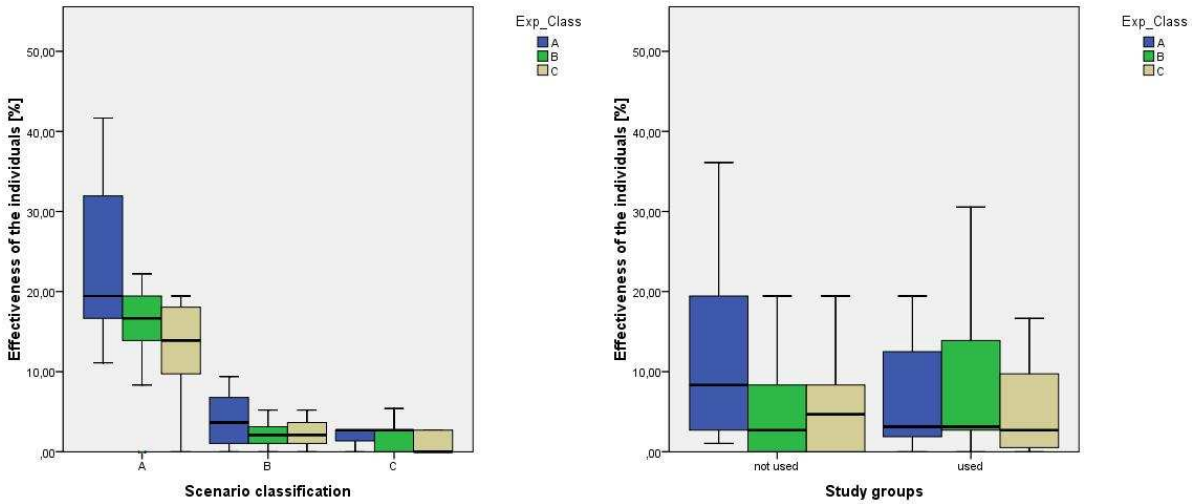


Figure 8.18.: Impact of experience on the effectiveness for the LiveNet-System/SC-E (Individual level)

individuals.

The box plots visualize the findings. The similarity to the LiveNet-System can be seen easily. Remarkable is the box plot showing the effectiveness between the two study groups. Both, the treatment and the control group, achieved very close results. This indicates that the (change) categories did not affect the individuals.

Switching to the expert-based ranking system does not change the big picture. Again a p-value of 0,455 states no impact of giving scenario categories to the individuals. Very astonishing is the fact that all medians for class B scenarios are very much the same. No explanation for this phenomenon can be given.

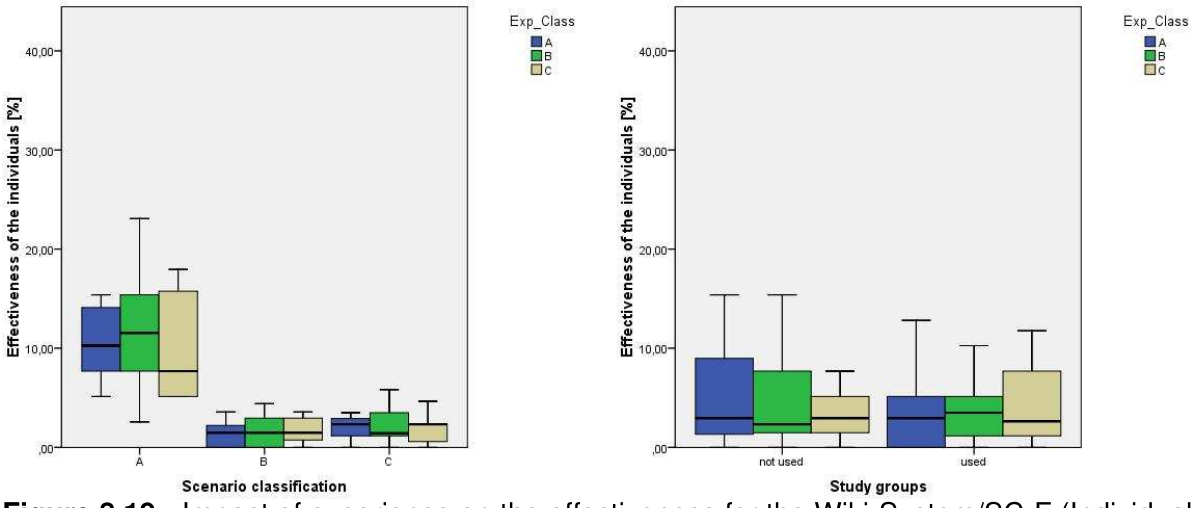


Figure 8.19.: Impact of experience on the effectiveness for the Wiki-System/SC-F (Individual level)

Effectiveness of the individuals in relation to scenario score (LiveNet/SC-E)

Scen_Class	Exp_Class	Mean	N	Std. Deviation	Median
A	A	23,6111	8	10,70693	19,4444
	B	15,8730	35	7,33606	16,6667
	C	15,6566	11	11,06049	13,8889
	Total	16,9753	54	8,97538	16,6667
B	A	4,0365	8	3,45201	3,6458
	B	2,4702	35	1,82332	2,0833
	C	2,6515	11	2,52434	2,0833
	Total	2,7392	54	2,28304	2,0833
C	A	3,0405	8	3,36871	2,7027
	B	2,1907	35	2,13119	2,7027
	C	2,2113	11	3,15608	,0000
	Total	2,3208	54	2,52661	2,7027
Total	A	10,2294	24	11,64351	4,6875
	B	6,8447	105	7,83113	3,1250
	C	6,8398	33	9,13506	3,1250
	Total	7,3451	162	8,77633	3,1250

Effectiveness of the individuals in relation to the study groups (LiveNet/SC-E)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	12,6814	9	12,42716	8,3333
	B	6,0790	54	7,38667	2,7027
	C	8,4157	18	11,16452	4,6875
	Total	7,3319	81	9,08967	2,7027
used	A	8,7581	15	11,32660	3,1250
	B	7,6554	51	8,27158	3,1250
	C	4,9487	15	5,68858	2,7027
	Total	7,3583	81	8,50813	3,1250
Total	A	10,2294	24	11,64351	4,6875
	B	6,8447	105	7,83113	3,1250
	C	6,8398	33	9,13506	3,1250
	Total	7,3451	162	8,77633	3,1250

Table 8.16.: Individual effectiveness - LiveNet/SC-E

Viewing the box plots does not provide any further answers to this. Contrarily all means and standard deviations are different. Probably the scenario ranking list for the Wiki-System is rather likewise to the frequency-based list.

However, analyzing the systems combined the picture is blurred. Well experienced participants worked more effectively on the LiveNet-System. The statement is supported by a p-value of 0,042 testing the overall effectiveness depending on the overall experience score. Remarkable is the fact that for experienced and less experienced individuals the effectiveness is closer between both systems. Astonishingly, testing the influence of experience within the systems, the p-values change to 0,149 for the LiveNet-System and 0,601 for the Wiki-System.

Applying a p-test shows that differences between the systems can be expected (p-value: 0,038). This might be the answer to the changing t-test results above. The gain or loss in effectiveness cannot be explained by the experience of the individuals alone, the systems play a certain role too.

Effectiveness of the individuals in relation to scenario score (Wiki/SC-F)

Scen_Class	Exp_Class	Mean	N	Std. Deviation	Median
A	A	12,8205	7	8,63206	10,2564
	B	11,4630	34	5,09841	11,5385
	C	10,0910	11	5,51132	7,6923
	Total	11,3555	52	5,67350	10,2564
B	A	1,3505	7	1,46987	1,4706
	B	1,7734	34	1,65031	1,4706
	C	2,4637	11	3,32184	1,4706
	Total	1,8625	52	2,07349	1,4706
C	A	2,3256	7	1,89883	2,3256
	B	1,9436	34	1,55815	1,4289
	C	1,7970	11	1,50388	2,3256
	Total	1,9640	52	1,56934	2,0102
Total	A	5,4989	21	7,23832	2,9412
	B	5,0600	102	5,55729	2,7526
	C	4,7839	33	5,31471	2,9412
	Total	5,0607	156	5,72349	2,9412

Effectiveness of the individuals in relation to the study groups (Wiki/SC-F)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	6,7567	12	8,83669	2,9485
	B	5,3161	48	5,88884	2,3256
	C	4,5479	15	5,04643	2,9412
	Total	5,3930	75	6,23708	2,3256
used	A	3,8218	9	4,25159	2,9412
	B	4,8323	54	5,29043	3,4884
	C	4,9806	18	5,66628	2,6334
	Total	4,7530	81	5,22362	3,4884
Total	A	5,4989	21	7,23832	2,9412
	B	5,0600	102	5,55729	2,7526
	C	4,7839	33	5,31471	2,9412
	Total	5,0607	156	5,72349	2,9412

Table 8.17.: Individual effectiveness - Wiki/SC-F

Comparison to the original study

Assuming that the individuals were much more familiar with the Wiki-System, it seems that experience still does have a relevant impact on the effectiveness of the individuals. But the support of change scenario categories does not. This is contrary to the original study where the (change) scenario categories did influence the individuals.

But, as mentioned, experience seems to have some impact on the effectiveness of the individuals (during a brainstorming session). The findings of the original study concerning this matter have been not replicable.

Effectiveness of the individuals in relation to scenario score (Wiki/SC-E)

Scen_Class	Exp_Class	Mean	N	Std. Deviation	Median
A	A	11,0204	7	4,49598	8,5714
	B	10,2521	34	5,12636	8,5714
	C	9,3434	11	5,29218	8,5714
	Total	10,1633	52	5,01251	8,5714
B	A	3,2857	7	3,03942	2,0000
	B	2,7966	34	1,88642	2,0000
	C	2,7348	11	2,57035	2,0000
	Total	2,8494	52	2,17157	2,0000
C	A	1,9048	7	1,53348	2,2222
	B	1,9749	34	1,71373	2,2222
	C	2,2659	11	1,72741	2,2222
	Total	2,0270	52	1,66681	2,2222
Total	A	5,4036	21	5,14175	4,0000
	B	5,0079	102	4,97086	3,0000
	C	4,7814	33	4,74536	2,7778
	Total	5,0132	156	4,91868	3,0000

Effectiveness of the individuals in relation to the study groups (Wiki/SC-E)

Categories	Exp_Class	Mean	N	Std. Deviation	Median
not used	A	6,6997	12	5,99065	4,7222
	B	4,9688	48	5,03728	3,0000
	C	4,3392	15	4,26844	2,7027
	Total	5,1198	75	5,04389	3,0000
used	A	3,6755	9	3,29673	2,2222
	B	5,0426	54	4,95819	3,0000
	C	5,1499	18	5,20258	3,6111
	Total	4,9146	81	4,82923	3,0000
Total	A	5,4036	21	5,14175	4,0000
	B	5,0079	102	4,97086	3,0000
	C	4,7814	33	4,74536	2,7778
	Total	5,0132	156	4,91868	3,0000

Table 8.18.: Individual effectiveness - Wiki/SC-E

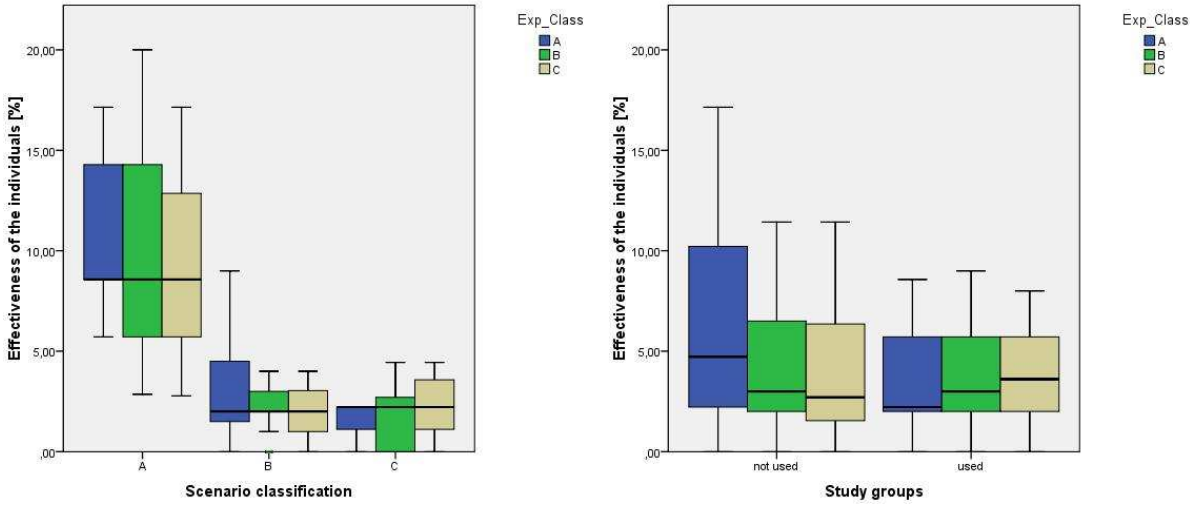


Figure 8.20.: Impact of experience on the effectiveness for the Wiki-System/SC-E (Individual level)

8.2.4. Comparison of scenario scoring and expert scoring

Now, as figures for the frequency-based ranking system and the expert-based ranking system are similar the question is whether it is meaningful in an industrial context to charge (external) experts for scenario ranking, or not.

As a first step, the frequency oriented and the expert oriented techniques are compared with regard to their tasks of assigning scenarios to a specific category. “Similar Classification” comprises all the scenarios which are classified equally as the same or similar in both techniques. For the LiveNet-System 161 scenarios and for the Wiki-System 170 scenario could have been compared. The reason was that only scenarios which were in both ranking lists were usable for comparison.

LiveNet

About 43 % of all scenarios have been ranked similar in both ranking approaches. “Closely Matched” summarizes all the scenarios with only minor differences within their classification. Basically this involves the cases A-B, B-C and vice versa. Here, 53 % have been collected. The variable “Different” counts those scenarios which are rated completely differently (A-C and C-A). Only 4 % of all the scenarios fit this category.

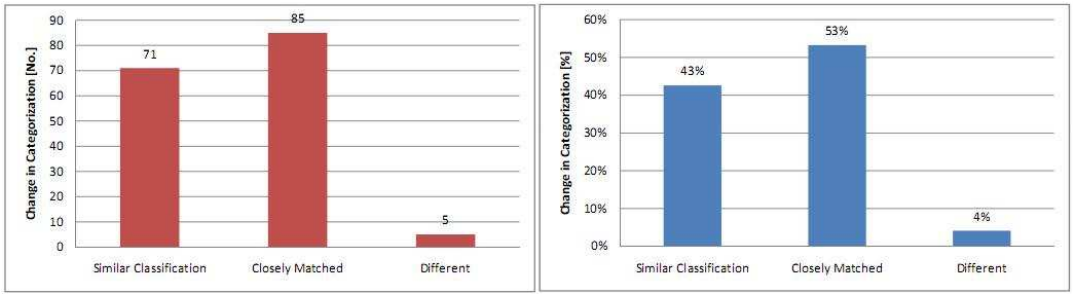


Figure 8.21.: Comparison of SC-F and SC-E for the LiveNet-System

Wiki

Contrary to the LiveNet-System “Similar Classified” scenarios are most common category with 49 %. “Closely Matched” ones are second with 45% but are still very close to the top. The group “Different” scenarios is the lowest, with a rating equally to the LiveNet-System.

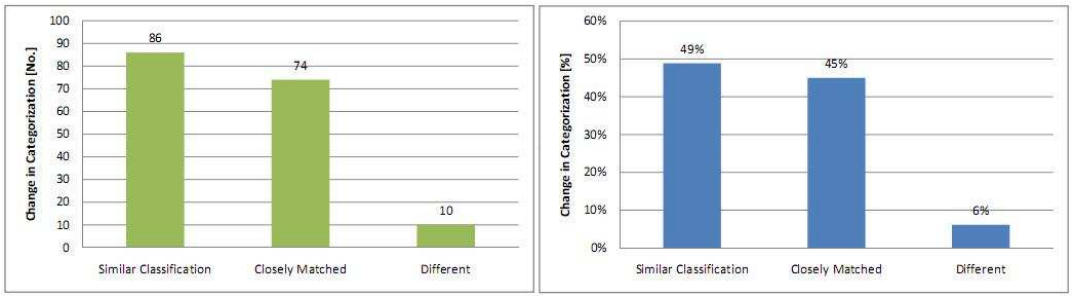


Figure 8.22.: Comparison of SC-F and SC-E for the Wiki-System

Both Systems

Thus the findings of both systems are very close to each other stating that overall, the participants classified the scenarios similar to the experts. Hence, this suggests that a frequency-based prioritization can substitute an expert rating.

Comparison to the original study

Like in the original study the figures state that the individuals ranked the scenarios similar compared to experts. Getting back to the industrial context it can be stated, that it is not worth to hire expensive experts for scenario classification - if the individuals developing the scenarios are selected thoroughly.

8.3. Does increasing the team size help to improve the effectiveness of scenario development?

Two heads are better than one. That teams find more scenarios than individuals has been proven above. But in industrial projects the cost are often a key factor. So the efficiency of such teams is important.

Individual brainstorming offers the advantage that everyone can think without any interruption or influence. On the other hand, serious disadvantages have to be considered. Several important aspects are missing: The broader experience of team (involvement of more brains), avoiding of mistakes and overlapping scenarios or different points of views (e.g. of different stakeholders) etc.

According to the findings so far teams perform especially well regarding change scenarios of higher quality. Therefore only class A and class B scenarios are considered for measurement.

Theoretically, teams should perform better than individuals. Nevertheless, this will most likely not be a linear growth. Moreover it is expected to see some kind of turning point where there will be only slightly better results for every growth in team size.

Descriptive Data

As expected, every new team member improves the effectiveness of the scenario brainstorming (theoretically). The highest gain in effectiveness is achieved by adding a second team member. Increasing the team size more and more lowers the growth rate of a productive output. Nevertheless, the improvement in the effectiveness of the scenario brainstorming remains still significant. The findings hold true for both systems as well as for the treatment and control groups within the system.

Mean	Teamsize	1	2	3	4	5	6	7
LiveNet	TG, Class A	10,63%	19,06%	25,98%	31,86%	37,01%	41,49%	45,60%
	CG, Class A	10,62%	19,60%	27,22%	33,74%	39,36%	44,65%	48,91%
	TG, Class B	2,36%	4,71%	7,03%	9,34%	11,63%	13,81%	16,02%
	CG, Class B	2,41%	4,79%	7,14%	9,45%	11,73%	13,86%	16,08%
Wiki	TG, Class A	13,02%	23,58%	32,44%	40,07%	46,76%	52,28%	57,72%
	CG, Class A	15,48%	27,80%	37,94%	46,54%	54,00%	60,88%	66,69%
	TG, Class B	2,46%	4,40%	6,55%	8,67%	10,75%	15,45%	17,67%
	CG, Class B	2,22%	4,86%	7,16%	9,40%	11,56%	13,45%	15,51%

Std.Dev.	Teamsize	1	2	3	4	5	6	7
LiveNet	TG, Class A	5,36%	6,62%	7,29%	7,66%	7,88%	8,14%	8,22%
	CG, Class A	6,57%	7,75%	8,14%	8,17%	8,03%	7,63%	7,47%
	TG, Class B	1,81%	2,45%	2,91%	3,27%	3,56%	3,74%	3,90%
	CG, Class B	2,03%	2,73%	3,23%	3,61%	3,89%	4,10%	4,29%
Wiki	TG, Class A	6,49%	7,78%	8,38%	8,66%	8,77%	8,63%	8,65%
	CG, Class A	7,63%	8,71%	9,11%	9,26%	9,28%	9,13%	9,14%
	TG, Class B	3,24%	4,38%	5,20%	5,83%	6,32%	7,16%	7,29%
	CG, Class B	1,74%	2,31%	2,70%	2,99%	3,23%	3,63%	3,53%

Table 8.19.: Impact of team size on brainstorming effectiveness - Descriptive Data

Along with the shrinking of the growth rate, the standard deviation increases when the team size is enlarged. Hence, the larger the team size, the more certain are the calculations. The relation between both figures is not constant meaning that the standard deviation gets less important with every increase of the team size. It is important to add that the calculations for the teams with 6 and 7 team members are based on samples as the data set had become too large for calculations with desktop database programs (here MS-Access).

LiveNet

Interpreting the diagrams for several mean values and standard deviations, the similarity between the treatment and the control group is salient. While the growth rate of the mean value for class A scenarios flattens during increasing the team size, almost no decrease for class B scenarios is noticeable. The curves for the treatment group and the control group regarding class B scenarios are so close that they are overlaying.

However, the growth rate of the standard deviation decreases more at the beginning and levels off more and more after the team size has been further increased. Notice that the curves for the treatment and the control group for class B are nearly exately the same.

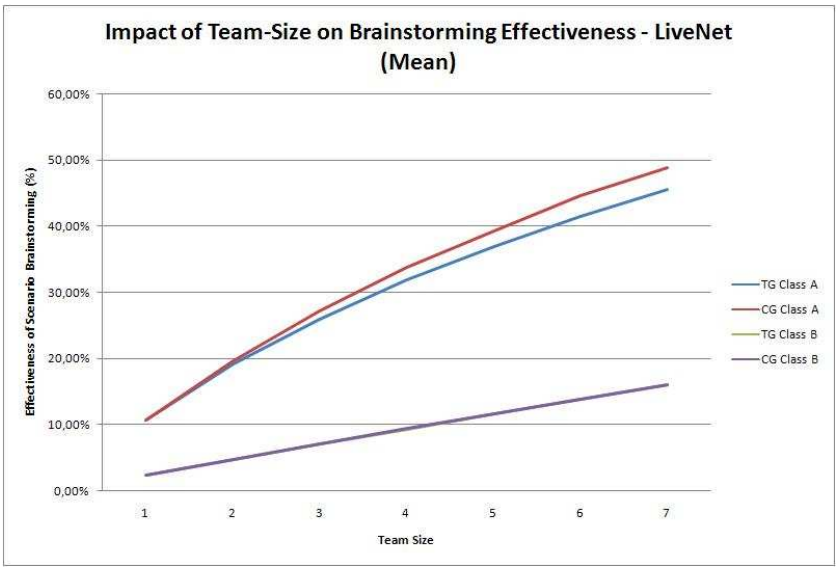


Figure 8.23.: Impact of team size on brainstorming effectiveness - LiveNet (mean)

Wiki

The big picture is more or less about the same for the Wiki-System. Again for class B scenarios, the mean values are very close to each other and the growth rate shows only very small decreases. But the standard deviations of the treatment group in case of class B scenarios are higher than those of the control group.

This is largely due to the varying group sizes. There were only 25 individuals in the treatment group entailing higher standard errors.

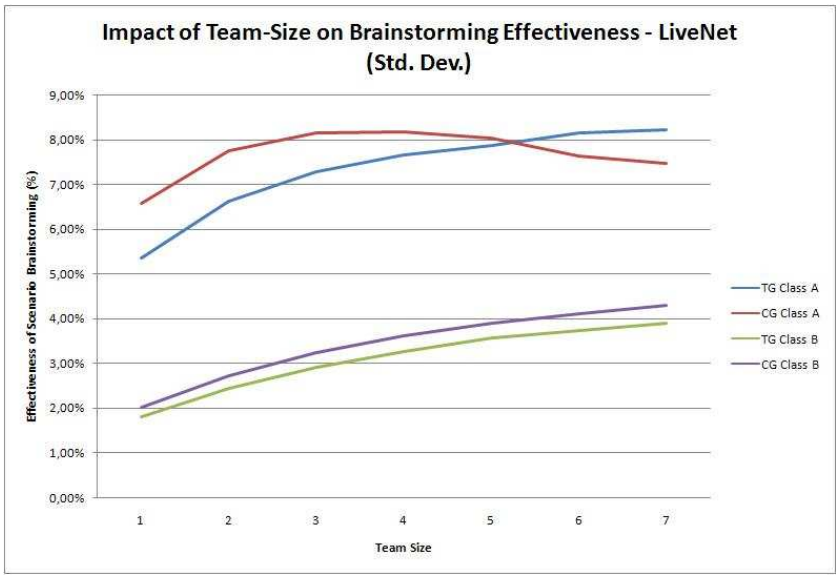


Figure 8.24.: Impact of team size on brainstorming effectiveness - LiveNet (standard deviation)

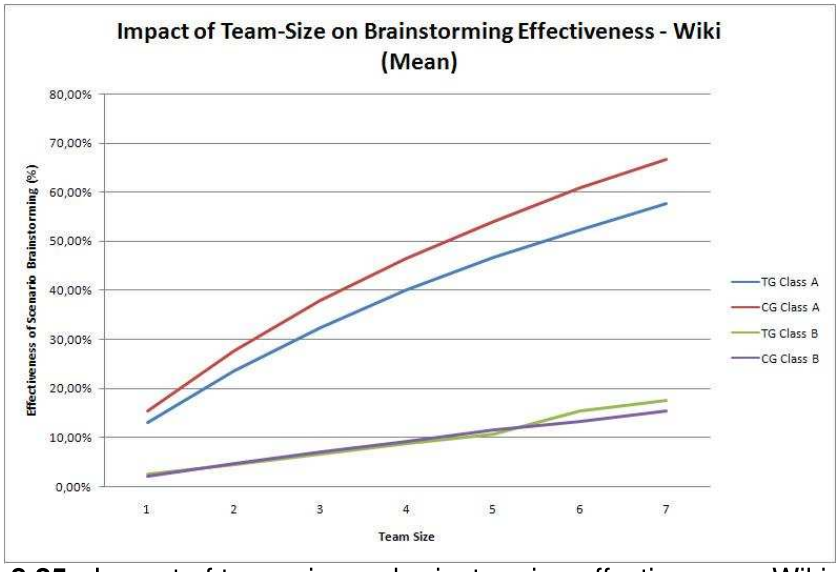


Figure 8.25.: Impact of team size on brainstorming effectiveness - Wiki (mean)

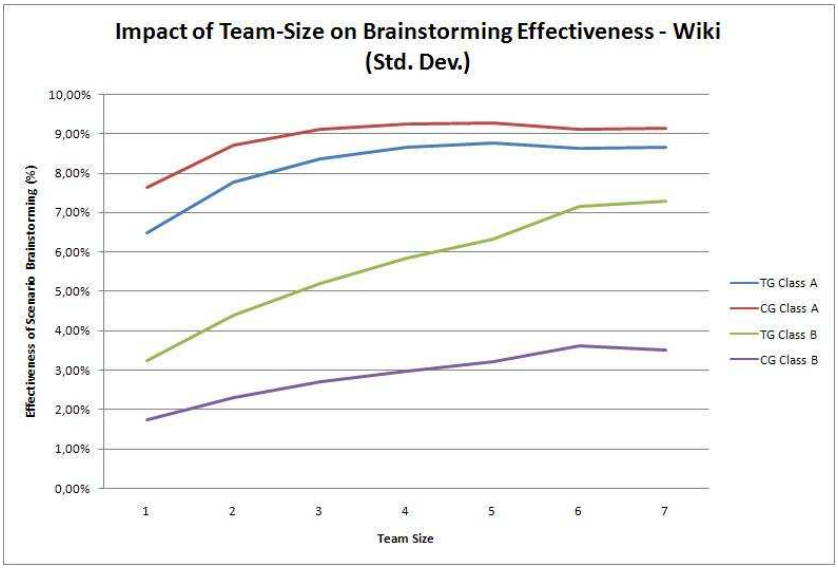


Figure 8.26.: Impact of team size on brainstorming effectiveness - Wiki (standard deviation)

Conclusion

According to the results, teams with 2 or 3 teammates seem to be most economic and effective supporting the findings of the original study. Furthermore, larger teams continue to make sense since the rate of effectiveness keeps increasing. The question is the economic value. How much costs are produced by a team meeting in relation to the scenarios found. This involves the problem of how many scenarios must be found. No general answer can be given as the decisions have to be made during the actual software project.

Comparison to the original study

The results affirm the statements of the original study. In fact both findings are very similar. Adding members to the team leads always to a gain in effectiveness. But on exceeding a team size of three, this gain is diminishing more faster.

9. Discussion and interpreting the results

So far all data sets, figures, calculations and values have been discussed. Well, what major findings can be concluded from them? In short, the major ones are that the support of (change) scenario categories is quite unimportant. Real teams are best to avoid less-important scenarios but the team members may be rivaling. Experts do not necessarily do a better job and the optimal team size seems to be three. In the follow up, these various points are considered in detail.

9.1. Individual performance

First, the focus is on the individual sessions. Analyzing the figures again, it seems that the support with (change) scenario categories did not have any influence on the resulting outcome. Furthermore, the system used has not helped or hindered the participants to create (high) quality scenarios. The rather large gap in terms of found scenarios between the original study and this replication is conspicuous. Unfortunately, no good explanation can be given, why fewer scenarios were created this time. The hypothesis *H1.1*, stating that the provision of software change categories will not improve the quality of scenario profiles developed for software architecture evaluation at an individual level, cannot be rejected.

9.2. Real team performance

Similar conclusions apply to the real teams. Those who were given the (change) scenario categories did not produce significantly different results. Additionally, the allocation to a certain system did not affect the quantity/quality of the output of the real teams. Unfortunately real teams loose scenarios compared to individuals working on their own.

However, preferring real teams to individual brainstorming has a big advantage. This advantage is also present when comparing real to nominal teams. They are optimal in eliminating or

avoiding less-important (also unimportant) scenarios. Again the hypothesis *H1.2*, stating the same as H1.1 but considering real teams, cannot be rejected.

9.3. Nominal team performance

Nominal teams cannot be evaluated just in isolation. They have to be compared with real teams. In their virtual environment the nominal teams performed better than the real teams. Contrarily to the individuals and the real teams, the support of (change) scenario categories and the allocation to one of the systems seem to have some impact on the performance overall. But, in terms of the LiveNet-System standalone, no impact of scenario categories was measurable.

But the optimal virtual solutions impressively showed the amount of discussion effort and/or rivalry between the real team members. Not only did the nominal teams produce more scenarios. Comparing the gains and losses of both, the real teams lost more Class A than class B scenarios. Not only did they lose in terms of quantity but also in terms of quality.

The hypothesis *H1.3* can be (carefully) rejected. Nominal teams who are given software change categories do perform slightly better. But the data and p-values state that also the system influences the results. Further research activities is necessary in this area.

9.4. Impact of reference profiles and experience

Using reference profiles shows similar results when comparing frequency based and expert based rankings. The individuals performed similarly regardless of the system or the study grouping. A group of IT-experienced individuals assesses the quality of scenarios as well as IT-experts.

At any rate, it seems that individuals with above-average experience tend to find more critical scenarios. But the margins of the figures are very slight. Nevertheless, also the effectiveness of well experienced individuals is slightly higher. Yet again, the support of (change) scenario categories and the allocation to a system do not exercise any significant influence.

Therefore the hypothesis *H2.1* can be rejected. Experience does improve the performance and the effectiveness.

9.5. Scenario and expert scoring

Expert and frequency related scoring/ranking can be viewed in a more economic way. A review of the deviations between the different scenario classes shows again that the frequency ranking is very similar as to the one of the experts. Therefore, especially when real teams in combination with individual sessions come into play, no external experts have to necessarily be used in an industrial environment. Of course the involved stakeholders must have certain skills, knowledge and education. It is assumed that the people in charge of software projects are carefully considering this.

Yet, one important fact should be pointed out. The experts rated already found scenarios afterwards. It would be an interesting comparison if the experts tried to find and rate scenarios themselves. Perhaps then the two scenario lists would be more different or not even comparable anymore. Nevertheless, the hypothesis *H2.2* cannot be rejected. The number of identified critical scenarios is similar for expert ranked and frequency related scores.

9.6. Team size effects

What would be the effect if the team size is smaller or larger? How do such changes influence effectiveness and is there an optimal team size? At the beginning, the curve depicting effectiveness rises strongly but increasingly levels off when adding more and more team members. As soon as the (team) size of a limited population increases, this is the mathematically logical consequence. If the nominal team size were at its maximum, i.e. all participating individuals, then the effectiveness would be 100 percent and the standard deviation zero. Similar findings are true for the standard deviation. But the curve does not level off evenly; it decreases more when a certain team size has been exceeded.

Similar to the original study, it can be stated, that the optimal economic team size is around 2 to 3 team members. If the team size increases further the effectiveness to be gained is shrinking. Of course, this statement is offered without taking into account any further variables like costs per additional team member and benefits of gained (critical) scenarios. Therefore the hypothesis *H3.1* can be rejected. If the team size is enlarged by additional team members the number of identified scenarios rises too.

9.7. Reasons for discrepancies of the results

On reviewing and comparing the results of both the original and the study replication, deviations have obviously occurred. Stating that one of them is correct is rather hazardous. In fact, it would be the right approach to explore why differences possibly happened and to question what factors might have influenced and affected the results.

Replications often involve the problem of compatibility with the original study and its conditions and environment including their constraints. Contrary results do not always entail rejection of established findings. Rather replications often get results under various conditions by exploring variables and factors which have a significant influence on the results and therefore also on the evaluated technique, procedure or others.

This chapter deals with possible problem areas pertaining to such factors and variables. Research in IT-science (see above in this work) and the evaluation of the experience and feedback questionnaires are used to find for possible hints.

9.7.1. Type of study/replication

As described in the chapter 5 “Replications” this study replication can more or less be accepted as a “literal” or “exact” replication. The work of (Lung et al.) states that in IT-science that kind of replication type has some disadvantages, especially because of the involvement of human subjects. Variability between the original study and replications as well as between replications themselves is inevitable. The variety of changed conditions and changes in the environment is large. It can be concluded that different results may not nullity those of the original study, as several factors could be affecting the replication. But nevertheless, they provide further insight about the topic which must be considered.

9.7.2. Study design

The fact that the treatment group in the first run of the experiment was acting as the control group in the second run was probably leading to learning effects. As the change categories, unlike within the original study, did not have any influence on the performances, these effects might have occurred. This would diminish the information value of this replication.

9.7.3. Timetable

Time is a fundamental aspect in relation to productivity as more results in less time using the same input often means less cost per defined unit. Unfortunately, a lack of this resource can result in a diminished output. Human beings perform negatively under stress so that both the quantity and quality of their work suffers.

In the section about study scheduling (see section 7.1.2 “Study schedule”) it was already pointed out that the individual brainstorming session had to be extended due to time problems. The individuals were asked in the feedback questionnaire whether they had enough time to complete their tasks or not. Possible answers were “zero” respectively yes, and “one” respectively no.

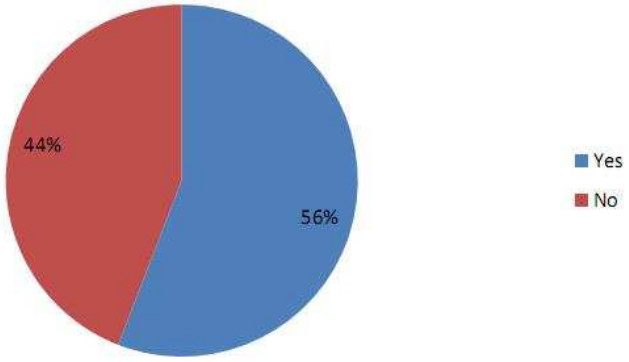


Figure 9.1.: Answer distribution to question: Did you have enough time for scenario brainstorming?

According to the figure above the majority stated having had time. But this is just slightly more than half of the participants. Even with a time extension of 15 minutes, 44 percent of the individuals still did not have enough time. Thus can be assumed that the quantity and/or the quality of the results of a large number of participants is below their normal standard.

9.7.4. Documentation

Corresponding to recent papers in IT-science (Shull et. al, Vegas et al), providing replicator with exhaustive documentations is a key issue for enabling sufficient replications. There should be included an exact description of the experiment, study processes, descriptions, results, interpretation, etc. in such a “lab package”.

The package about the original contained a lot of information, including design details, process guidelines, the used software systems, the original results, and more. Unfortunately, this doc-

umentation provided no explanation or commentary about the original findings. Furthermore, several analysis approaches were only mentioned in papers and had to be reconstructed.

Compared to recent IT-studies, the specific package of documentation was probably of good quality and contained substantial material. Nevertheless, important parts were lacking.

9.7.5. Collaboration

In conformity with the literature (Vegas), the cooperation between the researchers and the replicators can be described as “occasional”. During the design and planning phases of the replication, only limited coordination and teamwork took place. During the actual performance of the replication, some meetings and advice as well as support during the execution were given. Due to the required efforts (distance, time, workload ...) hardly any collaboration besides email or a similar communication channel would have been possible.

Despite these factors, there might have been more adjustments between the two studies possible. It is hard to say whether factors adulterating the results could have been avoided or eliminated.

9.7.6. Participants

This is probably the factor differing most widely to the original study. By contrast, during this experiment more less- experienced individuals were involved. On average, the individuals in the original study shared at least a similar educational level and had gained more experience from industrial work. Additionally they received two lectures of 2 hours, dedicated exactly to the experiment.[17, p.3] Furthermore, there were entry-barriers in the form of a minimum exam-success to join the research experiment. [17, p.3] It can therefore be conjectured that these participants may have been better prepared.

9.7.7. Sample size

Compared to the original study the number of participating individuals is almost double. A small sample size often leaves room for errors when compared to the whole or a larger subset of the population considered.[61] It is important to mention that not only the size but also the variety of the population should be reflected by the sample.

Due to the data set available, it cannot be established whether the sample size caused errors to happen and, if that was the case, which results were affected more. Nevertheless, this issue must be considered.

9.7.8. Cultural context

Due to the findings of (Lung), this involves several “problem areas”.[67] The most important factor is probably the aspect of the language. The original study was held in English and the whole study materials were written in this language as well. Since the location was in a country whose mother tongue was actually English, this was warranted and surely the most efficient way.

On average Austrian students do have (or should have) competitive skills in English, of course. It is safe to say that most of the individuals participating in the replication would have been more productive if they could have used their mother tongue. Furthermore, some of individuals did not have a German speaking background (e.g. family background) or were exchange students. Especially during the individual brainstorming sessions quite some problems arose. A few students really had problems understanding the study materials and recognizing what their tasks were. One major obstacle was the use of lots of rarely used expressions and specific vocabulary. Recalling the average self-estimate of the individuals, it showed that most of them did not have very good let alone excellent language skills [see section 7.3.3 “Language skills”].

Categorization of the language skills of the individuals revealed some gaps in knowledge. Together with the fact that many participants suffered under time stress it is reasonable to assume that these circumstances diminished the quantity and quality of the results. According to the questionnaires, 26 percent did not follow or only barely followed the guidelines. Only 17 percent executed the tasks strictly according to the instructions.

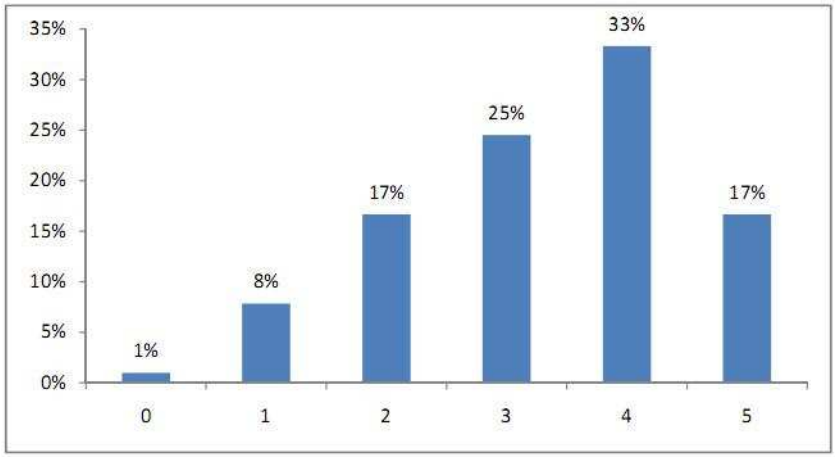


Figure 9.2.: Answer set to the question: Did you follow the instructions?

Moreover, about 25 percent marked the study instructions as not or barely helpful and only 13 percent found them very helpful. Unfortunately, almost none of the participants made any further comments. Therefore it can be assumed that, along with other factors, language skills were problematic and a decisive element.

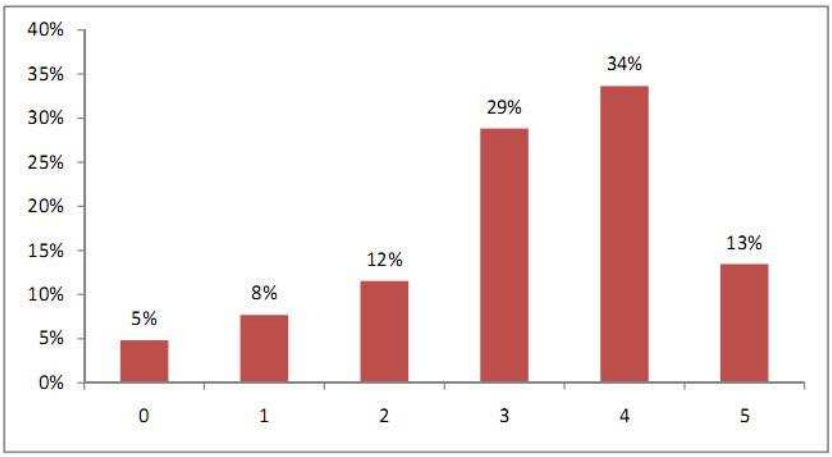


Figure 9.3.: Answer set to the question: Did find the instructions helpful to you?

9.7.9. Knowledge base

As already pointed out, the participants were ranked behind those of the original study particularly in terms of working experience. Accordingly, their knowledge of software architectures as well as of evaluation techniques might also be less distinctive. Taking these two factors into account, this has a possible impact on the performance of the individuals.

Another difference between the individual participants of the original experiment and the study replication is the educational and/or university background. Enrollment restrictions and (considerably) university fees in the Anglo-American states may have inducted a number of people to participate in the study, but also because they were interested either in the topic or they wanted to accumulate extra credit points. For sure, this is quite a vague theory but some minor influence may be given.

9.7.10. Summary

It cannot be measured, not even estimated if and to what extent all of the mentioned issues had an impact on the results. According to the sources in IT-science considered here, there is evidence that they do have at some small impact. Some of the relations and their consequences seem reasonable, of course. Yet, it cannot be proven that in this specific replication they led to any effects. It may be beneficial for further research work to pay more attention to specific

cultural issues. Whereby the shortage of resources barely allows to spend efforts and money to solve this matter. However, translating the study materials or at least the instructions for the tasks could have a positive impact on the performance of the participants, especially if they do not have a certain level of experience.

10. Conclusion

Replication work is essential in science.[21, 22] It is important that calculations and studies are repeated, results and findings are compared and that, in general, scientific research work is evaluated. Performing a replication to an already existing study is quite difficult. Many constraints and factors must be considered, often forcing the replicators to make adoptions. Therefore the comparability of the results is sometimes limited.

Unfortunately, some findings of this study replication question the knowledge gained by the original study. Again the data set was collected during an empiric experiment which was designed very similar to its predecessor (see chapter 8.1 “Experiment design”). But more participants joined the experiment and two separate systems were used. The researchers were put into the position to be able to perform two separate experiment runs. This improved the data set and made the information more valuable.

This thesis presents the results of a repeated empirical study. One major topic was to gain further knowledge about individual brainstorming and team meeting effects. Further it aimed at assessing the impact of the chosen approach for the scenario finding process of a software architecture evaluation (top-down and bottom-up). Either software categories were used to guide the individuals respectively the teams, or not. In order to get more insight into the topic, real teams were also compared to nominal counterparts. The results showed that teams lose scenarios, unfortunately also critical ones, compared to individuals. But they also sorted out all less-important scenarios.

The suggestion of the original study, to use an architecture evaluation process without a meeting in order to maximize effectiveness of scenario development, is still valid. Also the suggestion to introduce new approaches in order to keep the scenario gains while avoiding any losses - especially of high-quality scenarios keeps relevant.[20, 19] The findings regarding the usage of scenario categories are contrarily. No statistical impact of using scenario categories for scenario finding processes has been found. Therefore, the effort to define and chose such categories is questioned. Further research work is urgently required.

Due to the original study as well as to prior research work, the impact of (individual) experience on the results of design reviews was measured.[6, 10] The idea behind this research topic is still the same: The presence of wrong people during review sessions is a major problem with conventional designs of review approaches. But it is claimed to involve stakeholders actively. The original study postulated that they should have as much experience possible in

order to successfully conduct software architecture evaluation including the development of high-quality scenarios. The findings there supported this statement and were supported too by the ones of this replication.

Like in the original study, the scenario brainstorming effectiveness was measured to achieve comparability between the scenario classes. Again the rate of effectiveness is best in terms of critical scenarios regardless of the experience. But in contrast to the original study, the experience had some impact on the effectiveness. Thus it is suggested again for future empirical studies to consider participants with more variety of backgrounds (e.g., different architectural backgrounds, usage-oriented rather than technical backgrounds). Considering the different approaches and the individual experience the opposite of the original study was found. There is no statistical evidence that providing change categories helped or hindered individuals regardless of their experience. Probably, the number and the variety of the participants were too small. Further research work is required to answer that.

Another hypothesis made in the original study was that the experience of teams (expressed by the size of the real or nominal team) probably leads, based on individual brainstorming activities, to additional scenarios. The benefits of an increased team size as well as its cost are important factors in software architecture evaluating.[20] The findings of this replication support the statements of the original study. Also nominal teams were used for calculations this time. Larger teams found more scenarios and more critical scenarios. And the optimal team size in economic terms is around 3 team members. But for critical software projects it is probably advisable to apply evaluations with more team members in order to get a large rate of effectiveness. Considering larger teams it might be meaningful to have a diverse background of experience. Perhaps more and better scenarios can be found that way. But as mentioned in the original study, further research work is necessary to evaluate that recommendation.

This work showed the problems of replication work in science. Not infrequently, it is impossible to exactly repeat an empiric study. Therefore any adoptions or modifications must be carefully considered. Nevertheless, more data was gathered delivering more insight into the topic of scenario creating processes and scenario-based software architecture evaluating. Also, if not all findings were replicable, new and important questions have been made. The cultural context, learning effects and different systems can probably lead to different results. It seems meaningful not only to do more research but to concentrate, respectively isolate, on certain variables in order to check their influence in detail.

Future work

Besides or additionally the thoughts, conclusions and suggestions mentioned above, following pursuing research work might be worth to consider specifically. The impact of the evaluation approach (using predefined scenario categories or not) has to be tested more carefully. Both results - of the original study and the study replication - are contrary. But the design of the

replication probably caused learning effects which diminished the value of the findings.

Along with that the advantages or disadvantages of real teams compared to individuals have to be investigated more specifically. Teams find more scenarios but lose many and important scenarios. The question is whether team evaluation or individual brainstorming is to favor in terms of effectiveness and efficiency.

Finally the team size effectiveness has to be considered especially in terms of economic aspects. Basically the point of interest is how much a certain team size costs compared to its effectiveness and efficiency. This is important to know in order to convince industrials to adopt the research suggestions.

Bibliography

[1] Open System Lab Pervasive Technology Labs at Indiana University. Extreme programming, 2005. Available from: `http://www.osl.iu.edu/~lums/swc/www/xp.html` [cited 08.10.2009].

[2] Sommerville I. *Software Engineering*. Addison-Wesley, 2001.

[3] IBM. Rational unified process, 2009. Available from: `http://www.ibm.com/developerworks/rational/library/jun05/norlund/norlundfig2.gif` [cited 08.10.2009].

[4] Thomas M. Usecasemap, 2009. Available from: `http://www.softwarepractice.org/mediawiki/images/1/1b/M2UCM1.png` [cited 08.10.2009].

[5] Michael Fagan. Design and code inspections to reduce errors in program development. pages 575–607, 2002.

[6] Winkler D. Biffl S., Ali Babar M. Impact of experience and team size on the quality of scenarios for architecture evaluation. *Proceedings of the 12th international conference on evaluation and assessment in software engineering*, 2008. p. 1 - 10.

[7] Lars Lundberg, Jan Bosch, Daniel Hggander, and Per olof Bengtsson. Quality attributes in software architecture design. In *Proceedings of the IASTED 3rd International Conference on Software Engineering and Applications*, pages 353–362, 1999.

[8] Starke G. *Effektive Software-Architekturen: ein praktischer Leitfaden*. Hanser, München, 2008.

[9] Richard N. Taylor and Andre van der Hoek. Software design and architecture the once and future focus of software engineering. In *FOSE '07: 2007 Future of Software Engineering*, pages 226–243, Washington, DC, USA, 2007. IEEE Computer Society. `doi:http://dx.doi.org/10.1109/FOSE.2007.21`.

[10] Linda M. Northrop Paul C. Clements. Software architecture: An executive overview.

Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

[11] Department of Commerce's National Institute of Standards and Technology. Software errors cost U.S. economy $59.5 billion annually, 2002. Available from: http://www.nist.gov/public_affairs/releases/n02-10.htm [cited 08.10.2009].

[12] John C. Knight. Focusing software education on engineering. *SIGSOFT Softw. Eng. Notes*, 30(2):3–5, 2005. doi:http://doi.acm.org/10.1145/1050849.1050852.

[13] Hoang Pham and Xuemei Zhang. A software cost model with warranty and risk costs. *IEEE Trans. Comput.*, 48(1):71–75, 1999. doi:http://dx.doi.org/10.1109/12.743412.

[14] Lloyd G. Williams and Connie U. Smith. Pasa*SM*: a method for the performance assessment of software architectures. In *WOSP '02: Proceedings of the 3rd international workshop on Software and performance*, pages 179–189, New York, NY, USA, 2002. ACM. doi:http://doi.acm.org/10.1145/584369.584397.

[15] Rick Kazman, Len Bass, Gregory Abowd, and Mike Webb. Analyzing the properties of user interface software falsch? Technical report, Pittsburgh, PA, USA, 1993.

[16] Rick Kazman, S. Jeromy Carrière, and Steven G. Woods. Toward a discipline of scenario-based architectural engineering. *Ann. Softw. Eng.*, 9(1-4):5–33, 2000. doi:http://dx.doi.org/10.1023/A:1018964405965.

[17] Dietmar Winkler, Stefan Biffl, and Muhammad Ali Babar. Eliciting better quality architecture evaluation scenarios: A controlled experiment on top-down vs. bottom-up. In *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 348–350, New York, NY, USA, 2008. ACM. doi:http://doi.acm.org/10.1145/1414004.1414078.

[18] Gyuhyun Kwon, Dong Han Ham, and Wan Chul Yoon. Evaluation of software usability using scenarios organized by abstraction structure. In *ECCE '07: Proceedings of the 14th European conference on Cognitive ergonomics*, pages 19–22, New York, NY, USA, 2007. ACM. doi:http://doi.acm.org/10.1145/1362550.1362557.

[19] Dietmar Winkler, Stefan Biffl, and Muhammad Ali Babar. An empirical investigation of scenarios gained and lost in architecture evaluation meetings. In *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 348–350, New York, NY, USA, 2008. ACM. doi:http://doi.acm.org/10.1145/1414004.1414078.

[20] Dietmar Winkler, Stefan Biffl, and Muhammad Ali Babar. An empirical investigation of scenarios gained and lost in architecture evaluation meetings. In *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 348–350, New York, NY, USA, 2008. ACM. doi:http://doi.acm.org/10.1145/1414004.1414078.

[21] J. Brüderl. Replikation in der sozialwissenschaften, 2008. Available from: http://www.sowi.uni-mannheim.de/lehrstuehle/lesas/studium/hauptstudium.pdf.

[22] B. D. McCullough and H. D. Vinod. Verifying the solution from a nonlinear solver: A case study: Reply. *American Economic Review*, 94(1):391–396, March 2004. Available from: http://ideas.repec.org/a/aea/aecrev/v94y2004i1p391-396.html.

[23] Cat Kutay and Muhammad Ali Babar. Teaching three quality assurance techniques in tandem - lessons learned. *Quality Software, International Conference on*, 0:307–312, 2005. doi:http://doi.ieeecomputersociety.org/10.1109/QSIC.2005.62.

[24] Bass L. Clements P. Kazman R., Abowd G. Scenario-based analysis of software architecture. *Software, IEEE*, 1995. Volume: 13, Issue: 6 On page(s): 47-55 ISSN: 0740-7459.

[25] Klein M. Weinstock C. Barbacci M., Longstaff T. Quality attributes. Technical report, Carnegie Mellon University, Pennsylvania, USA, 1995.

[26] PCMag-The independent guide to technology. Definition of: software architectur, 2008. Available from: http://www.pcmag.com/encyclopedia_term/0,2542,t=software+architecture%&i=51662,00.asp [cited 06.01.2009].

[27] Mayr H. *Projekt Engineering*. Carl Hanser Verlag, Leipzig, 2001.

[28] Tortora G. Ambriola V. *Advances in Software Engineering and Knowledge Engineering*. World Scientific Pub Co, London, 1993.

[29] Kazman R. Bass L., Clements P. *Software Architecture in Practice*. Addison-Wesley, Amsterdam, 2003.

[30] Soni D. Hofmeister C., Nord R. *Applied Software Architecture: A Practical Guide for Software Designers*. Addison-Wesley, Amsterdam, 1999.

[31] Dumke R. *Eine Einführung für Informatiker und Ingenieure: Systeme Erfahrungen, Methoden, Tools*. Vieweg & Sohn, Wiesbaden, 2001.

[32] Koreimann D. *Grundlagen dr Software-Entwicklung*. Oldenburg Verlag, München, 2000.

[33] PerOlof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture-level modifiability analysis (alma). *J. Syst. Softw.*, 69(1-2):129–147, 2004. doi:http://dx.doi.org/10.1016/S0164-1212(03)00080-3.

[34] Dewayne E. Perry and Alexander L. Wolf. Foundations for the study of software architecture. *SIGSOFT Softw. Eng. Notes*, 17(4):40–52, 1992. doi:http://doi.acm.org/10.1145/141874.141884.

[35] Keller W. Software engineering für große informationssysteme. Vorlesungsunterlagen. Available from: http://www.schmidp.com/public/segbis07/(01a)Organisatorische%20Einfuehrung.pdf.

[36] IEEE. Ieee standard 1471-2000, 2007. Available from: http://ieeexplore.ieee.org/ISOL/standardstoc.jsp?punumber=4278470.

[37] Van Vliet H. Bosch J. Lassing N., Bengtsson P. Experiences with alma: Architecture-level modifability analysis. *The Journals of Systems and Software Volume 61*, 2002.

[38] Mussbacher G. Amyot D., Mansurov N. Understanding existing software with use case map scenarios. In *Telecommunications and beyond: The Broader Applicability of SDL and MSC*, pages 124–140, 2007.

[39] Mussbacher G. Amyot D. Introduction to use case maps, 2001. Available from: www.itu.int/itudoc/itu-t/com17/urn/urnp5_pp7.ppt [cited 08.10.2009].

[40] Amyot D. UCM quick tutorial version 1.0, 1999. Available from: http://jucmnav.softwareengineering.ca [cited 08.10.2009].

[41] Thaller G. *Software-Qualität*. VDE Verlag, Berlin, 2000.

[42] Biffl S. Qualitätssicherung. Script, 2004. Available from: qse.ifs.tuwien.ac.at/courses/skriptum/.../05P_QS_WID_20040204.pdf.

[43] CEFE-CAD/CAM Entwicklungsgesellschaft Arbeitsgruppe 19 Wendt D. Klassische fehler in der software-entwicklung. Firmeninterner Artikel, 1995.

[44] Bosch J. Bengtsson P. An experiment on creating scenario profiles for software change. *Annals of Software Engineering*, 2000. pp. 59-78;.

[45] Obbink H. Ionita M, Hammer D. Scenario-based software architecture evaluation

methods: An overview. Available from: http://en.wikipedia.org/wiki/Scenario [cited 15.01.2009].

[46] Van Vliet H. De Bruin H. *Scenario-Based Generation and Evaluation of Software Architectures*, pages 128–139. Springer Berlin/Heidelberg, 2008.

[47] Business Dictionary. Definition of scenario, 2009. Available from: http://www.businessdictionary.com/definition/scenario.html [cited 28.07.2009].

[48] Klein M. Carrière J. Kazman R., Barbacci M. Experience with performing architecture tradeoff analysis. In *Proceedings of the International Conference on Software Engineering, New York: ACM Press*, pages 54–63, 1999.

[49] D. L. Parnas and D. M. Weiss. Active design reviews: principles and practices. *J. Syst. Softw.*, 7(4):259–265, 1987. doi:http://dx.doi.org/10.1016/0164-1212(87)90025-2.

[50] Kazman R. Klein M. Clements, P. *Evaluating software architectures: methods and case studies*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[51] Dietmar Winkler, Michael Halling, and Stefan Biffl. Investigating the effect of expert ranking of use cases for design inspection. In *EUROMICRO '04: Proceedings of the 30th EUROMICRO Conference*, pages 362–371, Washington, DC, USA, 2004. IEEE Computer Society. doi:http://dx.doi.org/10.1109/EUROMICRO.2004.49.

[52] Parag C. Pendharkar and James A. Rodger. The relationship between software development team size and software development cost. *Commun. ACM*, 52(1):141–144, 2009. doi:http://doi.acm.org/10.1145/1435417.1435449.

[53] Adam Porter, Harvey Siy, Audris Mockus, and Lawrence Votta. Understanding the sources of variation in software inspections. *ACM Trans. Softw. Eng. Methodol.*, 7(1):41–79, 1998. doi:http://doi.acm.org/10.1145/268411.268421.

[54] Lawrence G. Votta, Jr. Does every inspection need a meeting? In *SIGSOFT '93: Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering*, pages 107–114, New York, NY, USA, 1993. ACM. doi:http://doi.acm.org/10.1145/256428.167070.

[55] Lesley Pek Wee Land, Chris Sauer, and Ross Jeffery. Validating the defect detection performance advantage of group designs for software reviews: report of a laboratory experiment using program code. In *ESEC '97/FSE-5: Proceedings of the 6th European*

SOFTWARE ENGINEERING conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering, pages 294–309, New York, NY, USA, 1997. Springer-Verlag New York, Inc. doi:http://doi.acm.org/10.1145/267895.267917.

[56] Chris Sauer, D. Ross Jeffery, Lesley Land, and Philip Yetton. The effectiveness of software development technical reviews: A behaviorally motivated program of research. *IEEE Transactions on Software Engineering*, 26(1):1–14, 2000. doi:http://doi.ieeecomputersociety.org/10.1109/32.825763.

[57] Adam A. Porter and Philip M. Johnson. Assessing software review meetings: Results of a comparative analysis of two experimental studies. *IEEE Trans. Softw. Eng.*, 23(3):129–145, 1997. doi:http://dx.doi.org/10.1109/32.585501.

[58] Steven D. Schafersman. An introduction to science, 1994. Available from: http://www.freeinquiry.com/intro-to-sci.html.

[59] Hamermesh D. Replication in economics. Technical report, National bureau of economic research, 2007. Available from: http://www.nber.org/papers/w13026.pdf?new_window=1.

[60] Forrest J. Shull, Jeffrey C. Carver, Sira Vegas, and Natalia Juristo. The role of replications in empirical software engineering. *Empirical Softw. Engg.*, 13(2):211–218, 2008. doi:http://dx.doi.org/10.1007/s10664-008-9060-1.

[61] Victor Basili. The role of controlled experiments in software engineering research. In *Empirical Software Engineering Issues. Critical Assessment and Future Directions*, pages 33–37, Berlin, Heidelberg, Germany, 2007. Springer.

[62] Hussy W. Einzelfallforschung und zeitreihenanalyse. Script, 1999. Available from: www.uni-koeln.de/phil-fak/fs-psych/serv_pro/skripte/evalua/Grundlagen.rtf.

[63] F. Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, pages 246–263, 1886.

[64] Michael J. Barany. Science's language problem, 2005. Available from: http://www.businessweek.com/technology/content/mar2005/tc20050317_4179.htm.

[65] Carmen Zannier, Grigori Melnik, and Frank Maurer. On the success of empirical studies in the international conference on software engineering. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 341–350, New

York, NY, USA, 2006. ACM. doi:http://doi.acm.org/10.1145/1134285.1134333.

[66] Sira Vegas, Natalia Juristo, Ana Moreno, Martín Solari, and Patricio Letelier. Analysis of the influence of communication between researchers on experiment replication. In *ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pages 28–37, New York, NY, USA, 2006. ACM. doi:http://doi.acm.org/10.1145/1159733.1159741.

[67] Jonathan Lung, Jorge Aranda, Steve M. Easterbrook, and Gregory V. Wilson. On the difficulty of replicating human subjects studies in software engineering. In *ICSE '08: Proceedings of the 30th international conference on Software engineering*, pages 191–200, New York, NY, USA, 2008. ACM. doi:http://doi.acm.org/10.1145/1368088.1368115.

[68] Robert P. Biuk-Aghai and Igor T. Hawryszkiewycz. Analysis of virtual workspaces. In *DANTE '99: Proceedings of the 1999 International Symposium on Database Applications in Non-Traditional Environments*, page 325, Washington, DC, USA, 1999. IEEE Computer Society.

[69] Brewer Marilynn. Research design and issues of validity. *Handbook of Research Methods in Social and Personality Psychology*, 2000.

[70] Martin Höst, Björn Regnell, and Claes Wohlin. Using students as subjects—a comparative study ofstudents and professionals in lead-time impact assessment. *Empirical Softw. Engg.*, 5(3):201–214, 2000. doi:http://dx.doi.org/10.1023/A:1026586415054.

[71] Helic D. Software architecure - quality attributes, 2008. Available from: http://coronet.iicm.tugraz.at/sa/s5/sa_qa.html [cited 01.01.2009].

[72] International Organization for Standadization. Iso9126, 2008. Available from: http://en.wikipedia.org/wiki/ISO_9126 [cited 01.01.2009].

[73] SoftwareArchitectures. Discipline » designing architecture » quality attributes, 2009. Available from: http://www.softwarearchitectures.com/go/Discipline/DesigningArchitecture/QualityAttributes/tabid/64/Default.aspx [cited 01.01.2009].

[74] Uwe Z. Software-architektur für große informationssysteme, 2005. Available from: http://wi.wu-wien.ac.at/home/uzdun/teaching/oo/05-swarch.pdf [cited 06.01.2009].

[75] Ann M. Hickey, Douglas L. Dean, and Jr. Jay F. Nunamaker. Establishing a foundation

for collaborative scenario elicitation. *SIGMIS Database*, 30(3-4):92–110, 1999. doi:http://doi.acm.org/10.1145/344241.344247.

[76] Emmanuel Letier, Jeff Kramer, Jeff Magee, and Sebastian Uchitel. Monitoring and control in scenario-based requirements analysis. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 382–391, New York, NY, USA, 2005. ACM. doi:http://doi.acm.org/10.1145/1062455.1062527.

[77] Christian Seybold, Silvio Meier, and Martin Glinz. Scenario-driven modeling and validation of requirements models. In *SCESM '06: Proceedings of the 2006 international workshop on Scenarios and state machines: models, algorithms, and tools*, pages 83–89, New York, NY, USA, 2006. ACM. doi:http://doi.acm.org/10.1145/1138953.1138969.

[78] Neil Maiden and Suzanne Robertson. Developing use cases and scenarios in the requirements process. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 561–570, New York, NY, USA, 2005. ACM. doi:http://doi.acm.org/10.1145/1062455.1062555.

[79] Hussy W. Replikationen. Script, 2005. Available from: www.psych-methoden.uni-koeln.de/veranstaltungen/evaluation/einzelfall/Einzelfall-09-Replikation-2005.pdf.

[80] Rick Kazman, Len Bass, Mark Klein, Tony Lattanze, and Linda Northrop. A basis for analyzing software architecture analysis methods. *Software Quality Control*, 13(4):329–355, 2005. doi:http://dx.doi.org/10.1007/s11219-005-4250-1.

A. Appendix

Experience Questionaire		
Student-ID		
Name		
LVA	☐ QS-VU	☐ Testen
Session	☐ Tuesday, June 3rd	☐ Friday, June 6th

Please answer the questions in this questionnaire.

Note, that your answers will be treated anonymously. Your ID and name will be used to link your answers to your scenario reports. We will remove all personal data gathered for evaluation purposes afterwards.

1. Demographic Data

Age		Gender	

2. How do you rate your ability to understand English documents?

no experience						excellent
	0	1	2	3	4	

3. How many programming courses have you attended?

4. Estimate your general software engineering experience.

no experience						excellent
	0	1	2	3	4	

5. Estimate your project management experience.

no experience						excellent
	0	1	2	3	4	

6. Estimate your experience in software development (implementation).

no experience						excellent
	0	1	2	3	4	

7. Estimate your experience in developing software in a professional environment (e.g., working in a company).

no experience						Excellent
	0	1	2	3	4	

8. Estimate the time, you are participating in professional software engineering projects (in months / years).

9. Estimate the size of the largest project, you participated in (in person months/years)

10. Estimate your experience in quality assurance.

no experience						excellent
	0	1	2	3	4	

11. Estimate your experience in quality assurance in a professional environment (e.g., working in a company).

no experience						excellent
	0	1	2	3	4	

12. Estimate your experience with requirements and design documents.

no experience						excellent
	0	1	2	3	4	

13. Estimate your experience with Use Cases.

no experience						excellent
	0	1	2	3	4	

14. Estimate your experience in Software Design review / inspection?

no experience						excellent
	0	1	2	3	4	

15. Estimate your experience with architecture reviews.

no experience						excellent
	0	1	2	3	4	

16. Estimate your experience in developing of scenarios.

no experience						excellent
	0	1	2	3	4	

17. Estimate your experience in using collaborative tools.

no experience						excellent
	0	1	2	3	4	

18. Estimate your experience in chatting (skype, ICQ, etc.)

no experience						excellent
	0	1	2	3	4	

Please use the backside of this page for additional comments.

Thank you for your contribution!

Software Requirements Specifications for a collaborative tool to support distributed software architecture evaluation process

A collaborative application to support the distributed software architecture (SA) evaluation process is required. This web-based application will support synchronous (same-time, different places) and asynchronous (different time- different places) activities for SA process. This collaborative application shall provide an intuitive and user friendly environment to perform various activities (i.e., planning, scenario development and prioritisation, identifying architectural approaches, documenting and annotating design patterns, viewing architectural design, evaluating architectural design with respect to scenarios, interpreting results etc.) of SA evaluation process.

A member of SA evaluation team accesses the application using a web browser. Once successfully connected to the application, a team member can perform a number of tasks according to his/her role in the process. For example, evaluation team leader can plan the evaluation session by identifying various participants and scheduling their diaries. He/she can also perform a number of activities to make sure all the required documents and SA descriptions and required views will be available before the evaluation starts. The evaluation participants may use the tool to access and study the artefacts assigned to him/her. This application allows a user to add, remove, edit, or copy various artefacts during the process. A user can use a web browser at his/her desktop computer to view and annotate an artefact assigned to him/her. All annotation is viewable by all participants. Participants may use various architectural views to document and annotate architectural design decisions. Software architect may access different patterns to show their use in the architecture. The evaluation team may use various evaluation techniques, e.g. checklist-based, scenarios-based, questionnaire, prototyping, simulations etc. during the evaluation sessions. It is quite useful if the technique to be used in the process and relevant support material are available online. It is also quite handy for evaluation manager to find information about the level of expertise, skills, and availability of various members of the organization who can be assigned the role of an evaluator.

A web-based tool to support generic collaborative activities has been developed. This collaborative tool, LiveNet, has been implemented with the following features:

- A user can register with the system. Once registered, a user is assigned a portal to perform a number of collaborative and knowledge management activities. For example, a user can create a number of workspaces, one for each unique activity that needs to be performed, a number of roles can be created and these roles can be assigned to various activities and individual participants of those activities can be placed in those groups for ease of privilege and security management.
- Once registered and assigned to a particular portal or workspace, a user can use a web browser to access and view an online version of the documents prepared and uploaded to the workspaces.
- All the users have full editing privileges to any artefact assigned to the user. A user needs to select the desired artefact and click "edit" link to make changes to that artefact.
- A user can provide a brief description of the artefact created in a workspace. The system makes this brief description available to all the users who can access that particular artefact.
- If this tool is used to support a particular activity of software development process, a large number of documents can be made available to the members of the team by uploading those

documents. For example, design diagrams, checklists to ensure the quality of the process, an online version of any standards that must be complied with.

- A user can send text message to other members of the team on various issues of importance. A user can also use discussion forum to discuss various issues and discussion forums can serve like organisational memory to be accumulated over the life time of the project.
- A user can assign various artefacts to different team members.
- The system provides an online chat room that can be used for synchronous meetings, while discussion forum can be used for asynchronous meetings.
- A user can send an email to other members involved in the same activity.
- A user can invite new members to join a group or activity.

Now suppose that your organisation plans to use LiveNet regularly to support a distributed software architecture evaluation process. To successfully support a distributed SA evaluation process, LiveNet needs to provide a number of features to help various roles (such as evaluation team leader, evaluation manager, evaluators, participants etc.) to perform various tasks. Try to think of the changes that would be required in LiveNet in its current as well as next three years life. Specify those required changes in terms of the change scenarios to characterise **modifiability** (using the skill you learned during training sessions).

Software Requirements Specifications for a collaborative tool to support distributed software architecture evaluation process

Change Categories

Change categories are expected to address possible occurring changes over the next period, e.g., over the next three years. The system architecture and scenarios must consider these issues. Based on business analysis, we identified 6 basic change categories.

Try to come up with at least one scenario for each of the following categories and refer to them in the data capturing sheet.

1. User interface changes (UI)
2. Security policy changes (SP)
3. Performance changes (PC)
4. Communication channels and/or mechanism changes (CM)
5. Workflow features changes (WF)
6. Content management requirements changes (CM)

Software Requirements Specifications for Wiki System

Wiki is a web-based collaborative content management system (CMS). It provides an intuitive and user friendly environment to create and organize various contents.

A user can access the Wiki system using a web browser. Once within the Wiki system environment, a user can create new pages and edit existing pages on the fly. All the pages can be linked between each other by using hyperlinks under a naming convention. This means a user can transverse between different pages by just clicking on the appropriate link. Wiki system uses very simple and common text formatting rules to simplify the process of writing new pages and linking them to each other. The idea behind the simple text formatting rules is to keep text editing task quite easy and intuitive.

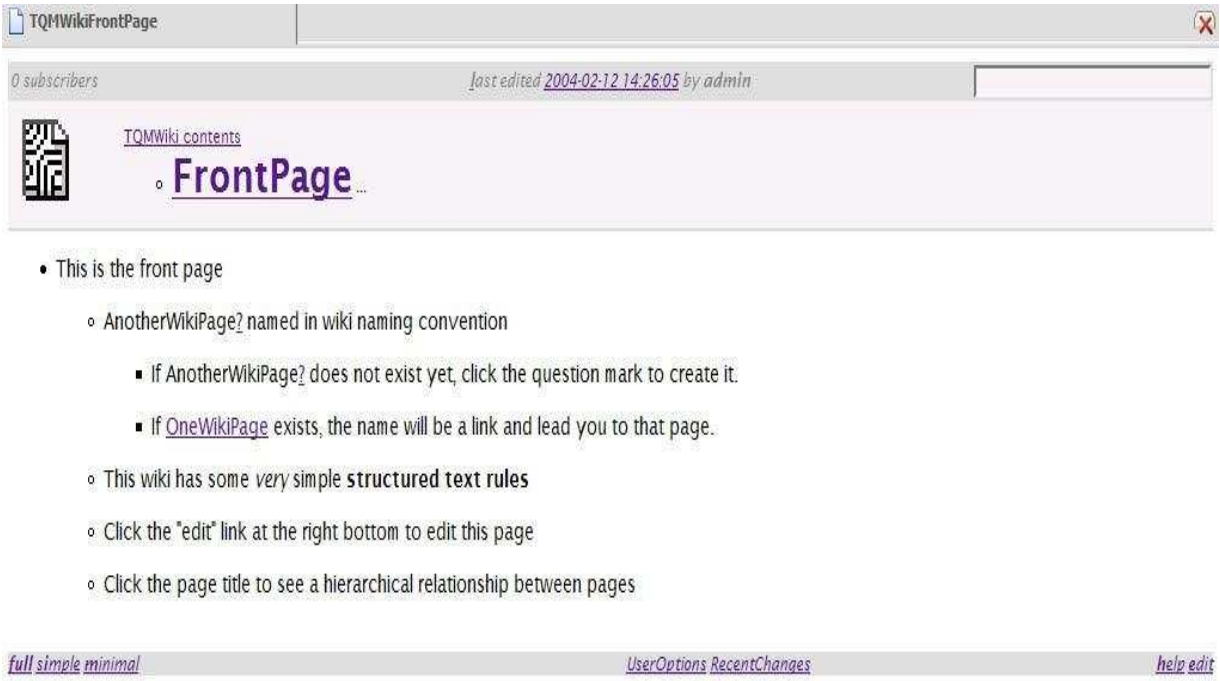
The Wiki system is implemented on an open source framework, which provides some common content management system services. A simple working prototype has been developed with the following features:

- All the users have full editing privileges to any page they want to edit. A user can click "edit" link on the desired page to change the text on that page using an editing form.
- Wiki system follows a naming convention for a page name. The naming convention is: capitalized words joined together. For example: APageName, SoftwareEngineering, IssuesOnDesignQuality etc.
- Wiki system formats and links the text content according to the simple structured text formatting rules.
- When the Wiki system finds a text content following its naming convention, it makes that text a hyperlink for a target page. If the target page does not exist, a question mark is placed at the end of the hyperlink to remind the user that this page needs to be created.
- Wiki system keeps the log of changes made in a page. This log can be displayed in unix diff format.
- Users can subscribe to the page change through email.
- Wiki system allows the users to make certain changes in the environment through option setting page. E.g. name, email, time zone etc.

Now imagine that you are a regular user of the Wiki system for managing and sharing your content using the web-based system. Try to imagine the changes you would like to see in this system over the next three years. Please document those possible changes in terms of the change scenarios (using the skill you learned during training sessions).
However, please do **NOT** propose large functionality changes which are not in align with the basic principles of the system.

Screenshots of the Wiki-System

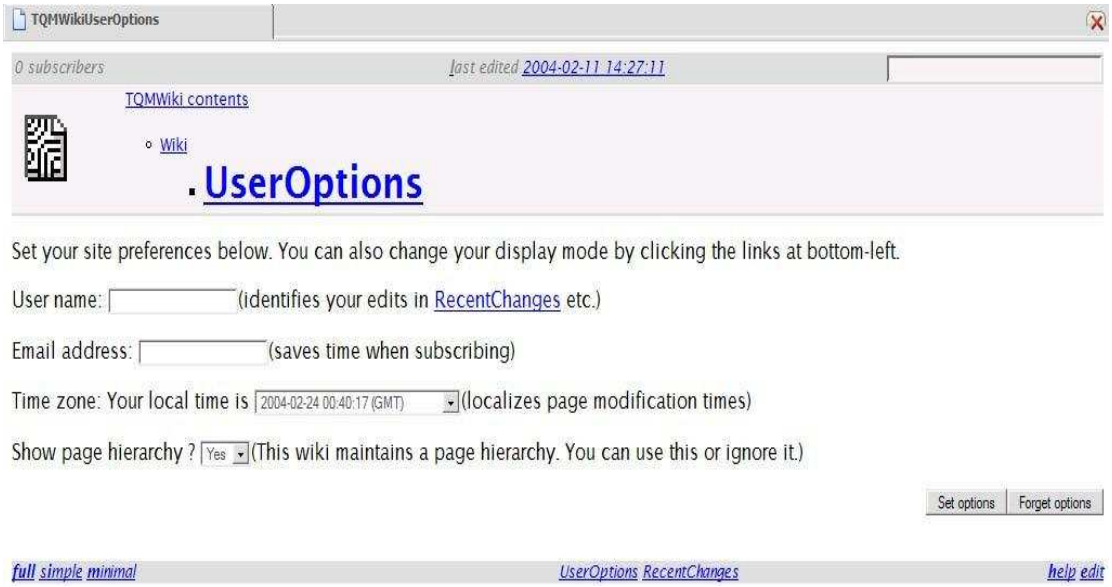
1. Front Page



2. Editing Form

3. Change History

4. User Option



Software Requirements Specifications for Wiki System
Change Categories

Change categories are expected to address possible occurring changes over the next period, e.g., over the next three years. The system architecture and scenarios must consider these issues. Based on business analysis, we identified 6 basic change categories.

Try to come up with at least one scenario for each of the following categories and refer to them in the data capturing sheet.

1. User interface changes (UI)
2. Security policy changes (SP)
3. Performance changes (PC)
4. Notification policy changes (NP)
5. Content editing rules changes (CE)
6. Meta data related changes (MD)

Individual Scenario Brainstorming

Page: 1

Identification	Course Assignment	Session Assignment	Application	Scenario Categories*:
Student-ID: _____	☐ QS-VU	☐ Tuesday, June 3rd	☐ LiveNet	☐ Used
Name: _____	☐ Testen	☐ Friday, June 6th	☐ Wiki	☐ Not Used

Notes:

- **No**: Number = Ongoing number
- **Scenario description**: Description of the identified scenario (keep it short)
- **Importance**: Estimate the importance of the scenario: A…critical (very important), B … important, C less important.
- **Likelihood**: Estimate the likelihood of the scenario to occur: A … most likely, B…likely, C .. unlikely.
- **Scenario Categories***:
  - **Categories used**: write down the category this scenario belongs too
  - **Categories NOT used**: write down in which category this scenario could fit into.

Start time: _____

End Time: _____

Duration: _____

Pages: _____

No.	Scenario-Description	Importance	Likelihood	Scenario Categories*

Individual Scenario Brainstorming	Student-ID:	Name:	Page:

No.	Scenario-Description	Importance	Likelihood	Scenario Categories*

Team Meeting								**Page: 1**
Team Member 1	Student-ID:		Name:		Role		☐ QS-VU	☐ Testen
Team Member 2	Student-ID:		Name:		Role:		☐ QS-VU	☐ Testen
Team Member 3	Student-ID:		Name:		Role		☐ QS-VU	☐ Testen

Session Assignment	**Application**	**Scenario Categories*:**	**Meeting Style**
☐ Tuesday, June 3rd ☐ Friday, June 6th	☐ LiveNet ☐ Wiki	☐ Used ☐ Not Used	☐ Toolsupport ☐ F2F

Notes:

- **No**: Number = Ongoing number
- **Scenario description**: Description of the identified scenario (keep it short)
- **Importance**: Estimate the importance of the scenario: A…critical (very important), B … important, C less important.
- **Likelihood**: Estimate the likelihood of the scenario to occur: A … most likely, B…likely, C .. unlikely.
- **Scenario Categories***:
  - **Categories used**: write down the category this scenario belongs too
  - **Categories NOT used**: write down in which category this scenario could fit into.

Start time: _____

End Time: _____

Duration: _____

Pages: _____

No.	Scenario-Description	Importance	Likelihood	Scenario Categories*
1				

Team Meeting			Page:
Student ID (1):	**Student ID (2):**	**Student ID (3):**	

No.	Scenario-Description	Importance	Likelihood	Scenario Categories*

Individual Feedback Questionnaire		
Student-ID		
Name		
LVA	☐ QS-VU	☐ Testen
Session	☐ Tuesday, June 3rd	☐ Friday, June 6th
Application	☐ LiveNet	☐ Wiki
Scenario Categories	☐ Used	☐ Not used

Please answer the questions in this questionnaire,

Note, that your answers will be treated anonymously. Your ID and name will be used to link your answers to your scenario reports. We will remove all personal data gathered for evaluation purposes afterwards.

1. Did you have enough time for scenario brainstorming?

   Yes ☐ (0) No ☐ (1)

2. If not, how much additional time would you need (in minutes)?

3. Did you follow the instructions (guidelines) of the applied method during the execution?

Never							All the time
	0	1	2	3	4	5	

4. Have the instructions been helpful to you? If not, please provide suggestions for improvement on the backside of this sheet.

No							Yes
	0	1	2	3	4	5	

5. Did you find the method easy to be applied?

Simple							Difficult
	0	1	2	3	4	5	

6. Do you think that the scenario brainstorming method is useful?

Not useful							Very useful
	0	1	2	3	4	5	

7. How useful is the applied method to identify scenarios?

Not useful							Very useful
	0	1	2	3	4	5	

8. Would you use the method for scenario brainstorming in the future?

No							Yes
	0	1	2	3	4	5	

9. Overall number of scenarios

8a. How many scenarios did you identify?

8b. Estimate the percentage [%] of identified scenarios.

0-10%	10-20%	20-35%	35-50%	50-75%	> 75%

8c. How confident are you with your estimation?

Unsure							Very sure
	0	1	2	3	4	5	

9. Number of critical scenarios

9a. How many critical scenarios did you identify?

9b. Estimate the percentage [%] of identified critical scenarios.

0-10%	10-20%	20-35%	35-50%	50-75%	> 75%

9c. How confident are you with your estimation?

Unsure							Very sure
	0	1	2	3	4	5	

10. Only, if you applied scenario categories

10a. Did you find the scenario categories helpful for scenarios brainstorming?

Not at all							Definitely, yes
	0	1	2	3	4	5	

10b. Estimate the number of scenarios, you would have found without scenario categories.

0-10%	10-20%	20-35%	35-50%	50-75%	> 75%

10c. Estimate the number of critical scenarios, you would have found without scenario categories.

0-10%	10-20%	20-35%	35-50%	50-75%	> 75%

Please use the backside of this page for additional comments.

Thank you for your contribution!

Team Feedback Questionnaire (F2F)		
Student-ID		
Name		
Team-ID		
LVA	☐ QS-VU	☐ Testen
Session	☐ Tuesday, June 3rd	☐ Friday, June 6th
Application	☐ LiveNet	☐ Wiki
Scenario Categories	☐ Used	☐ Not used

Please answer the questions in this questionnaire.

Note, that your answers will be treated anonymously. Your ID and name will be used to link your answers to your scenario reports. We will remove all personal data gathered for evaluation purposes afterwards.

1. How many team-members did you know prior to this meeting?

0	1	2

2. Did you have enough time for the team meeting?

Yes ☐ (0) No ☐ (1)

3. If not, how much additional time would you need (in minutes)?

4. Did you follow the instructions (guidelines) of the applied method during the execution?

Never						All the time
	0	1	2	3	4	

5. Have the instructions been helpful to you? If not, please provide suggestions for improvement on the backside of this sheet.

No						Yes
	0	1	2	3	4	

6. Do you think that a team meeting is helpful for scenario elicitation?

No						Yes
	0	1	2	3	4	

7. How satisfied are you with the team meeting in general?

Unsatisfied						Satisfied
	0	1	2	3	4	

8. How satisfied are you with the discussion in today's team meeting?

Unsatisfied						Satisfied
	0	1	2	3	4	

9. How satisfied are you with the discussion outcome of today's team meeting?

Unsatisfied						Satisfied
	0	1	2	3	4	

10. How effective was the communication during the meeting?

Not effective						Very effective
	0	1	2	3	4	

11. Please rate the communication process you used for generating scenarios in your team:

	0	1	2	3	4	
Impersonal						Personal
Bad						Good
Difficult						Easy
Informal						Formal
Simple						Complex

12. Please describe the **conversation** you had with your team members for discussing scenarios:

0 .. Strongly disagree 4 .. Strongly Agree	0	1	2	3	4
Pleasant					
Agreeable					
Interesting					
Argumentative					
Cooperative					

Please use this page for additional comments on the Face-To-Face team meeting

Thank you for your contribution!

Team Feedback Questionnaire (Tool)		
Student-ID		
Name		
Team-ID		
LVA	☐ QS-VU	☐ Testen
Session	☐ Tuesday, June 3rd	☐ Friday, June 6th
Application	☐ LiveNet	☐ Wiki
Scenario Categories	☐ Used	☐ Not used

Please answer the questions in this questionnaire.

Note, that your answers will be treated anonymously. Your ID and name will be used to link your answers to your scenario reports. We will remove all personal data gathered for evaluation purposes afterwards.

1. How many team-members did you know prior to this meeting?

0	1	2

2. Did you have enough time for the team meeting?

Yes ☐ (0) No ☐ (1)

3. If not, how much additional time would you need (in minutes)?

4. Did you follow the instructions (guidelines) of the applied method during the execution?

Never						All the time
	0	1	2	3	4	

5. Have the instructions been helpful to you? If not, please provide suggestions for improvement on the backside of this sheet.

No						Yes
	0	1	2	3	4	

6. Do you think that a team meeting is helpful for scenario elicitation?

No						Yes
	0	1	2	3	4	

7. How satisfied are you with the team meeting in general?

Unsatisfied						Satisfied
	0	1	2	3	4	

8. How satisfied are you with the discussion in today's team meeting?

Unsatisfied						Satisfied
	0	1	2	3	4	

9. How satisfied are you with the discussion outcome of today's team meeting?

Unsatisfied						Satisfied
	0	1	2	3	4	

10. How helpful do you believe the collaboration tool was to discuss scenarios in the meeting?

Not helpful						Helpful
	0	1	2	3	4	

11. How well could you interact with the remote participants in the meeting?

Not well						Very well
	0	1	2	3	4	

12. How effective was the communication during the meeting?

Not effective						Very effective
	0	1	2	3	4	

13. Please rate the communication process you used for generating scenarios in your team:

	0	1	2	3	4	
Impersonal						Personal
Bad						Good
Difficult						Easy
Informal						Formal
Simple						Complex

14. Please describe the **conversation** you had with your team members for discussing scenarios:

0 .. Strongly disagree 4 .. Strongly Agree	0	1	2	3	4
Pleasant					
Agreeable					
Interesting					
Argumentative					
Cooperative					

15. Did you find LiveNet useful for the team-meeting?

No						Yes
	0	1	2	3	4	

16. Would you apply a tool like LiveNet in the future?

No						Yes
	0	1	2	3	4	

Please use this page for additional comments on the Tool-Supported team meeting and suggestions for improvement.

Thank you for your contribution!

Final Questionnaire		
Student-ID		
Name		
LVA	☐ QS-VU	☐ Testen
Session	☐ Tuesday, June 3rd	☐ Friday, June 6th
Scenario Categories	☐ Used	☐ Not used

Please answer the questions in this questionnaire.

Note, that your answers will be treated anonymously. Your ID and name will be used to link your answers to your scenario reports. We will remove all personal data gathered for evaluation purposes afterwards.

1. Overall, did you feel **you** performed well in developing scenarios for non-functional requirements in:

	a distributed arrangement using the collaborative tool.
	both arrangements.
	a face-to-face arrangement.

2. Overall, did you feel your **group** performed well in developing scenarios for non-functional requirements in:

	a distributed arrangement using the collaborative tool.
	both arrangements.
	a face-to-face arrangement.

3. Did you feel that using the collaborative tool had any positive or negative affect on your **group discussion**? e.g., you may have been able to discuss issues more quickly (a positive effect) or you may have found it more difficult to discuss issues (a negative effect).

	Large positive effect
	Small positive effect
	No effect
	Small negative effect
	Large negative effect

4. Compared with face-to-face group meeting, do you feel that a collaborative tool based group meeting is

	more efficient?
	equally efficient?
	less efficient?

5. Do you like to contribute your opinion on sensitive issues during a meeting **anonymously** if its possible?

	Yes		No

6. Overall, what **type of meeting** you would like for generating scenarios, a face-to-face or distributed arrangement using a collaborative tool?

	Face-to-face arrangement
	Distributed arrangement using collaborative tool

7. While using collaborative tool to generate scenarios, were you able to **concentrate** on brainstorming and generating scenarios as well as in the face-to-face meeting?

	Yes		No

8. Please provide some comments of your experience regarding the following topics (see also backside of this questionnaire).

   a) Describe the effect that the collaborative tool had on your group meeting compared to a face-to-face meeting.

b) Were there any aspects of the **face-to-face** meeting that **facilitated/hindered** efficient scenario discussion?

c) Were there any aspects of the **distributed meeting** that **facilitated/hindered** efficient scenario discussion?

d) How can we **improve the efficiency and effectiveness** of the distributed meeting arrangement supported by a collaborative tool?

e) What are the **three most difficult issues** you faced while developing scenarios to specify non-functional requirements during this exercise?

Thank you for your contribution!