# An Empirical Study of Scenarios Gained and Lost in Architecture Evaluation Meetings

Dietmar Winkler
Vienna University of Technology
Favoritenstr. 9/188
A-1040, Vienna, Austria
+43 1 58801 18801
Dietmar.Winkler@tuwien.ac.at

Stefan Biffl
Vienna University of Technology
Favoritenstr. 9/188
A-1040, Vienna, Austria
+43 1 58801 18810
Stefan.Biffl@tuwien.ac.at

Muhammad Ali Babar
Lero, University of Limerick
Castletroy
Limerick, Ireland
+353 61 233639
malibaba@lero.ie

## ABSTRACT
An important element in scenario-based architecture evaluation is the development of scenarios by holding meetings of stakeholders. As the team meeting is an expensive activity, studying the effectiveness of meetings is an important research question. In this paper, we report the findings from analyzing the data collected in a controlled experiment aimed at empirically studying the effectiveness of scenario development meetings in terms of gained and lost scenarios. A secondary researched issue was whether or not a top-down technique for eliciting scenarios can improve the performance of a team meeting compared to a bottom-up technique. Findings from data analysis question the effectiveness of holding meetings for developing scenarios since more important scenarios were lost than gained in these meetings. Data results also provide empirical support to our assertion that top-down scenario development technique is better than the bottom-up technique.

## Categories and Subject Descriptors
K.6.3 [**Software Engineering**]: Software Management – Software process.

## General Terms
Measurement, Management, Experimentation.

## Keywords
Architecture evaluation, quality attributes, scenarios.

## 1. INTRODUCTION
Software architecture (SA) plays a vital role in achieving desired quality attributes (such as performance, security, and modifiability) in a system. That is why practitioners and researchers have been emphasizing the importance of addressing quality-related issues at the architecture level. The idea of predicting the quality of a software-intensive system from a high-level design description originated in Parnas's work on software modularization [33] and has recently emerged as an important quality assurance (QA) technique known as software architecture evaluation. There is a range of scenario-based architecture evaluation methods such as Architecture Trade-offs Analysis Method (ATAM) [23], Architecture Level Modifiability Analysis (ALMA) [27], and Performance Assessment of Software Architecture (PASA) [37]. The effectiveness of these approaches heavily depends on the ability of stakeholders to identify high-quality scenarios, as these scenarios are a key input to the evaluation process [9][25].There are several scenario-generation approaches such as brainstorming workshops [4], interviews [27], and use case analysis [37].

In the architecture evaluation process there are two steps to gather and refine scenarios (similar to the Fagan inspection process [16][17], another QA technique): 1. individual scenario generation and 2. discussion of individual scenarios in a team meeting to improve the quality (and possibly the quantity) of scenarios for architecture evaluation.

Recent research in the software inspection area has focused mainly on improving the effectiveness and efficiency of individual defect detection through improved defect detection techniques [6][13][14] and on assessing and optimizing inspection gains in team meetings [12]. An important challenge of applying inspections in industry is the large variation in inspectors' performance and the associated risk of ineffective but costly inspection meetings. While effectiveness of inspection team meetings has been a topic of significant research over the years, the effectiveness of team meetings on the quality of scenario profiles has not yet been empirically investigated in software architecture research. A secondary research issue is whether or not a structured technique (i.e., top-down) for eliciting scenario profiles can improve the performance of a team meeting compared to an unguided scenario elicitation technique (i.e., bottom-up).

This paper presents the results of analyzing the quantitative data gathered in a controlled experiment [1] for gaining a better understanding on the effectiveness of team meetings for developing quality attributes scenarios in the software architecture evaluation process and the impact of using two techniques (top-down and bottom-up) on the performance of a team based on the quality of scenarios that are developed to characterize quality attributes required by a system.

The paper is structured as following. Section 2 provides a brief overview on related work. Section 3 introduces the research questions and hypotheses. Section 4 summarizes main aspects of the experiment performed to gather the empirical data. Section 5 presents results followed by discussion on the results in Section 6. Section 7 concludes the paper.

## 2. BACKGROUND and MOTIVATION
This section summarizes relevant research from the areas of software architecture evaluation and software inspection.

## 2.1 Architecture evaluation

Software quality attributes of a software system such as performance, security, or changeability can be supported or inhibited by the software architecture of a software-intensive system [7]. Thus the evaluation of software architecture at an early stage has gained significant interest. Scenarios provide context for evaluating architecture to work with concrete examples enabling the user to understand their detailed effect [28]. Scenarios also help draw conclusions on the adequacy of a proposed architecture and available alternatives. Thus many mature software architecture evaluation methods are scenario based [3][22]. A set of scenarios is called a scenario profile.

The software architecture community has developed many frameworks for eliciting, structuring, and classifying scenarios. For example, Lassing *et. al.* [29] proposed a two-dimensional framework for eliciting scenarios, Kazman *et. al.* [25] proposed a generic 3-dimensional matrix to elicit and document scenarios. Bass *et al.* [7] provided a six-element framework to refine and structure scenarios. Scenarios used in software architecture evaluation are classified into various categories such as: direct scenarios, indirect scenarios, complex scenarios, use case scenarios, growth scenarios, and exploratory scenarios [7][24][29][31]. The Software Engineering Institute (SEI) has enumerated a collection of general quality-attribute scenarios that are intended to help characterize most commonly known quality attributes [8] such as performance, modifiability, and usability. A general scenario is, in effect, a template for generating a specific quality-attribute scenario. For example, two (abbreviated) modifiability general scenarios are:

- "Changes to the platform occur" and;
- "System needs to serve requests arrived from users."

Since not all general scenarios for a particular quality attribute may be relevant to a particular system or class of systems, an evaluator must identify relevant scenarios that should be made system specific [30]. We believe that general scenarios also help instigate thinking for system-specific scenarios called concrete scenarios [1]. However, general scenarios can also be classified according to domain-specific software change categories to help an evaluator to develop those general scenarios that may be more relevant and should be made system specific with the help of stakeholders for a particular system.

Despite the well recognized importance of having good quality scenarios and significant cost of having meetings for developing good quality scenarios, with one exception [9], there has been little research on determining the most effective way of gathering quality scenarios from stakeholders. Based on a controlled experiment, Bengsston and Bosch [9] concluded that prepared teams performed better than unprepared teams and individuals in terms of quality of the scenario profiles developed. However, there has been no research on understanding the scenarios gains and lost during meetings. Considering the importance of scenarios in architecture evaluation, and cost, logistics, and scheduling difficulties involved in arranging evaluation meetings, we believe that it is an important research issue. Hence, the research reported in this paper is motivated by the practical need to empirically determine an effective approach to guide the scenario development process on individual and team levels in order to gather high quality scenarios for architecture evaluation in a cost effective manner.

## 2.2 Software inspection team meetings

Similar to the architecture evaluation, software inspection also aims at assessing the quality of artifacts in the software development process. Some software inspection approaches also use scenarios to support finding quality issues [21]. While the results of the software inspection process are defect reports, the architecture evaluation process steps we investigate result in a list of evaluation scenarios. The general process seems sufficiently similar to consider taking experiences from software inspection to generate hypotheses for studying different aspects of the architecture evaluation process.

Initially, Fagan's software inspection [16] viewed the inspection team meeting as the key process step, while more recent approaches starting with Parnas and Weiss [32] focused more on individual inspection work to lower inspection effort as the team meeting is much more expensive than individual work. During individual work, inspectors can work in parallel and from different points of views, while in the team meeting only a limited number (usually two) of the inspectors can effectively interact at the same time while the others are just listening. Empirical results on software inspection meetings' effectiveness differ considerably. Fagan reported inspection meetings to be very effective [17][16], while more recent studies reported contradictory results [19][34][35][36].

Software inspection reports identify several potential benefits of meetings: 1. *Synergy:* The synergy effect assumes that meeting dynamics help find new defects. However, Votta [36] reports individuals already recorded 9 out of 10 defects that came out of the team meeting and thus only very little synergy can be achieved. Bianchi *et al.* [11] report that meeting losses (i.e., real defects found during individual preparation but then dismissed in the meeting as irrelevant or false positives) significantly outweigh meeting gains (i.e., defects newly found during the team meeting). Especially defects reported by only one inspector during individual work were lost. Johnson and Tjahjono [19][20] observed a substantial degree of synergy (30% to 40% of new defects detected). 2. *Identification of False Positives:* Land *et al.* [26] report that team meetings have a clear advantage over individual defect detection in discriminating between true defects and false positives. False positives are defect reports, which actually are not true defects. False positives can become a problem if they occur frequently because they incur costs, e.g., time spent on trying to diagnose and repair false positives. 3. *Soft Benefits* are meeting benefits apart from synergy and false positive reduction, including the sharing of review experiences, the dissemination of product-related knowledge and the creation of collective ownership for the review outcome among reviewers [19][20].

However, in the inspection area the reduction of false positives and the soft benefits seem not to justify the meeting costs. Similar to the concept of *False positives,* software architecture evaluation has a concept called *irrelevant scenarios,* those which are considered not relevant to the system whose architecture is being evaluated or scenarios which are not expected to be

materialized in a given duration, e.g., 3 to 5 years. Before evaluating an architecture, irrelevant scenarios are identified and excluded from the evaluation. The identification of irrelevant scenarios may be based on prioritization or experience of evaluators and/or stakeholders. Thus the research reported in this paper focuses on the evaluation of the synergy effect of the team meeting in software architecture evaluation, i.e., relevant architecture evaluation scenarios gained or lost in team meeting.

# 3. RESEARCH HYPOTHESES

The context for this experiment is a scenario development workshop (such as Quality Attribute Workshop (QAW) [4]) where stakeholders develop scenarios to precisely specify quality attributes (such as performance, reliability and security). These scenarios are used to assess the capability of a proposed architecture options with regards to the desired quality attributes characterized by these scenarios [7].

Like [9], we use a two-stage process of developing scenarios. First, each individual constructs a scenario profile alone. A profile is a set of scenarios. Second, individuals come together in teams to construct a joint scenario profile. We have mentioned that domain-specific software change categories can help stakeholders to develop those scenarios that may be more relevant. It is considered that the provision of software change categories can have most impact on the first stage of the scenario development process. Scenarios produced by individuals who are given the software change categories should include a larger proportion of the most relevant changes than scenarios produced by individuals who are not given the software change categories [1]. However, we believe that the provision of the software change categories can also help groups of stakeholders to perform better in scenario development meetings.

For empirical evaluation, we investigate the performance of individuals and teams to find scenarios in 3 categories: very important scenarios, important scenarios, and less important scenarios. The scenarios are assigned to one of the three categories (i.e., A, B, and C) based on the score assigned to each of the scenarios. The score of a scenario is assigned based on the number of times that scenario is reported by individuals and real teams (see Section 4.3.3 for details). The individuals use either a structured approach to find scenarios guided by so-called change categories (the treatment group) or and ad hoc approach (the control group). Performance can be measured on individual or on team level; there are real teams, which conduct a team meeting, and nominal teams, which do not meet; thus the scenario lists of nominal teams can be directly derived from the individual scenario lists of the team members. We propose the following null hypotheses for the reported research:

H01: Individuals in the treatment group develop similar number of scenarios for each of the categories as individuals in the control group.

H02: In the scenario development team meeting participants find no more new scenarios than they loose compared to individual work.

H03: 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.

The alternative hypotheses for this research are:

**H11 Individual guidance effects***:* Individuals in the treatment group find more scenarios than individuals in the control group. There are many reports in software inspection that support the effectiveness of so-called reading techniques, which support the inspector with specific guidance in identifying defects, e.g., checklists or scenario-based reading [6][11][17][19]. We expect the guided approach (i.e., provision of the software change categories) to enable individuals to find more scenarios in general and more important scenarios in particular as the change categories should ensure that a participant is unlikely to overlook an important category of quality attribute required of architecture.

**H12 Team effects:** In the architecture evaluation team meeting participants find more new evaluation scenarios than they loose compared to individual work. With this research, we follow the analysis procedure proposed by Bianchi *et al.* [11]. However, we apply the approach to architecture evaluation and extend their analysis by assessing the influence of different scenario generation techniques on the results. As scenario generation is more concerned with providing more scenarios than with the elimination of false positives, which is an important aspect in software inspection. We expect architecture evaluation teams to use only little time to discard scenarios but concentrate on adding new scenarios, which should yield considerably more if not more important scenarios compared to the individual inputs to the team meeting.

**H13 Team guidance effects:** Teams who use a guided method for scenario elicitation perform better than teams who use a non-guided method. Similar to the individual guidance effects we expect teams who use a structured approach to focus on eliciting more important scenarios in the given change categories.

# 4. EXPERIMENT DESCRIPTION

This section provides an overview on the experiment design, conduct, and threats to validity.

## 4.1 Experimental Design and Variables

The experiment design was a randomized balanced design, which used the same experimental materials for both treatments and assigned the subjects randomly to each treatment [38]. Both the assignment of individuals to treatment groups and to working teams was randomized using a sort card method of randomization. There were 12 participants in each the treatment groups, and 4 teams of 3 persons each in the team part of the experiment. Individuals in the scenario development process apply system requirements (functional and non-functional), process instruction, and supporting material, e.g., guidelines for scenario elicitation, questionnaires for background information and skill collection. Output is a list of scenarios characterizing a required quality attribute (i.e., modifiability in this study).

Team development of scenarios: inputs are the scenarios developed by individual team members; output is a list of scenarios developed by a team. The team scenarios are developed in a team meeting in which the team scenarios are based on the scenarios developed by each member of a team during the first phase and based on the team brainstorming, interaction, and discussion.

Independent variable of this study is a list of domain specific categories of software changes provided to the participants dur-

ing scenario development activity, with one treatment: change categories provided, and one control: change categories not provided (represented by the treatment and control groups).

The dependent variable is the frequency of each scenario developed by the participants a) individually, b) in real 3-person teams, and c) in nominal 3-person teams.

- Individual performance: number of scenarios reported in each of the three scenario categories (A, B, C).
- Real team performance: number of scenarios reported in each of the three scenario categories by a team of 3 individuals, who conducted a team meeting to discuss their scenarios and converged to a common team list. From discussion new ideas for scenarios may emerge and false positives in the individual lists may be eliminated by team consensus.
- Nominal team performance: number of scenarios reported in each of the three categories by a team of 3 individuals, who form a non-communicating team, i.e., there is no team meeting but an editor combines the individual scenarios into a team scenario profile (no synergy and no removal of false positives).

Each scenario has been assigned one of the three scenario categories (A, B, and C) based on that scenario's score, which is the number of times that scenario is identified in all scenario profiles (individuals as well as teams). A category of scenario represents its relative importance in this study (see Section 4.3.3 for further details). It is also worth mentioning that we consider the number of scenarios mentioned by an individual or team in each of three categories but the performance comparison is mainly based on the number of scenarios in the most important category of scenarios called category A.

We have mentioned that this research is mainly interested in two things. Firstly, we want to evaluate the *effectiveness of scenario development meetings* in terms of *lost and gained scenarios*. Secondly, we want to learn more about the scenarios, which are actually lost during meetings.

As far as the scenario development meeting effectiveness is concerned, we analyze the number of scenarios gained and lost during a meeting. A *gained scenario* is a scenario newly introduced during the meeting (not included in an individual scenario profiles of the team members). A *lost scenario* is a scenario that was found during the individual preparation phase but was not included in the team scenario profile. We assume that this scenario was not accepted by the team during the meeting (not found on the team scenario list). For this analysis, we compare the performance of nominal and real teams. The performance of a *nominal team* is based on the performance of all team members during individual preparation. Its effectiveness depends on the scenario identification performance of each individual of an evaluation team during individual preparation and the scenario overlap among team members. The performance of a *real team* on the other hand is evaluated after the scenario development meeting where all individually developed scenarios are discussed and a team scenario profile is developed. The group scenarios are developed in a meeting following a simplified process of developing scenarios like QAW [4]. Each member of the group presents his/her individual scenarios for group discussion about the inclusion or exclusion of each scenario in the group scenario profile. The group members also brainstorm new scenarios to characterize the quality attributes.

## 4.2 Experimental Context and Subjects

The 24 participants in the study were recruited from a software architecture course offered at the University of New South Wales, Australia. The experiment was part of a scenario development workshop, which was one of the assessment tasks in that course [1]. The students were briefed about the objective and procedure of the study. They had the option of withholding their results from research. Written permission was sought from the participants to use their data in this study.

Most of the students were post-graduate students with the exception of 3 fourth-year undergraduate students, who had maintained average marks at 75% or better (a requirement to enroll in this course for undergraduates). The ratio of male and female was representative of the traditional software engineering courses and industry with only 5 female students. All of the participants were either working or had worked as information technology (IT) professionals with an average working experience of 4.5 years in the IT industry and were of an average age of 27 years. Their working experience typically had a good mix of design, coding, test, maintenance, and technical support activities.

## 4.3 Experimental material

Two lectures (2 hours each) were dedicated to topics directly related to the experimental study, i.e., quality attributes, software architecture evaluation, and approaches to brainstorm and structure general and concrete scenarios in order to characterize quality attributes. During the course, there was also one class exercise to brainstorm and structure scenarios for a system the students were familiar with.

One week before the study, all the participants received detailed information about the system, *LiveNet*, for which they were supposed to develop software change scenarios. One of the authors has used LiveNet to create a network of workspaces designed to support various activities of the software architecture evaluation process such as architecture presentation, scenarios development and impact analysis. Each workspace had roles, artifacts and different collaborative features. The participants were assigned different roles (such as software architect, software engineer, and maintainers) in a few workspaces and asked to interact with various features of the system. A short document describing various features of LiveNet was also provided a week before the study.

Before the study all the participants attended a 30 minutes refresher session covering the concepts of constructing change scenarios for architecture evaluation, quality attributes, general and concrete scenarios. However, our study did not require the participants to have any experience in architecture evaluation. The duration and format of our training was designed to make the participants representative of most stakeholders involved in real-world architecture evaluation, where stakeholders normally receive minimum training in creating scenarios.

### *4.3.1 Software requirements specifications*

This study used the Software Requirement Specification (SRS) for a web-based collaborative, LiveNet [15]. LiveNet provides a generic workflow engine and features to support collaboration among geographically distributed members of a team, e.g., synchronous chat, discussion forum, document repository, notification, roles, planning tools, and task assignment tool. LiveNet enables users to create workspaces and define elements of a particular workspace. LiveNet also supports emergent business processes. We prepared a simplified version of an SRS and a description of the system to provide the participants with as clear a picture of the system as possible.

### *4.3.2 Software Change Categories Used*

We have mentioned that a classification of software change categories can be used as a guide to help stakeholders to come up with better quality scenarios [1]. A scenario classification scheme can be derived from the application domain, knowledge of potentially complex scenarios, or some other source of engineering knowledge. In order to derive the categories of changes used in our research, we draw upon in-depth knowledge of the collaborative-applications domain and the types of complex changes made overtime in LiveNet. We followed an iterative process of building the classification scheme. Based on more than five years experience with LiveNet as researchers and users, we came up with a list of major categories of changes most likely required in a web-based groupware like LiveNet. Then that list was reviewed by the chief research investigator and software architect of LiveNet, both of whom were involved with the project throughout its life. Based on their feedback, the list of change categories was refined. Following is a brief description of each of these categories:

- User Interface (UI) – Changes in the User Interface of application.
- Security Policy (SP) – Changes needed for increased security of application and content.
- Performance and Scalability (PS) – Changes required for increased performance or handling more users without decreasing performance.
- Workflow Management (WM) – Changes to provide various features to support different business processes.
- Content Management (CM) – Changes needed to improve/add content management features.

### *4.3.3 Scheme for Marking Scenario Profiles*

In order to assess and compare the performance of the individuals in the control and experimental groups, and real and nominal teams, we needed a suitable marking scheme. Our previous studies [2][1] in this line of research have used a marking scheme that ranks scenario profiles by comparing them with a reference scenario profile [9]. However, for this study, we decided to come up with another marking scheme, which is based on score for each scenario that reflects the frequency of occurrence of that particular scenario in all scenario profiles (i.e., individual and groups). That means the score of an individual scenario is based on the frequency of that scenario being reported by individuals and real teams. This marking scheme assumes that a higher number of occurrences indicates a higher importance of a scenario.

By using this marking scheme, we identified the importance of each scenario by counting the occurrences of each scenario in all individual and teams scenarios profiles (32 total, 24 individual and 8 team profiles). The score given to each scenario was used to classify that particular scenario in one of the three scenario categories, which reflect the relative importance of a scenario. The overall score was calculated by summarizing the individual and real team scores (frequency of a scenario found). The top 20% of scenarios based on a descending order of all scenarios' frequencies are considered as "most important" (class A) with scenarios' score > 30. After the top 20%, another 40% are considered as important (class B) with scenarios' score between 4 and 30, and the last 40% are considered less important (class C) with scenarios' score < 4. Here score means the frequency of occurrence of each scenario.

According to this method, the actual scenario profile for each individual and team must be re-coded into a standard format for analysis. This approach to assigning scenarios to different categories is based on the assumption that the importance of a scenario can be determined by the number of scenario profiles in which that particular scenario appears. The appearance of a scenario in a scenario profile shows that the creator of that profile considered that scenario to be relevant. Hence, the more participants mention a particular scenario, the more relevant it becomes. That means the most relevant scenario will have the highest frequency of occurrence in all the scenario profiles created by individuals and real teams.

## 4.4 Experiment Execution

The experiment was conducted as a part of scenario development workshop for the course as mentioned in section 3.2. Prior to the experiment, the number of teams were identified based on the expected number of the participants (3 members each team). Each team was assigned a name and three participants were assigned to each team. Each team was allocated to one of the two experimental conditions (treatment and control). The participants to the teams and the teams to the experimental conditions were assigned randomly using card sort randomization.

All the 24 participants arrived according to the schedule. There was a 30 minute briefing session to revise the lecture material on software architecture evaluation process, generating quality sensitive scenarios, and LiveNet system. Participants were given a document describing the collaborative application, architecture evaluation process and example scenarios.

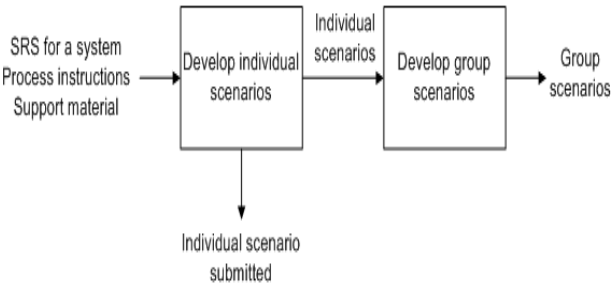


**Figure 1: Two-step process of developing a scenario profile.**

After the briefing session, the participants were given a simplified version of requirements for LiveNet. The participants in the treatment group also received a document describing the five

categories of most commonly occurring changes in a collaborative application like LiveNet. They were encouraged to use the categories to stimulate their thinking about the types of changes that may be expected to occur over the coming three years during scenario development exercise. The participants followed the two phase process of developing individual and team scenarios shown in Figure 1. The second step is performed in a meeting session, which is held face-to-face.

The participants were asked to develop software change scenarios individually for 35 minutes. When 35 minutes of time had passed the profile of individuals were collected, photocopied and returned to them. All the participants were asked to join their respective teams to develop team scenarios for 40 minutes.

Once 40 minutes of time had elapsed, the team scenario profiles were collected. After developing individual and team scenario profiles, there was a debriefing session during which the participants also filled a post-session questionnaire. However, the analysis of the data collected through the questionnaire is not within the scope of this paper.

## 4.5 Validity considerations

Every empirical study has to deal with several threats to internal and external validity. In the following sub-sections, we discuss the major threats to this study and the countermeasures we applied.

### *4.5.1 Threats to internal validity*

Internal validity is the degree to which the values of dependent variables can only be attributed to the experimental variables. [38]. In order to avoid bias in allocating participants to the treatment groups, we randomized the assignment by using a sort card method. We wrote the names of the participants and groups on plain cards. After shuffling the cards, we assigned one card to each group (treatment and control) without seeing the individual's or group's name on the card.

Another threat to the internal validity of our experiment is the appropriateness of the approach to classifying into three categories as an indication of their respective importance based on the frequency of occurrence of each scenario. This approach is similar to the method of measuring the quality of scenarios profiles developed for architecture evaluation and has been used in several studies [9] including ours. Moreover, we argue that deciding about the relative importance of each scenario based on its frequency is similar to the common approach to prioritizing quality attribute scenarios based on stakeholders' votes [4].

Another potential threat associated with this approach is the skill, knowledge, and bias of the person, who recodes each scenario, removes duplication, and assesses them semantic equivalence in each scenario profile before counting their occurrences. We addressed this issue by having two researchers perform these tasks independently Any disagreement regarding occurrences of each scenario in all profiles was resolved before counting its occurrences.

### *4.5.2 Threats to external validity*

External validity is the degree to which the results can be generalized, i.e. transferable to other similar situations. In particular, it is important to consider whether the participants are representative of the stakeholders who would undertake architecture evaluation in the industry, and whether the experimental materials and process are representatives of the process and materials used in industrial architecture evaluations.

In an industrial evaluation, stakeholders may have a variety of different backgrounds (such as software engineering, marketing, management and sales). This was not the case in our experiment. All the participants had educational and professional backgrounds in either computer science or software engineering. This means that our results are more likely to generalize to stakeholders with a technical background than stakeholders with a non-technical background.

Secondly, stakeholders in an industrial situation are more likely to have considerable experience of the application being evaluated, whereas the participants in our experiment only had limited knowledge of LiveNet. That means our results are most likely to apply to stakeholders with not very extensive experience of application being evaluated.

The participants had limited experience of software architecture evaluation and of developing scenarios for quality attributes. As far as we are aware, organizations normally do not provide extensive training to their employees for software architecture evaluation or developing quality-sensitive scenarios. Thus, the experience of the experimental participants is likely to be similar to that of stakeholders performing an industrial evaluation

The software requirements specifications used in the experiment is relatively short and simple compared with a typical industrial one. However, in industry stakeholders would be given both more requirements and a more time to develop their scenarios. Finally, there may be a threat to the external validity if the scenario development process used in our study is not representative of the industrial practices for developing scenario profiles for software architecture evaluation. However, the scenario development process in our experiments was similar to the one used for most of the scenario-based software architecture evaluation methods, which gather scenarios to characterize quality requirements to be fulfilled by a proposed software architecture through brainstorming workshops like QAW [4].

# 5. RESULTS

## 5.1 Data analysis procedure

The data analysis takes as inputs the scenarios developed during the experiment by individuals and real teams. Based on the individual and team scenario lists, we calculated the frequency of each reported scenario by individual and real teams. The frequency of each reported scenario is the baseline for placing that particular scenario into a category, which represents its importance. As preparation for statistical evaluation, the scenario descriptions were matched and divided into 3 scenario categories: ("A", "B", and "C") (see Section 4.3.3 for details about the marking scheme).

Performance in the 3 scenario categories was measured from individuals, real 3-person teams, and nominal 3-person teams with respect to the scenario classes (A, B, C). We identified the number of scenarios based on the individual/team scenario list

for individuals and real teams. Regarding nominal (i.e., non-communicating) teams, we determined the team scenario list by combining the individual lists of a team (a scenario must be identified by at least one team member).

For statistical analysis, we apply descriptive statistics, i.e. mean, standard deviation, and box plots to visualize the results. Furthermore, we also apply the non parametric Mann-Whitney-test at a significance level of 95% to test our hypotheses.

We gathered 104 unique scenarios from 32 scenario profiles, i.e., a set of scenarios, (24 individual scenario profiles and 8 team scenario profiles). Using the above-mentioned classification scheme, we classified all the scenarios into relevant categories. Table 1 presents an overview of the scenarios classified in each of the three categories.

**Table 1: Scenario Classification.**

| Class A | | Class B | | Class C | | Total | |
|---|---|---|---|---|---|---|---|
| No. | % | No. | %. | No. | % | No. | % |
| 22 | 21% | 41 | 39% | 41 | 39% | 104 | 100% |

## 5.2 Scenarios found by Individuals

Individuals reported on average around 9 scenarios; participants in the treatment group more than 10 scenarios and participants in the control group 8 scenarios [1]. Applying the Mann-Whitney test, we observed a significant difference in the control and treatment groups' members (p-value: 0.037). Table 2 summarizes the results of individual participants placed in the control and treatment groups.

**Table 2: Number of scenarios reported by individuals.**

| Scenario category | Control Group | | Treatment Group | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Class A | 4.3 | 2.10 | 6.8 | 2.29 |
| Class B | 3.0 | 1.95 | 3.5 | 2.02 |
| Class C | 0.7 | 0.99 | 0.4 | 0.90 |

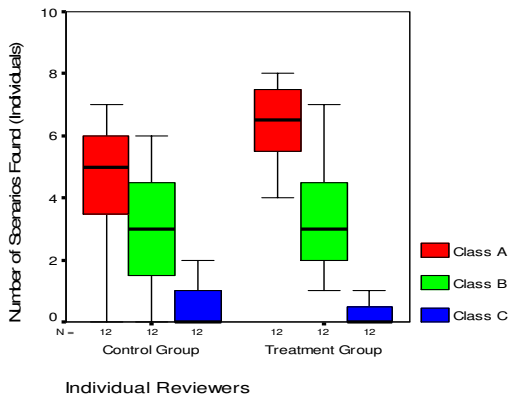


**Figure 2: Number of scenarios in each category reported by individuals.**

Figure 2 compares the number of scenarios found in the 3 scenario categories for the treatment and control groups. Participants in the treatment group found overall more scenarios than participants in the control group. Particularly in the most important category A scenarios, the treatment group participants reported significantly more scenarios (p=0.015). We do not observe any significant differences for class B (important) and class C (less important scenarios).

## 5.3 Scenarios found by real 3-person teams

Teams of 3 persons reported after their meeting on average around 15 scenarios, about 90% more scenarios than an average individual. Guidance for the treatment group resulted on average in notably more class A scenarios, but less class B and class C scenarios than the control group. The treatment group shows for class A scenarios less variance, possibly due to better guidance provided through the change categories during the scenarios development process. Table 3 provides a deeper insight into the results of real teams regarding control/treatment groups and different scenario categories.

**Table 3: Number of Scenarios found by Real Teams.**

| | Control Group | | Treatment Group | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Class A | 5.8 | 4.79 | 9.8 | 1.26 |
| Class B | 6.8 | 3.10 | 5.3 | 2.87 |
| Class C | 2.8 | 4.27 | 1.5 | 1.29 |

## 5.4 Scenarios found by nominal 3-person teams

Nominal teams of 3 persons reported on average 21 scenarios, around 140% more scenarios than an average individual.

**Table 4: Number of Scenarios found by Nominal Teams.**

| | Control Group | | Treatment Group | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Class A | 10.0 | 3.37 | 13.5 | 2.65 |
| Class B | 8.8 | 3.78 | 8.5 | 3.00 |
| Class C | 2.0 | 2.71 | 1.3 | 1.89 |

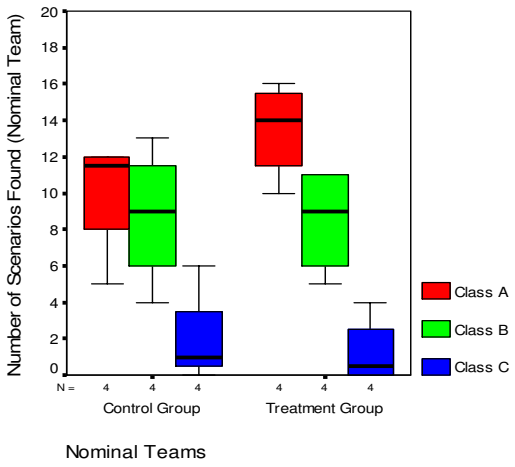


**Figure 3: Number of scenarios reported by nominal teams.**

The treatment group found on average notably more class A scenarios, but less class B and class C scenarios than the control group (similar finding as with the real teams). Table 4 summarizes the number of scenarios of nominal teams for the control

and treatment group regarding scenario categories. Figure 3 presents a box-plot according to the number of identified scenarios by nominal teams (i.e., a scenario was found by at least one team member) regarding scenario classes and groups.

Note that the number of identified scenarios is higher for all important (i.e., class A and B scenarios) for the nominal teams than for the real teams. Regarding class C, i.e. less important scenarios, we could observe advantages for the real team.

## 5.5 Comparison of nominal and real team performance

Real 3-person teams reported on average around 15 scenarios, while nominal 3-person teams reported on average 22 scenarios, which is a very interesting finding as nominal teams appear to be more effective than the real teams in terms of the number of scenarios reported. It is also an indication of meeting loss which occurs as a result of some individual scenarios not being able to make their way to the group scenarios. Thus a real 3-person team reported 70% more scenarios than an individual, while taking about 100% more time than the time the individuals took. A nominal team reported 135% more scenarios than an individual with the same amount of time because three members of the nominal team worked concurrently.

While real teams reported a comparable number of class C scenarios, a considerable number of important and a significant (p=0.03) number of very important scenarios was lost in the meeting step, on average 2.6 class B scenarios and 3 class A scenarios. Figure 4 presents the number of identified scenarios by real teams and nominal teams and table 5a summarizes mean and standard deviation for this evaluation.

**Table 5a: Number of Scenarios found by teams.**

| Scenario category | Real Team | | Nominal Team | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Class A | 7.8 | 3.88 | 11.8 | 3.37 |
| Class B | 6.0 | 2.88 | 8.6 | 3.16 |
| Class C | 2.1 | 3.00 | 1.6 | 2.20 |

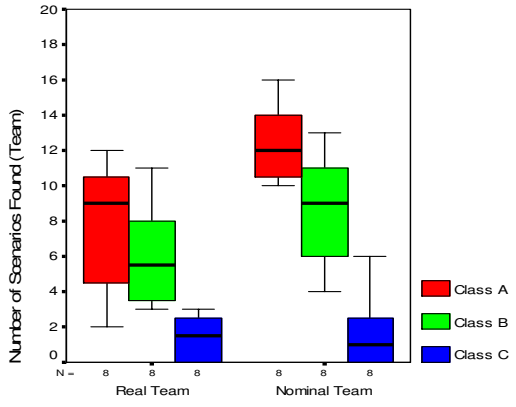


**Figure 4: Number of scenarios in each category reported by real and nominal teams.**

Meeting gain and loss is a quite interesting question because a team meeting might be helpful to elicit additional scenarios (positive team effect) or may hinder scenario elicitation (negative team effect). Thus, we compared the real team results and the results of a nominal team. Within a nominal team, a scenario has been identified if at least one team member noted the scenario in his/her record, without performing a team meeting. Table 5b presents the results of the gained and lost scenarios regarding scenario categories.

**Table 5b: Scenario gain and loss by scenario class.**

| | Gain | | Loss | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| Class A | 3.0 | 2.14 | 7.0 | 2.73 |
| Class B | 4.4 | 2.51 | 7.0 | 2.73 |
| Class C | 2.1 | 3.00 | 1.6 | 2.20 |

A closer look at scenarios gained and lost in the team meeting step reveals: Only for the least important class of scenarios on average more scenarios were reported than lost. For important and very important scenarios on average 3 to 4 more scenarios were lost than gained. These results seriously question the effectiveness of team meetings as conducted in the experiment.

# 6. Discussion

Our research in the area of software architecture evaluation aims to reduce the time, resources and skills required to effectively and efficiently evaluate proposed architectures. Our assertion is that one way of achieving this goal is to improve the scenarios development activity of the software architecture evaluation process. This assertion is based on several reasons. Developing scenarios is considered to be the most expensive and time consuming activity of architecture evaluation. The accuracy of the results of evaluation exercise is largely dependent on the quality of the scenarios used in the evaluation [9][25].

Scenarios are usually developed in group meetings. As the group meeting is an expensive undertaking, an important question is whether the meeting is worthwhile. Empirical studies on software inspection meetings have provided mixed results. This paper reports an empirical research aimed at studying the effectiveness of group meetings for developing scenarios based on scenarios gained and lost. Moreover, this study also further investigate our premise that a top-down (provision of domain-specific scenario categories) technique for developing scenarios is better than a bottom-up (brainstorming without any support material) technique.

This paper presents the results of data analysis to assess the effects of team meetings based on scenarios gained and lost and effects of the provision of software change categories to be used for guiding the scenario development process on the performance of individuals and groups working in real as well as nominal teams. Based on our experiences in conducting scenarios development workshops and empirical findings from knowledge acquisition and decision making disciplines, we assumed:

**H11 Individual guidance effects***:* a) Individuals in the treatment group reported significantly more scenarios in general. Summarizing all scenario classes we observed significant differences (p-value: 0.037) of treatment and control group at a significance level of 95% (Mann-Whitney Test).

H11b) Individuals in the treatment group report more important scenarios in particular as the change categories should ensure

that a participant is unlikely to overlook an important architecture quality category.

Data from the experiment supports that individuals from the treatment group reported significantly more class A scenarios (p=0.015).

**H12 Meeting effect on group performance:** In the software architecture evaluation team meeting for developing scenarios, participants find more new scenarios than they loose compared to individual work. Analysis of the data from this study reveals that 3-person teams reported only 50% to 150% more scenarios than an individual working alone. Moreover, the real teams reported (around 30%) less class A and class B scenarios than nominal teams. However, the meeting loss is independent of the provision with software change categories to guide the scenarios development process.

This hypothesis was not confirmed through the data analysis as team meetings lost on average more scenarios than were newly found. Rather we found a significant loss (p=0.030) for the most important scenario class A. This finding is similar to the experience with software inspection teams. We find this result very surprising as the elicitation of scenarios seems to be easier than the decision whether a defect report is a false positive. Thus we see considerable room for improvement of the process and tool support of architecture evaluation meetings as proposed in [13].

**H13 Team guidance effects:** Teams who use a guided method for scenario elicitation perform better than teams who use a non-guided method. Similar to the individual guidance effects, we expect teams who use a structured approach to focus on eliciting more important scenarios in the given change categories. We observed a significant difference (p-value=0.006) of real and nominal teams regarding the number of identified scenarios (scenario classes A, B, and C).

## 7. Conclusion and Further Work

Our analysis of the data collected from a controlled experiment has provided some very useful insights into the significant aspects of meetings for developing scenarios during architecture evaluation. It has also provided empirical evidence to support some of our assumptions and experiences in designing and conducting quality scenarios workshops. In the reported experimental context, software architecture evaluation meetings were not effective for scenario generation compared to the collection of the results from individual preparation. Therefore, we question the effectiveness of workshop style meetings of large number of stakeholders to develop scenarios for evaluating architectures as described in [4] and suggest using an architecture evaluation process without a meeting in order to maximize effectiveness of scenario generation.

Alternatively, new approaches should be introduced with the goal to keep scenario gains while avoiding scenario losses.

Different scenario elicitation techniques applied during individual preparation do not significantly influence meeting performance, i.e. the ratio of scenarios gained and lost. Actually this result seems rather surprising especially when considering that there should be a higher scenario overlap among participants who follow a guided approach. Thus, teams using a guided approach could loose more scenarios than teams using no structured approach. However, this effect could not be observed, possibly as architecture evaluation scenarios may be more diverse than defects reported in software inspection.

For further work in the area of architecture evaluation meetings in general and scenarios development meetings in particular, we suggest the introduction and assessment of new techniques and/or tools such as groupware support systems or electronic meeting systems that are well established in other areas. We assert that such tools can help overcome many problems related to paper-based meetings, e.g., a maximum of two participants communicating at the same time. Therefore, we believe that such tools can significantly increase the effectiveness and especially efficiency of architecture evaluation meetings. We have provided empirical evidence to support the some of proposed solutions in [4], however, further empirical research is required to study the proposed solutions more rigorously and thoroughly.

## REFERENCES


[1] Ali-Babar M. and Biffl S.: *Eliciting Better Quality Architecture Evaluation Scenarios: A Controlled Experiment on Top-Down vs. Bottom-Up*, Proceedings of the International Symposium on Empirical Software Engineering, 2006.

[2] Ali-Babar M., Kitchenham B. and Maheshwari P.: *The Value of Architecturally Significant Information Extracted from Patterns: A Controlled Experiment*. Proc. Australian Software Engineering Conference, 2006.

[3] Ali-Babar M., Zhu L., and Jeffery R.: *A Framework for Classifying and Comparing Software Architecture Evaluation Methods*. 15th Australian Software Engineering Conference, 2004.

[4] Ali-Babar M, Kitchenham B, Jeffery R., Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiement, Empirical Software Engineering Journl, 13(1): pp. 39-62, 2008

[5] Barbacci M.R, et al.: *Quality Attribute Workshops (QAWs)*, Tech Report CMU/SEI-2003-TR-016, SEI, Carnegie Mellon University, USA, 2003.

[6] Basili V.R., Green S., Laitenberger O., Lanubile F., Shull F., Soerumgaard S., and Zelkowitz M.: *The Empirical Investigation of Perspective-Based Reading*. Empirical Software Engineering: An International Journal 1, 2, pp. 133-164, 1996.

[7] Bass L., Clements P. and Kazman, R.: *Software Architecture in Practice*. Addison-Wesley, 2003.

[8] Bass L., Klein M. and Moreno G.: *Applicability of General Scenarios to the Architecture Tradeoff Analysis Method*. Technical Report CMU/SEI-2000-TR-014, Software Engineering Institute, Carnegie Mellon University, 2001.

[9] Bengtsson P. and Bosch J.: *An Experiment on Creating Scenario Profiles for Software Change*. *Annals of Software Engineering*, *9*, pp. 59-78, 2000.

[10] Bengtsson P., Lassing N., Bosch J., and Vliet, H.: *Analyzing Software Architectures for Modifiability*. Technical Report HK-R-RES-00/11-SE, Hogskolan Karlskeona/Ronneby, 2000.

[11] Bianchi A., Lanubile F., Visaggio G.: *A Controlled Experiment to Assess the Effectiveness of Inspection Meetings*, Proc. Metrics 01, London, 2001.

[12] Biffl S., and Halling M.: *Investigating the defect detection effectiveness and cost benefit of nominal inspection teams*, IEEE Transactions on Software Engineering, Vol 23, Issue 5, p385-397, 2003.

[13] Biffl S., Grünbacher P., and Halling M.: *A Family of Experiments to Investigate the Effects of Groupware for Software Inspection*, Journal of Automated Software Engineering, Vol 13/3, p373-394, 2006.

[14] Biffl S., and Halling M.: *Software Product Improvement with Inspection*, Proc. of Euromicro 2000 Workshop on Software Product and Process Improvement, Maastricht, IEEE Comp. Soc. Press, 2000.

[15] Biuk-Aghai R.P. and Hawryszkiewyez I.T.: *Analysis of Virtual Workspaces. Proceedings of the Database Applications in Non-Traditional Environments*, 1999.

[16] Fagan, M.: *Design and Code Inspections To Reduce Errors In Program Development*, IBM Systems J., vol. 15, no. 3, pp. 182-211, 1976

[17] Gilb T., and Graham D.: *Software Inspection*, Addison-Wesley, 1993.

[18] Host M., Regnell B. and Wohlin C.: *Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. Empirical Software Engineering*, *5*, pp. 201-214, 2000.

[19] Johnson P. M., Tjahjono D.: *Assessing software review meetings: A controlled experimental study using CSRS*, Proc. ICSE 97, Boston, 1997.

[20] Johnson P.M., Tjahjono D.: *Does Every Inspection Really Need a Meeting*, Empirical Software Engineering, 1998.

[21] Laitenberger, O., DeBaud, J.-M.: *An encompassing life cycle centric survey of software inspection*, Journal of Systems and Software 50(1): 5-31, 2000.

[22] Kazman R., Abowd G., Bass L. and Clements P.: *Scenario-Based Analysis of Software Architecture. IEEE Software Engineering*, *13* (6), pp. 47-55, 1996.

[23] Kazman R., Barbacci M., Klein M., and Carriere S.J.: *Experience with Performing Architecture Tradeoff Analysis. Proc. of the 21th International Conference on Software Engineering*, ACM Press, 1999.

[24] Kazman R., Bass L., Abowd G. and Webb M.: *SAAM: A Method for Analyzing the Properties of Software Architectures. 16th Int'l Conf. of Software Eng.*, 1994.

[25] Kazman R., Carriere S.J. and Woods S.G.: *Toward a Discipline of Scenario-based Architectural Engineering. Annals of Software Engineering, Kluwer Academic Publishers*, *9* (1-4), pp. 5-33, 2000.

[26] Land L.P.W., Jeffery R., Sauer C.: *Validating the Defect Detection Performance Advantage of Group Designs for Software Reviews*, Proc. ESEC / SIGSOFT FSE, 1997.

[27] Lassing N., Bengtsson P., Bosch J. and Vliet, H.V.: *Experience with ALMA: Architecture-Level Modifiability Analysis. Journal of Systems and Software*, *61* (1), pp. 47-57, 2002.

[28] Lassing N., Rijsenbrij D. and van Vliet, H.: *How Well can we Predict Changes at Architecture Design Time? Journal of Systems and Software*, *65* (2), pp. 141-153, 2003.

[29] Lassing N., Rijsenbrij D. and van Vliet, H.: *On Software Architecture Analysis of Flexibility, Complexity of Changes: Size isn't Everything. Proc. of 2nd Nordic Software Architecture Workshop*, 1999.

[30] Liu A., Bass L. and Klein M.: *Analyzing Enterprise JavaBeans Systems Using Quality Attribute Design*. Technical Report CMU/SEI-2001-TN-025, Software Engineering Institute, Carnegie Mellon University, 2001.

[31] Miller J., Wood M., Roper M.: *Further Experiences with Scenarios and Checklists*, Empirical Software Engineering, 3, 37-64, 1998.

[32] Parnas D. L. and Weiss D. M.: *Active design review: principles and practices*. Proc. 8th Int. Conf. on Software Engineering, pages 215-22, Aug. 1985.

[33] Parnas D.L.: *On the Criteria To Be Used in Decomposing Systems into Modules*, Communication of the ACM, 15(12): pp. 1053-1058, 1972.

[34] Porter, A. A., Johnson, P. M.: *Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies*, IEEE Transactions on Software Engineering, Vol. 23, No. 3, 1997.

[35] Seaman C.B., Basili V.R.: *Communication and Organization: An empirical study of Discussion in Inspection Meetings*, IEEE Transactions on Software Engineering, Vol. 24, No. 6, 1998.

[36] Votta L.: *Does every Inspection need a Meeting?* ACM Software Eng. Notes, vol. 18, no. 5, pp. 107-114, 1993.

[37] Williams L.G., and Smith C.U.: *PASA: A Method for the Performance Assessment of Software Architecture. Proc. of the 3rd Workshop on Software Performance*, 2002.

[38] Wohlin C., Runeson P., Höst H., Ohlsson M.C., Regnell B., and Wesslén A.: *Experimentation in Software Engineering - An Introduction*, Kluwer International Series in Software Engineering, Kluwer Academic Publishers, 2000.