IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 26, 2019, accepted June 6, 2019, date of publication June 12, 2019, date of current version June 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922265

An Empirical Study of the Effectiveness of Software Architecture Evaluation Meetings

M. ALI BABAR[1], HAIFENG SHEN[2], STEFAN BIFFL[3], AND DIETMAR WINKLER[3]

[1]School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia
[2]Peter Faber Business School, Australian Catholic University, Sydney, NSW 2060, Australia
[3]Institute of Information Systems Engineering, Vienna University of Technology, 1040 Vienna, Austria

Corresponding author: Haifeng Shen (haifeng.shen@acu.edu.au)

ABSTRACT One of the most important issues in scenario-based software architecture evaluation is the development of scenarios for characterizing a desired set of quality attributes by holding meetings of stakeholders. As such meetings are costly to arrange, studying the effectiveness of such meetings is an important research question. In this paper, we report on the findings from a software architecture evaluation empirical study aimed at investigating the effectiveness of scenario development meetings in terms of the quantity of gained and lost scenarios during the evaluation process. The findings from the experiment data analysis raise the question about the effectiveness of holding meetings for developing scenarios as more existing scenarios were lost than new scenarios were gained as a result of these meetings.

INDEX TERMS Architecture evaluation, process improvement, controlled experiment, scenario development, empirical study.

I. INTRODUCTION

Software architecture plays a vital role in achieving desired quality attributes (such as performance, usability, security, and modifiability) of a system [22]. That is why software practitioners and researchers have been emphasizing the importance of addressing quality-related issues at the architecture level [18]. 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 [39] and has recently emerged as an important quality assurance (QA) technique known as software architecture evaluation [30]. There are a range of scenario-based software architecture evaluation methods such as the Architecture Trade-offs Analysis Method (ATAM) [1], [26], the Architecture Level Modifiability Analysis (ALMA) [33], and Performance Assessment of Software Architecture (PASA) [44]. The effectiveness of these approaches heavily depends on the ability of stakeholders to identify high-quality scenarios, which are a key input to the software architecture evaluation process [11], [28]. There are several scenario-generation approaches such as workshops [7], interviews [33], and use case analysis [44].

For software architecture evaluation, scenarios are usually developed in a two-stage process similar to a Fagan inspection process [19], [20], another QA technique: (1) individual scenario development and (2) discussion of individual scenarios in a team meeting to improve the quality/quantity of scenarios. Recent research on software inspection has focused mainly on improving the effectiveness and efficiency of individual defect detection through improved techniques [16], [38] and on assessing and optimizing inspection gains in team meetings [2], [14]. An important challenge of applying inspections in industry is the associated risk of ineffective and costly inspection meetings [14]. In a similar spirit, it is important to ask the question about the effectiveness of team meetings on generating scenario profiles in a software architecture evaluation process.

To answer this question, we conducted an empirical study aimed at investigating the effectiveness of scenario development meetings in terms of the quantity of gained and lost scenarios as a result of these meetings during the evaluation process of a sample software application known as *LiveNet* [17]. It has already provided 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 tools. It also enables users to create workspaces and define elements

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Angiulli.

within a particular workspace that provides a context for cooperation without imposing a rigid process structure. The empirical study is based on a case scenario that is intended to extend *LiveNet* to support a distributed software inspection process by providing an intuitive and user friendly environment to help various roles (such as inspection moderator, inspectors, or authors) perform various activities (such as planning, individual preparation, inspection meeting, or follow up) of software inspection.

It is important to clarify that the research findings are only drawn from a quantitive method based on the quantity of scenarios. Quality of scenarios, which is equally important if not more, is not considered as it is beyond the scope of this paper and will be reported in subsequent work. This reported study is part of our long-running research efforts aimed at improving software architecture evaluation processes by reducing the time, resources and skills required to effectively and efficiently assess architecture variants with respect to desired quality attributes. A focal point of this work is to gain evidence-based understanding of different aspects of the activities involved in software architecture evaluation, especially on improving the development of scenario profiles to characterize desired quality attributes. Developing quality scenarios is not only one of the most important activities, but also considered to be the most expensive and time-consuming activity of architecture evaluation. The accuracy of the results of evaluation exercise largely depends on the quality of the scenarios used [11].

This paper reports the details of the experimental study, the findings, and the conclusions by closely following the guidelines for reporting controlled experiments [21]. The overall structure of the paper follows the standard practice of reporting empirical studies. Section II provides an overview of the related work and Section III introduces the research questions and hypotheses. Section IV summarizes main aspects of the experiment performed to gather the empirical data. Section V presents results of the data analysis followed by the discussion on the results in Section VI. Section VII concludes the paper with a summary of major contributions and future work.

II. RELATED WORK AND CONTEXT

This section summarizes relevant research from the perspectives of software architecture evaluation and software inspection.

A. SOFTWARE ARCHITECTURE EVALUATION

Quality attributes such as performance, security, usability, or modifiability can be supported or inhibited by the architecture of a software-intensive system [9]. Therefore, the evaluation of software architecture at an early stage has gained significant interest. Scenarios characterizing required quality attributes provide context for evaluating an architecture that works with concrete examples to enable users to understand their detailed effects [34]. Scenarios also help draw conclusions about the suitability of a proposed architecture and available alternatives. Many mature software architecture evaluation methods are scenario-based [12].

TABLE 1. An example scenario profile for the modifiability quality attribute.

Attribute factors	Scenario descriptions
Updating module	A client changes the format in which data is exchanged between the server system and the client's system.
COTS Replacement	The underlying database of the system is changed in a given time period and the system shall behave exactly as it did with the original database.
New functionality	The system needs to automatically notify individual customers and companies by email after an order has been completed.
	A new page needs to be added to the website for administration purposes.

The software architecture community has developed many frameworks for eliciting, structuring, and classifying scenarios, such as Lassing et al.'s two-dimensional framework for eliciting scenarios [33], Kazman et al.'s generic 3-dimensional matrix to elicit and document scenarios [28], and Bass et al.'s six-element framework to refine and structure scenarios [9]. Scenarios used in software architecture evaluation have been classified into various categories, such as direct scenarios, indirect scenarios, complex scenarios, use case scenarios, growth scenarios, and exploratory scenarios [9], [27], [35], [37]. The Software Engineering Institute (SEI) has enumerated a collection of general quality attribute scenarios intended to help characterize most commonly known quality attributes [6], [10]. A general scenario is, in effect, a template for generating a specific quality attribute scenario. A set of scenarios is called a scenario profile. Table 1 shows an example scenario profile of the modifiability quality attribute, which can serve as a general scenario profile to be concretized for a specific system such as a Web-based ticketing system.

Since not all general scenarios for a particular quality attribute may be relevant to a particular system or a class of systems, a system's stakeholders must identify relevant scenarios that should be made system-specific [36]. It has been claimed that general scenarios also help instigate thinking for system-specific scenarios called concrete scenarios [7]. However, general scenarios can also be classified according to domain-specific software change categories to help stakeholders develop those general scenarios that may be more relevant and should be made system-specific with the help of stakeholders for a particular system.

Most of the well-known scenario-based architecture evaluation methods require stakeholders to develop scenarios for characterizing quality attributes required by a system based on the business goals in face-to-face meetings [5]. Since a software architecture cannot be expected to fulfil the expectation of an unbounded list of scenarios, the stakeholders are also required to rank the generated scenarios in terms of their importance by assigning their respective votes to each of the generated scenarios. The prioritization of scenarios enables

the evaluation team to focus on the most important scenarios as the time and resources available for architecture evaluation are usually limited [18]. The importance of scenarios is defined by the stakeholders through placing their vote on each scenario or through counting the frequency of the occurrence of a scenario in different scenario profiles.

Despite the well-recognized importance of having high quality scenarios and the significant cost of having meetings for developing good quality scenarios, with some exceptions such as those reported in [11], there has been little research on determining the most effective way of gathering quality attribute scenarios from stakeholders. Based on a controlled experiment, Bengtsson and Bosch concluded that prepared teams (i.e., teams whose members first develop their own individual scenarios before joining a team to develop team scenarios) performed better than unprepared teams (i.e., teams whose members do not develop individual scenarios before generating team scenarios) and individuals (i.e., those who only develop their own individual scenario profiles without joining any team to develop team scenarios) in terms of the quality of scenario profiles developed [11]. Biffl et al. further reported on the impacts of experience and team size on the quality of scenario profiles developed for architecture evaluation [15].

However, there has been only little research on empirically studying and understanding the effectiveness of meetings for generating quality attributes scenarios (like QAW [7]) in terms of scenarios gained and lost during meetings. Considering the importance of scenarios in software architecture evaluation, and the cost, logistics, and scheduling difficulties involved in arranging evaluation meetings, we believe that the effectiveness of team meetings for generating scenarios is an important research issue. Hence, the research reported in this paper is motivated by the practical need to empirically determine the effectiveness of having meetings for generating quality attribute scenarios in terms of the number of scenarios gained and lost.

B. SOFTWARE INSPECTION TEAM MEETINGS

Similar to architecture evaluation, software inspection also aims at assessing the quality (in terms of finding a high number of relevant defects) of artifacts in a software development process. Some software inspection approaches also use scenarios to support finding quality issues [2], [31]. While the results of a software inspection process are defect reports, the architecture evaluation process steps, which 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 an architecture evaluation process with a special focus on team meeting performance.

Initially, Fagan's software inspection [19] viewed the inspection team meeting as the key process step, while more recent approaches starting with Parnas and Weiss [40] have focused more on individual inspection work to lower the overall 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, whereas in the team meeting only a limited number (usually two) of the inspectors can effectively interact at the same time while the others have to listen. Empirical results on the effectiveness of software inspection meetings differ considerably. Fagan reported inspection meetings to be very effective [20], [40], while other studies found contradictory results [14], [24], [41]–[43]. Nevertheless, software inspection research has identified several potential benefits of meetings.

First is *Synergy*. The synergy effect assumes that meeting dynamics help identify new defects. Johnson and Tjahjono [23], [24] observed a substantial degree of synergy (30% - 40% of new defects detected). However, Votta [43] reported that 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. [13] reported that meeting losses (i.e., real defects found during individual preparation but then dismissed in the meeting as irrelevant or false positives) significantly outweighed meeting gains (i.e., defects newly found during the team meeting). Especially defects reported by only one inspector during individual work were lost. Empirical studies by Biffl, Halling, and Grünbacher [16] confirmed Bianchi et al.'s findings. Second is *Identification of False Positives*. Land et al. [32] reported that team meetings had a clear advantage over individuals in discriminating between true defects and false positives. False positives, which report false defects, can become a problem if they occur frequently as they incur costs, e.g., time spent on trying to diagnose and repair them. Last is *Soft Benefits*. Apart from synergy and reduction of false positives, soft benefits are meeting benefits that include sharing of review experiences, dissemination of product-related knowledge, and creation of collective ownership for the review outcome among the reviewers [23], [24].

Unfortunately, in the inspection area the reduction of false positives and the soft benefits do not seem to justify the cost of meetings [14]. Similar to the concept of false positives, software architecture evaluation has a concept called irrelevant scenarios, which are considered not relevant to the system whose architecture is being evaluated, or not expected to be materialized in a foreseeable future (e.g., 3 to 5 years). Before evaluating an architecture variant, irrelevant scenarios should be identified and excluded from the process. The identification of irrelevant scenarios may be done through a prioritization process or based on the experiences of evaluators and/or stakeholders. Therefore, this paper focuses on the evaluation of the synergy effect of team meetings in the context of software architecture evaluation, particularly relevant architecture evaluation scenarios gained or lost in a team meeting.

III. RESEARCH CONTEXT, QUESTIONS AND HYPOTHESES

This section summarizes the research context, three major research questions and three derived hypotheses.

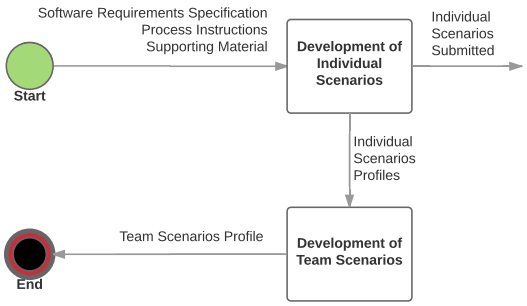

FIGURE 1. Two-step process of developing a scenario profile.

A. RESEARCH CONTEXT

The context for this study is a scenario development workshop such as QAW [7], where stakeholders develop scenarios to precisely specify quality attributes. These scenarios are used to assess the capability of a proposed architecture with regard to the desired quality attributes characterized by those scenarios [9].

The software architecture research community promotes the use of a two-stage process to develop quality attribute scenarios [11], where each participant first constructs an individual scenario profile and all individuals then come together in a team meeting to construct a team scenario profile, as depicted by Figure 1. A scenario development process usually follows two approaches: (1) the top-down approach in which stakeholders are provided with domain-specific software change categories to guide their development of scenarios, and (2) the bottom-up approach in which software change categories are not provided to stakeholders. Reports in literature [10], [33] suggest that the top-down approach can help stakeholders develop more relevant scenarios. It has been argued that the top-down approach can have most impact on the individual stage of the scenario development process. Scenarios produced by individuals who use the top-down approach should include a larger proportion of the most relevant changes than those produced by individuals who use the bottom-up approach [3].

However, there is a need to investigate whether the top-down approach can also help teams of stakeholders perform better in scenario development meetings. Therefore, this empirical study will investigate the impact of the scenario development approach (i.e., top-down vs. bottom-up) on both the individual and team performance. The individuals who use the bottom-up approach without specific guidance are in the control group, while those who use the top-up approach to develop scenarios guided by the given change categories are in the treatment group. For the purpose of evaluating the performance of individuals and teams, developed scenarios are assigned to one of the three classes: *A (critical)*, *B (important)* and *C (less important)* based on their scores determined by a marking scheme. Such a score is calculated using the number of times a scenario is reported by the stakeholders (individuals or teams), which is a typical approach used in architecture evaluation research.

To evaluate the performance of conducting team meetings against that of not conducting team meetings, we distinguish between real teams that conduct meetings and nominal teams that do not. The scenario profile of a nominal team is directly derived from the scenario profiles produced by the individual members in the team. We elect to make nominal teams of 3 persons because the performance of these teams is supposed to be compared against that of the real teams of 3 persons. Moreover, we do not consider the experiences of the team members while making up the nominal teams because we did not take that into account while making up the real teams who participated in the experiment.

B. RESEARCH QUESTIONS, VARIABLES, AND HYPOTHESES

This section details the research questions, studied variables, and hypotheses for the reported experiment.

1) RESEARCH QUESTIONS

There are many reports in software inspections that support the effectiveness of so-called reading techniques, which provide an inspector with a specific guidance in identifying defects, e.g., checklists, perspective-based or scenario-based reading techniques [8], [12], [13], [20], [24], [45]. By applying the concept of guiding stakeholders in developing quality scenarios, we intend to first explore the following two research questions:

RQ1: *Does the top-down approach help stakeholders develop significantly more critical (class A) scenarios* ***at individual level*** *than the bottom-up approach does?*

RQ2: *Does the top-down approach help stakeholders develop significantly more critical (class A) scenarios* ***at team level*** *than the bottom-up approach does?*

We conjecture that the top-down approach would enable individuals to develop more scenarios in general and more critical (class A) scenarios in particular as the change categories should ensure that a participant is unlikely to overlook an important category of changes. Similar to the impact of a top-down approach on the individual scenario development activity, we also expect the teams, whose individuals use the top-down approach, to focus on eliciting scenarios in the given change categories. The two hypotheses corresponding to **RQ1** and **RQ2** are **H1** and **H2** respectively:

- **H1:** The scenario development performance of an individual who uses the top-down approach during the development of scenario profile (treatment group) is significantly better than the individual who uses the bottom-up approach (control group).
- **H2:** The scenario development performance of the team who uses the top-down approach during the development of a team scenario profile (treatment group) is better than the team who uses the bottom-up approach (control group).

TABLE 2. Independent and dependent variables.

Research Question	Independent Variable		Dependent Variables	
	Variable	Value	Variables	Values
RQ1/RQ2	Scenario development approach	Top-down (treatment)	Frequency of each scenario in each class	Individual step
		Bottom-up (control)		Team step
RQ3	Effect of team meeting	Real meeting	Frequency of each scenario in each class	Real meeting
		Nominal meeting		Nominal meeting

The last, also the primary, research question in this work is the following:

RQ 3: *Does a real team meeting gain significantly more new scenarios than it loses existing scenarios from the individual development step?*

To justify the need for a real team meeting, it would be expected that the number of newly generated scenarios should exceed the number of abandoned existing scenarios developed by individuals as a result of the meeting. In a nominal meeting, no new scenario would be generated and no existing scenario would be abandoned. The corresponding hypothesis is **H3**:

- **H3**: The performance of a real team is significantly better than the performance of a nominal team.

For this research question, we follow the analysis procedure proposed by Bianchi et al. [13] for software inspection. However, we apply the approach to the scenario development activity of the software architecture evaluation and extend their analysis by assessing the impact of different scenario development approaches on the results. Since developing scenarios is more concerned with providing more scenarios [25] than with eliminating false positives as required in software inspection, we expect software architecture evaluation teams to only concentrate on adding new scenarios that should yield considerably more (if not more important) scenarios, while relatively little time would be spent on eliminating irrelevant scenarios.

2) INDEPENDENT AND DEPENDENT VARIABLES

Following the state of the practice in empirical research, Table 2 defines the independent and dependent variables for answering both research questions and for performance measurement of the collected data.

For **RQ1/RQ2**, the *independent variable* is the scenario development approach applied by the individuals and teams with the value being either top-down or bottom-up. A list of domain-specific categories of software changes were provided to the participants during the scenario development activity, with one treatment: change categories provided (top-down), and one control alternative: change categories not provided (bottom-up). The *independent variable* manipulated for **RQ3** is the effect of team meeting – the form of developing team scenarios with the value being either 3-person nominal meeting or 3-person real meeting.

The *dependent variables* are based on the frequency of each scenario in each of the 3 classes developed by the participants. For **RQ1**, the values include those at the individual development step and for **RQ2**, the values include those at the team development step. For **RQ3**, the values include those at the 3-person nominal meeting and those at the 3-person real meeting. Each scenario has been assigned to one of the three classes (A, B, or C) based on its score representing that scenario's relative importance, which is the number of times that scenario is identified in all scenario profiles (individuals as well as real teams). During a meeting, new scenarios may be developed and/or some existing scenarios from the 3 individual scenario profiles may be removed. A gained scenario is a scenario newly introduced during the meeting, which was not included in any of the individual scenario profiles of the team members. A lost scenario is a scenario found during the individual preparation phase by at least one team member, which however was not included in the team scenario profile probably because it was not accepted by the team during the meeting.

In particular, we consider the number of scenarios reported by an individual or a team in each of the three cases:

1) *Individual Performance:* number of scenarios reported by individuals in each of the three classes.
2) *Real-Team Performance:* number of scenarios reported in each of the three classes by a team of 3 individuals, who conduct a team meeting to develop the team scenario profile.
3) *Nominal-Team Performance:* number of scenarios reported in each of the three classes by a non-communicating team of 3 individuals. There is no team meeting but an editor combines the individual scenario profiles into a team scenario profile (no synergy effect or scenario gain/loss following the set union approach).

3) GLOSSARY OF TERMS

This section lists the terms used throughout the paper.

- *Scenario:* a case scenario used to evaluate a specific software quality attribute.
- *Scenario Profile:* a collection of scenarios from an individual or from a team.
- *Gained Scenarios:* new scenarios added to a team scenario profile.
- *Lost Scenarios:* existing scenarios identified by individuals that are not included in a team scenario profile.
- *Software Change Categories:* categories used to group software features that need to change.
- *Top-Down Approach:* stakeholders are provided with domain-specific software change categories to guide their development of scenarios.
- *Bottom-Up Approach:* software change categories are not provided to stakeholders.
- *Treatment Group:* the group of participants using the top-down approach.

- *Control Group:* the group of participants using the bottom-up approach.
- *Scenario Score:* the number of times the scenario is identified in all scenario profiles (individuals as well as real teams).
- *Scenario Class:* classification of a scenario based on its relative importance indicated by its score (A - Critical, B - Important, and C - less important).
- *Real Team:* a team that conducts team meetings.
- *Nominal Team:* a team that does not conduct team meetings.

IV. EXPERIMENT DESCRIPTION

This section describes the details about the experiment design, conduct, and threats to validity.

A. EXPERIMENTAL DESIGN

We decided to use a ''between-subject'' experimental design, considering it is more appropriate for the kind of study planned as it is unlikely to cause fatigue to the participants and it has very little risk of a carryover effect. For the advantages and disadvantages of this method, we refer the reader to Wohlin et al.'s work [46]. The experimental design was a randomized balanced design, which used the same experimental materials for all conditions (i.e., using top-down versus bottom-up approaches and in real versus nominal teams) and assigned the subjects randomly to each experimental condition using a card-sorting method of randomization. A total of 24 participants were involved in the study, who were divided into 4 teams of 3 persons, one for each of the experimental conditions.

Figure 1 in Section III-A depicts the basic experiment flow in two steps: (a) an individual work process for developing individual scenario profiles, and (b) a team meeting to develop a team scenario profile based on the inputs from the individual scenario profiles of the participants in each team. Each individual participant in the scenario development process uses system requirements, process instructions, and supporting material (e.g., guidelines for scenarios elicitation) to develop a scenario profile characterizing a required quality attribute. As the experiment is based on a case scenario that is intended to extend *LiveNet* to support a distributed software inspection process, the key quality attribute concerned is its modifiability.

For **RQ3**, we want to investigate: (a) the effectiveness of scenario development meetings in terms of the quantity of lost and gained scenarios, and (b) which scenarios are actually lost during meetings, a foundation to identify likely reasons for the loss of relevant scenarios. As far as the effectiveness of a scenario development meeting is concerned, we analyze the number of scenarios gained and lost during the meeting. For this analysis, we compare the performance of nominal and real teams. Real teams use the individual scenario profile of every team member as their input and develop an agreed team scenario profile as their output. The team scenario profile is developed in a team meeting based on the scenarios developed by each member of the team during the first stage through interaction and discussion. Nominal teams are non-communicating teams, where participants do not meet, but the individual scenario profiles are merged to one team scenario profile by an editor. An individual scenario is included in the nominal team scenario profile if it was reported by at least one team member; therefore, no reported scenario would be lost and no additional scenario would be gained.

B. EXPERIMENTAL LOGISTICS

This section includes experiment context, definition of participants, description of study materials, details on the software change categories, and the scheme for marking scenario profiles.

1) CONTEXT AND PARTICIPANTS

The 24 participants in the study were students enrolled in a software architecture course. The experiment was part of a scenario development workshop, which was one of the assessment tasks for the course. None of the authors was the lecturer in charge of the course. The students were briefed about the study's objective and procedure. Although participation in the workshop was compulsory, students had the option of withholding their results from research. Written permissions were sought from the participants to use their data in this study.

Most of the participants were postgraduate students with the exception of 3 fourth-year undergraduate students, who had maintained average marks at 75% or better (a requirement to enrol in this course for undergraduate students). 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.

2) TRAINING AND MATERIALS

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

One week before the study, all the participants received detailed information about the *LiveNet* system for which they were supposed to develop software change scenarios focusing on the modifiability quality attribute. One of the authors used the system to create a network of workspaces designed to support various activities of a software architecture evaluation process such as architecture presentation, scenario development and impact analysis. Each workspace

TABLE 3. Change categories for *LiveNet* pertaining to modifiability.

Category	Category Description
User Interface (UI)	Changes in the user interface of the application.
Security Policy (SP)	Changes needed for increased security of the application and its 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 for supporting different business processes.
Content Management (CM)	Changes needed to improve or add content management features.

had roles, artifacts and different collaborative features to provide the participants with an opportunity to familiarize with the system's collaborative features. A short document describing various features of the system was also provided a week before the study.

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

3) SOFTWARE CHANGE CATEGORIES

It has been claimed that a classification of software change categories can be used as a guide to help individual stakeholders come up with better quality scenarios [3]. 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 change categories used in our research, we drew upon in-depth domain knowledge in collaborative systems and the types of complex changes made over time in *LiveNet*. We followed an iterative process of building the scenario categorization scheme. Based on more than five years experience with *LiveNet* as researchers and users, we came up with a list of major change categories most likely required by a collaborative system like *LiveNet*. Then the compiled list was reviewed by the chief research investigator and software architect of *LiveNet*, both of them were involved in the project throughout its lifecycle. Based on their feedback, the list of change categories was refined. Table 3 presents a brief description of each of these categories relevant to the modifiability attribute:

4) SCHEME FOR MARKING SCENARIO PROFILES

In order to assess and compare the performance of the individuals in the control (using bottom-up approach) and treatment (using top-down approach) groups, and in the real and nominal teams, we needed a suitable marking scheme. Researchers in this line of research have used a marking scheme that ranks scenario profiles by comparing them with a reference scenario profile [4], [11]. However, we decided to come up with an additional marking scheme based on the score that reflects the frequency of each scenario's occurrence in all scenario profiles (i.e., in both individual and real teams).

As each individual scenario's score 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 implies a higher importance of a scenario. It is worth clarifying that this scheme only considers the frequency of scenarios; it does not factor in any quality aspect of scenarios through human voting/rating, which will be considered in future study though.

By using this marking scheme, we identified the importance of each scenario by counting the occurrences of each scenario in all individual and teams scenario profiles (32 total: 24 individual and 8 team profiles). The score given to each scenario was used to classify that particular scenario into one of the three classes reflecting the relative importance of a scenario. The overall score of a scenarios was calculated by the sum of the individual and real team scores. Scenarios were then ranked in a descending order of scores based on which the scenario classes were classified as follows.

1) *Class A (Critical)*: the top 20% of scenarios with the score greater than 30,
2) *Class B (Important)*: the following 40% of scenarios with the score between 4 and 30, and
3) *Class C (Less Important)*: the lowest 40% of scenarios with the score less than 4.

According to this marking scheme, each individual's or team's actual scenario profile must be re-coded into a standard format for analysis. This approach of assigning scenarios to different classes 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 is considered [11], which means the most relevant scenario will have the highest frequency of occurrences in all the scenario profiles created by individuals and real teams. This marking scheme is actually similar in essence to prioritizing scenarios by assigning votes that are allocated to stakeholders during a software architecture evaluation [18].

C. EXPERIMENT EXECUTION

The experiment was conducted as part of a scenario development workshop for a software architecture course as mentioned in Section IV-A. Figure 2 shows the major steps of the experiment execution. Prior to the experiment, the number of teams was identified based on the expected number of participants. Each team was assigned a name

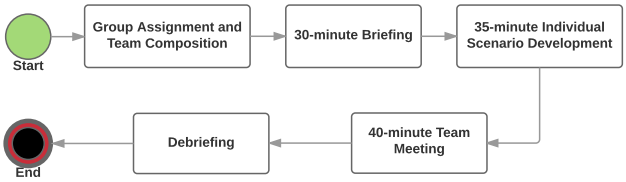

FIGURE 2. Experiment execution.

and 3 participants and allocated to one of the experimental conditions (i.e., using top-down versus bottom-up approaches and in real versus nominal teams). Assignment of participants to teams and allocation of teams to experimental conditions were based on a card-sorting 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 the *LiveNet* system. Participants were also given a document describing the collaborative system, architecture evaluation process and example scenarios.

After the briefing session, the participants were given a simplified version of *LiveNet*'s requirements specification. The participants in the treatment group also received a document describing the five categories (see Table 3) of the most commonly occurring changes in a collaborative system 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 their scenario development exercises. The participants followed the two-step process of developing individual and team scenarios outlined in Figure 1 of Section III-A.

The participants were first asked to develop software change scenarios individually for 35 minutes. After that, the individual scenario profiles were collected and photocopied before being returned to them. All the participants were then asked to join their respective teams to develop team scenarios for 40 minutes in a face-to-face meeting session following a simplified process of developing scenarios like QAW [7]. Each member of a team presented their individual scenarios for the group to discuss which of them should be included in the team scenario profile. The team members also discussed new scenarios to characterize the modifiability quality attribute. After that, we collected the team scenario profiles before running a debriefing session during which the participants filled a post-session questionnaire; however, the analysis of the data collected through the questionnaire is beyond the scope of this paper.

D. VALIDITY CONSIDERATIONS

Every empirical study has to deal with several threats to validity. In the following sub-sections, we provide a detailed discussion on only two (i.e., internal and external) of the four well-known types of validity threats (i.e., internal, external, construct, and conclusion) and the countermeasures we applied. For the construct validity, we carefully thought through our concepts of scenarios in the context of scenario development workshops with the objective of comparing the performance between treatment and control groups. The study was based on developed scenarios, which represent the outcome of a scenario generation activity of software architecture evaluation. In addition, our marking scheme for measuring the performance of different experimental conditions, which was based on the frequency of reported scenarios, is actually similar to the prioritization scheme in a scenario development activity. Furthermore, one of the authors, who is an expert in software architecture evaluation methods, led the effort on the operationalization of the concepts used in this study.

For the conclusion validity, we focused on improving the reliability by: (a) using a measurement scheme (i.e., marking scenario profiles) based on previously used schemes and well-known practices (i.e., prioritizing scenarios by assigning votes or raising hands), and (b) by ensuring good implementation of the experimental study under the direct supervision of one of the authors specialised in software architecture evaluation methods.

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 [46]. In order to avoid bias in allocating participants to different experimental conditions, we randomized the assignment by using a card-sorting method. We wrote the names of the participants and the groups on plain cards. After shuffling the cards, we assigned one card to each group (treatment or control) without seeing the individual's or the group's name on the card.

Another threat to the internal validity of our experiment is the appropriateness of the approach to classifying scenarios into three classes as an indication of their respective importance based on the frequency of occurrences of each scenario. This approach is similar to the method of measuring the quality of scenario profiles developed for architecture evaluation, which has been used in several studies [11], including ours. Moreover, we argue that deciding on the relative importance of each scenario based on its frequency is similar in essence to the commonly adopted approach to prioritizing quality attribute scenarios based on stakeholders' votes [29].

A further potential threat associated with this approach is the skill, knowledge, and bias of the person who recoded, removed duplication of, and assessed the semantic equivalence of scenarios in each scenario profile before counting the occurrences of each scenario. We addressed this issue by delegating two researchers to perform these tasks independently; any disagreement regarding the occurrences of each scenario in all profiles was resolved before taking the final count.

2) THREATS TO EXTERNAL VALIDITY

External validity is the degree to which the results can be generalized (i.e., transferable to 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 material and process are representatives of those used in industrial architecture evaluations.

Firstly, in an industrial evaluation, stakeholders may have a variety of 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 those with a non-technical background. It is worth mentioning that although all participants were technically competent, in the case of less competent participants, they would still be more competent than non-technical participants.

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 only apply to stakeholders without much experience of the application being evaluated. Consequently, if a different application other than *LiveNet* was used, the results would be comparable as long as the participants had no much experience with it.

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

Fourthly, the software requirements specification used in the experiment is relatively short and simple compared with a typical industrial one. As a result, a relatively small number of participants were involved in the experiment. However, in an industrial evaluation, stakeholders would usually be given both more requirements and more time to develop their scenarios. Despite the small sample size, the conclusions made through statistical analysis still have practical implications, although more experiments with large sample sizes would make the case stronger to generalize the findings.

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 in software architecture evaluation. In fact, the scenario development process in our experiment is similar to the one used in most of the scenario-based software architecture evaluation methods that gather scenarios to characterize quality attributes to be fulfilled by a proposed software architecture through workshops like QAW [7].

V. RESULTS

This section presents: (a) the data analysis procedure, (b) the results of individual scenario development processes in two groups (treatment and control), (c) the results of real team scenario development processes in two groups (treatment and control), and (d) the team effects in terms of scenarios gained and lost during team meetings.

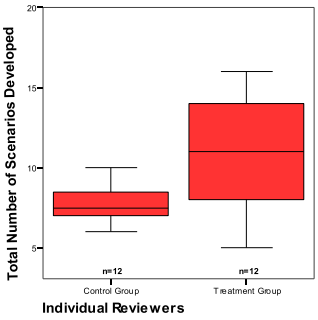

FIGURE 3. Number of scenarios per group of individuals.

A. DATA ANALYSIS PROCEDURE

The data analysis takes as inputs the scenarios developed by individuals and real teams during the experiment. Based on the individual and team scenario profiles, we calculated the frequency of each reported unique scenario in individuals' and real teams' scenario profiles. The frequency of each reported scenario is the baseline for placing a particular scenario into a certain class that represents its relative importance. As a preparation for statistical evaluation, the scenario descriptions were matched and divided into the three classes of: A (critical), B (important), and C (less important). Performance was measured from individuals, 3-person real teams, and 3-person nominal teams with respect to the three classes. We identified the number of scenarios based on the scenario profiles from individuals and real teams. For statistical analysis, we apply descriptive statistics, i.e., mean and 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 comprising 24 individual and 8 team profiles. Using the aforesaid classification scheme, we classified all the scenarios into relevant classes, as shown in Table 4.

TABLE 4. Scenario classification based on frequency.

Critical (A)		Important (B)		Less Important (C)		Total	
No.	%	No.	%	No.	%	No.	%
22	21.1%	41	39.4%	41	39.4%	104	100%

B. RQ1: INDIVIDUAL PERFORMANCE WITH REGARD TO SCENARIO DEVELOPMENT APPROACH

Figure 3 summarizes the results of individual participants placed in the control and treatment groups and Table 5 presents the details on these results. Individual participants report on average around 9.4 scenarios: 10.8 scenarios in the treatment group (using the top-down approach) and

TABLE 5. Number of scenarios per group of individuals.

	Mean	SD	p-value
Control Group	8.0	1.65	0.037(s)
Treatment Group	10.8	3.57	
All Participants	9.4	3.06	

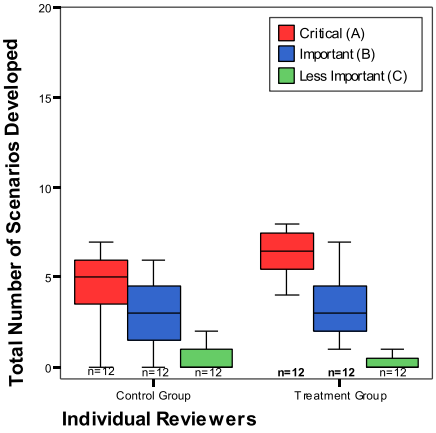

FIGURE 4. Number of individual scenarios per class and group.

8.0 in the control group (using the bottom-up approach). Applying the Mann-Whitney test, we observe a significant difference of the members in the treatment group over those in the control group (p-value = 0.037).

Figure 4 further compares the number of scenarios found in the three scenario classes between the treatment and control groups. Participants in the treatment group found overall more critical scenarios than those in the control group did. An interesting finding is that the number of identified important or less important scenarios in the treatment group is comparable to the corresponding one in the control group. For the critical scenarios, the participants in the treatment group reported significantly more scenarios than those in the control group (p-value = 0.015). In contrast, for the number of important or less important scenarios, we do not observe any significant difference. Table 6 summarizes the mean value and standard deviation as well as the Mann-Whitney test results.

Hypothesis **H1** is clearly accepted: the scenario development performance of an individual who uses the top-down approach during the development of scenario profile (treatment group) is significantly better than the individual

TABLE 6. Number of individual scenarios per class and group.

	Critical (A)		Important (B)		Less Important (C)	
	Mean	SD	Mean	SD	Mean	SD
Control Group	4.3	2.10	3.0	1.95	0.7	0.99
Treatment Group	6.8	2.29	3.5	2.02	0.4	0.90
p-value	0.015(s)		0.558(-)		0.406(-)	

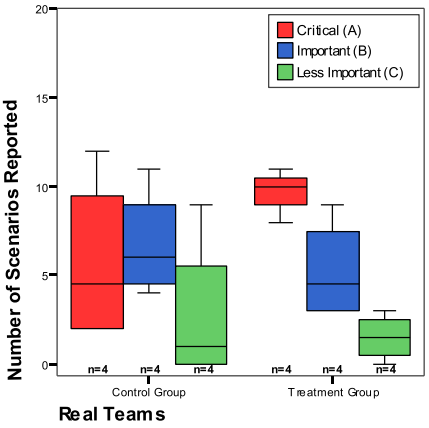

FIGURE 5. Number of real team scenarios per group and class.

who uses the bottom-up approach (control group). More specifically, individuals using the top-down approach produce significantly more critical (class A) scenarios than those using the bottom-up approach.

C. RQ2: TEAM PERFORMANCE WITH REGARD TO SCENARIO DEVELOPMENT APPROACH

To answer this research question, we will first investigate the real team performance, which is then followed by nominal team performance.

1) REAL TEAM PERFORMANCE

Real teams of 3 persons found after their meetings on average more than 15 scenarios, about 90% more than an average individual did. In detail, teams in the control group (using the bottom-up approach) identified 15.3 scenarios on average (SD = 4.99) and teams in the treatment group (using the top-down approach) found 16.5 scenarios on average (SD = 1.73). Teams in the treatment group shows a marginal advantage over those in the control group (on average more than 1 additional scenario found with a much smaller standard deviation), however we do not observe any significant difference (p-value = 0.554).

A more detailed analysis of different scenario classes shows that the guidance given to the treatment group results in notably more class A scenarios on average, but less class B and class C scenarios compared to the control group without the guidance. The treatment group also reveals less variance for class A scenarios, possibly due to the guidance provided through the change categories during the scenario development process. Figure 5 and Table 7 provide a deeper insight

TABLE 7. Number of real team scenarios per group and class.

	Critical (A)		Important (B)		Less Important (C)		Total	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Control Group	5.8	4.79	6.8	3.10	2.8	4.27	15.3	4.99
Treatment Group	9.8	1.26	5.3	2.87	1.5	1.29	16.5	1.73
p-value	0.243(-)		0.384(-)		0.882(-)		0.554(-)	

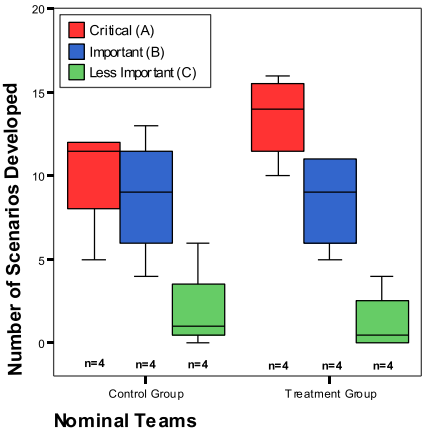

FIGURE 6. Number of nominal team scenarios per group and class.

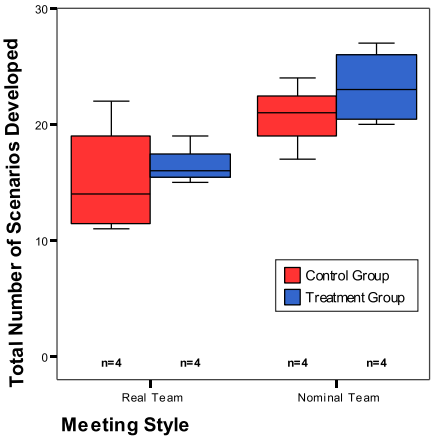

FIGURE 7. Number of team scenarios per group: Real vs. nominal teams (meeting style).

into the results of real teams regarding control/treatment groups and different scenario classes. However, we do not observe any significant difference between the control and treatment groups for any of the scenario classes.

2) NOMINAL TEAM PERFORMANCE

For the sake of fair comparison, for every real team, we used the same team members to build the corresponding nominal team and create the respective team scenario profile. As shown in Figure 6 and Table 8, the data analysis shows that nominal teams of 3 persons reported on average 20.8 (control group) and 23.3 (treatment group) scenarios, approximately 140% more than an average individual did. However, we do not observe any significant difference between the control and the treatment groups (p-value = 0.486). The treatment group found on average notably more class A scenarios, but less class B and class C scenarios than the control group did, which is similar to the finding with the real teams. A more detailed analysis reveals comparable results with respect to those of the real teams: there are advantages for critical scenarios but small disadvantages for important and less important scenarios; however we do not observe any significant difference between the control and the treatment groups.

In conclusion, hypothesis **H2** is clearly rejected: the scenario development performance of a real/nominal team who uses the top-down approach during the development of scenario profile (treatment group) is not significantly better than the team who uses the bottom-up approach (control group) in terms of the total number of developed scenarios or any of the scenario classes.

D. RQ3: TEAM PERFORMANCE WITH REGARD TO THE EFFECT OF TEAM MEETING

To answer this research question, we will first compare the performance of nominal teams with that of real teams, and then discuss scenario gain and loss with regard to the effect of team meeting.

1) PERFORMANCE COMPARISON BETWEEN NOMINAL AND REAL TEAMS

Figure 7 and Table 9 present the comparison of the group type (control vs. treatment group) and the meeting style (real vs. nominal teams). Real teams reported on average 15.9 scenarios, while nominal teams reported on average 22.0 scenarios, which is a very interesting finding as nominal teams appear to be more effective than the real teams in terms of the overall 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 team scenario profiles. While a 3-person real team reported 70% more scenarios than an individual did, it took about 100% more effort. In contrast, a nominal team reported 135% more scenarios than an individual did with the same

TABLE 8. Number of nominal team scenarios per group and class.

	Critical (A)		Important (B)		Less Important (C)		Total
	Mean	SD	Mean	SD	Mean	SD	Mean
Control Group	10.0	3.37	8.8	3.78	2.0	2.71	20.8
Treatment Group	13.5	2.65	8.5	3.00	1.3	1.89	23.3
p-value	0.146(-)		1.000(-)		0.544(-)		0.486(-)

TABLE 9. Number of team scenarios per group: Real vs. nominal teams (meeting style).

	Control Group		Treatment Group		All
	Mean	SD	Mean	SD	Mean
Real Teams	15.3	4.99	16.5	1.73	15.9
Nominal Teams	20.8	2.87	23.3	3.30	22.0
p-value	0.200(-)		0.029(s)		0.005(s)

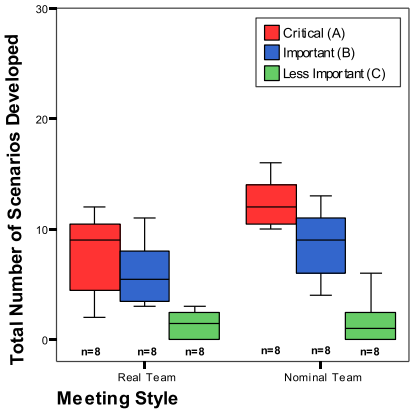

FIGURE 8. Number of scenarios per class and meeting style.

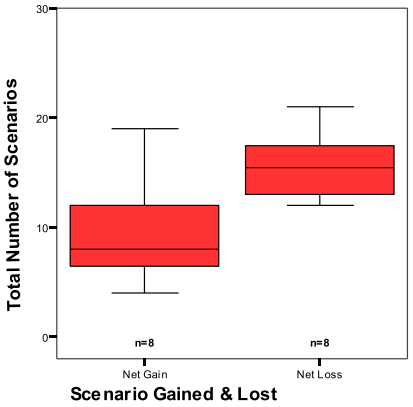

FIGURE 9. Number of scenarios gained/lost in a meeting.

amount of effort because the 3 members of the nominal team worked concurrently. We can observe a significant difference between the real and nominal teams in terms of the overall number of scenarios reported (p-value = 0.005).

A more detailed view on the impact of providing active guidance with change categories on the performance of teams (real vs. nominal) shows a significant advantage of nominal teams over real teams in the treatment group (p-value = 0.029). The control group also exhibits that advantage, but it is not significant.

Figure 8 and Table 10 present the results of scenario identification per class based on the frequency-based marking scheme. We observe a significant advantage of nominal teams over real teams in terms of the number of class A scenarios reported (p-value = 0.030). The class B also shows some advantage of nominal teams over real times, but it is not significant. For class C, the nominal teams identified less scenarios than the real teams did, which is again not significant. A possible reason might be the very low share of less important scenarios in the scenario profiles.

2) SCENARIO GAIN AND LOSS IN REAL TEAM MEETINGS

Scenario gain/loss during a software architecture evaluation meeting is an interesting question because the team meeting might help elicit additional scenarios (positive team effect) or hinder scenario elicitation (negative team effect). Thus, we want to compare the results from a real team with those from the corresponding nominal team. Figure 9 and Table 11 present the number (on average) of scenarios gained and lost in a meeting. We can observe a significant difference of the number of scenarios lost over the number of scenarios gained (p-value = 0.011).

Figure 10 and Table 12 present more detailed results of scenario gain/loss regarding the effect of providing guidance through change categories. It can be seen that: (a) more scenarios are lost than gained for both the control and the treatment groups; however only the treatment group shows a significant difference (p-value = 0.020), and (b) there is no significant difference between the control and the treatment groups regarding the number of gained/lost scenarios; we actually observe comparable results for both groups (e.g., on average 9.5 gained scenarios in both groups).

As the importance of scenarios can have a major impact on evaluating *LiveNet*'s software architecture, it is imperative to gain a deep insight into the effects of gained and lost scenarios

TABLE 10. Number of scenarios per class and meeting style.

	Critical (A)		Important (B)		Less Important (C)	
	Mean	SD	Mean	SD	Mean	SD
Real Teams	7.8	3.88	6.0	2.88	2.1	2.00
Nominal Teams	11.8	3.37	8.6	3.16	1.6	2.20
p-value	0.030(s)		0.102(-)		0.744(-)	

TABLE 11. Number of scenarios gained/lost in a meeting.

	Mean	SD
Scenarios Gained	9.5	4.72
Scenarios Lost	15.6	3.11
p-value	0.011(s)	

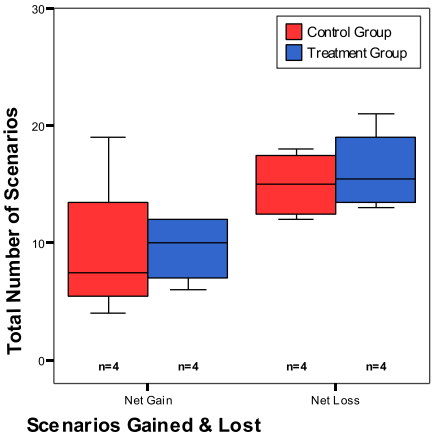

FIGURE 10. Number of scenarios gained/lost per group in a meeting.

TABLE 12. Number of scenarios gained/lost per group in a meeting.

	Control Group		Treatment Group	
	Mean	SD	Mean	SD
Scenarios Gained	9.5	6.56	9.5	3.00
Scenarios Lost	15.0	2.94	16.3	3.59
p-value	0.248(-)		0.020(s)	

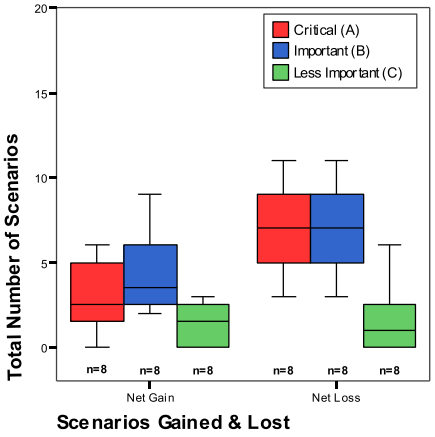

FIGURE 11. Number of scenarios gained/lost per class in a meeting.

regarding the scenario classes. Figure 11 and Table 13 present the results from the data analysis. We observe a significant difference of lost critical scenarios over gained critical scenarios (p-value = 0.008). The number of lost important scenarios is higher than the number of gained important scenarios, but it is not significant. For critical and important scenarios, on average 3 - 4 more scenarios are lost than gained. An interesting finding is that only for the less important class of scenarios, on average more scenarios are gained than lost, which again is not significant. These results seriously question the effectiveness of team meetings conducted in this experiment as the nominal teams identified more important (i.e., Classes A and B) scenarios than the real teams did. The real teams performed better than the nominal teams only for the less important scenarios.

In conclusion, hypothesis **H3** is clearly rejected: the performance of a real team is not significantly better than the performance of a nominal team both in terms of the number of scenarios developed and gained/lost as a result of the meeting. In fact, a nominal team significantly outperforms its corresponding real team in terms of the number of scenarios developed, specifically class A scenarios, and it does not incur any loss of scenarios.

TABLE 13. Number of scenarios gained/lost per class in a meeting.

	Critical (A)		Important (B)		Less Important (C)	
	Mean	SD	Mean	SD	Mean	SD
Scenarios Gained	3.0	2.14	4.4	2.51	2.1	3.00
Scenarios Lost	7.0	2.73	7.0	2.73	1.6	2.20
p-value	0.008(s)		0.064(-)		0.744(-)	

VI. DISCUSSION

This paper presents the results of data analysis to assess the effect of team meetings in terms of the number of scenarios gained and lost and the effect of the provision of software change categories to be used for guiding the scenario generation activity on the performance of individuals and teams (real and nominal teams). Based on our experiences in conducting scenario development workshops, previous empirical studies on this topic and empirical findings from knowledge acquisition and decision making disciplines, we postulated three research questions and three associated hypotheses.

The analysis of the experiment's data has enabled us to conclude that the provision of predefined change categories for guiding the generation of quality attribute scenarios has a positive impact on the activity at the individual level but not at the team level, accepting hypothesis **H1** but rejecting hypothesis **H2**. While the software architecture community has been using both top-down and bottom-up approaches to generating scenarios, there has been anecdotal evidence [10] that the provision of guidance in terms of seed scenarios can help generate better scenarios. Acceptance of hypothesis **H1** has clearly confirmed that, which will be added to evidence.

The findings from the data analysis performed to answer the research questions raised in this study provide further evidence that the provision of predefined category of scenarios can help generate significantly more scenarios at the individual level. Furthermore, we have also noted that the individuals in the treatment group reported particularly more critical scenarios than those in the control group did as the change categories should ensure that a participant is unlikely to overlook an important architecture quality category. Based on the data analysis, we assert that our research efforts have provided empirical evidence to the usefulness of "General Scenarios" [10] for improving the QAW [7] in which concrete quality attribute scenarios are generated because the "General Scenarios" are a kind of predefined category of scenarios that can guide the scenario generation activity.

In order to answer the primary research question, we intend to study the potential effect of scenario development meetings on the team performance by discovering whether the participants of scenario development meetings find more new scenarios than they lose existing ones compared to their individual work. Our analysis of the data for this study reveals that 3-person real teams reported only 50% - 150% more scenarios than an individual working alone. Moreover, real teams reported around 30% less class A and class B scenarios than nominal teams did, hence rejecting hypothesis **H3**. However, we find that the meeting loss is independent of the provision of the software change categories during the scenario generation activity. Our data analysis shows that more existing scenarios were lost than new scenarios were identified during team meetings. In particular, we observe a statistically significant loss for the critical scenarios. This

finding is similar to the experiences reported on software inspection teams. Rejection of both hypotheses **H2** and **H3** reveals that team scenario development process is actually focussed on synthesizing individual scenarios and eliminating so-called ''irrelevant'' scenarios rather than on developing new scenarios, something very similar to what usually happens in a team software inspection process [14].

Thus, we see considerable room for improvement in the process and tool support for software architecture evaluation meetings as proposed in [16] for software inspection. Moreover, these empirical findings challenge the anecdotal claims about the necessity and potential usefulness of having costly face-to-face meetings involving a large number of stakeholders for software architecture evaluation activities.

VII. CONCLUSIONS AND FURTHER WORK

The findings of this study have provided some very useful insights into significant aspects of meetings for developing scenarios during a software architecture evaluation process. This study has also provided an empirical evidence to support some assumptions and experiences in designing and conducting quality attribute scenario workshops. In the reported experimental context, software architecture evaluation meetings are not effective for scenario generation compared to the simple collection and union of the results from individual scenario profiles. Therefore, our findings question the effectiveness of workshop-style meetings involving a large number of stakeholders to develop scenarios as required by most of the scenario-based software architecture evaluation methods.

We suggest that there is an important need of devising and evaluating the usefulness and effectiveness of alternative ways of supporting software architecture evaluation processes, probably without long sessions of face-to-face meetings, in order to maximize the effectiveness of scenario generation activity. We also emphasize the importance of introducing new tools with the goal to keep scenario gain while avoiding scenario loss. For example, participants of scenario generation workshops like QAWs can benefit from the two phases of generating scenarios with a break in between during which the stakeholders can use asynchronous communication tools to get better understandings of the scenarios generated by others. The stakeholders can be asked to provide a brief rationale for justifying the importance and relevance of each of the scenarios with respect to the business goals of a system. A clear and well-documented rationale can help make a good justification for the inclusion of a scenario in a team scenario profile during team meetings.

Different scenario elicitation approaches (top-down or bottom-up) applied during individual preparation does not significantly influence the meeting performance in terms of scenarios gained and lost. Actually this result is quite interesting, especially considering that a high portion of overlapping scenarios could be expected among participants who follow a top-down approach. Thus, teams using a top-down approach could lose more scenarios than those using a bottom-up approach. However, we do not observe this effect in this study probably because architecture evaluation scenarios are more diverse than defects reported in software inspections.

We are conscious about the limitations of this work. First, the findings are based on one empirical study and replication of the study design by independent research groups is required to strengthen the validity of the empirical evidence. For this purpose, the authors welcome any request for obtaining a complete replication package. Second, the effectiveness of holding software architecture evaluation meetings is measured by the quantity of scenarios gained and lost during the meetings. Further study needs to take the quality of scenarios into consideration. Third, the marking scheme is based on the frequency of each reported scenario. Future work will also need to incorporate human prioritization through voting/rating. Last but not least, for further work in the area of software architecture evaluation meetings in general and scenario development meetings in particular, we would like to reiterate our emphasis on introducing 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 software architecture evaluation meetings. We intend to carry out further empirical research to thoroughly study and fully understand the implications of the proposed solutions.

ACKNOWLEDGMENTS

The authors are grateful to the participants of the experiment whose data have been analyzed to answer the research questions. We are also thankful to our colleagues for helping us run the experiment and to the reviewers for their constructive suggestions on improving the research work and the article.

REFERENCES

[1] S. Abrahão and E. Insfran, ''Evaluating software architecture evaluation methods: An internal replication,'' in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng. (EASE)*, 2017, pp. 144–153.
[2] Ö. Albayrak and J. C. Carver, ''Investigation of individual factors impacting the effectiveness of requirements inspections: A replicated experiment,'' *Empirical Softw. Eng.*, vol. 19, no. 1, pp. 241–266, 2014.
[3] M. A. Babar and S. Biffl, ''Eliciting better quality architecture evaluation scenarios: A controlled experiment on top-down vs. bottom-up,'' in *Proc. Int. Symp. Empirical Softw. Eng.*, 2006, pp. 307–315.
[4] M. Ali Babar, B. Kitchenham, and R. Jeffery, ''Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment,'' *Empirical Softw. Eng.*, vol. 13, no. 1, pp. 39–62, 2008.
[5] M. Ali Babar, L. Zhu, and R. Jeffery, ''A framework for classifying and comparing software architecture evaluation methods,'' in *Proc. 15th Australas. Softw. Eng. Conf.*, Apr. 2004, pp. 309–319.
[6] M. Anvaari, C.-F. Sørensen, and O. Zimmermann, ''Associating architectural issues with quality attributes: A survey on expert agreement,'' in *Proc. 10th Eur. Conf. Softw. Archit. Workshops (ECSAW)*, 2016, p. 11.
[7] M. R. Barbacci, R. Ellison, A. J. Lattanze, J. A. Stafford, C. B. Weinstock, and W. G. Wood, ''Quality attribute workshops (QAWs),'' Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2003-TR-016, 2003.
[8] V. R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M. V. Zelkowitz, ''The empirical investigation of perspective-based reading,'' *Empirical Softw. Eng. J.*, vol. 1, no. 2, pp. 133–164, 1996.

[9] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Reading, MA, USA: Addison-Wesley, 2003.
[10] L. Bass, M. Klein, and G. Moreno, "Applicability of general scenarios to the architecture tradeoff analysis method," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2000-TR-014, Oct. 2001.
[11] P. Bengtsson and J. Bosch, "An experiment on creating scenario profiles for software change," *Ann. Softw. Eng.*, vol. 9, pp. 59–78, May 2000.
[12] A. Bertolino, P. Inverardi, and H. Muccini, "Software architecture-based analysis and testing: A look into achievements and future challenges," *Computing*, vol. 95, pp. 633–648, Aug. 2013.
[13] A. Bianchi, F. Lanubile, and G. Visaggio, "A controlled experiment to assess the effectiveness of inspection meetings," in *Proc. 7th Int. Softw. Metrics Symp.*, Apr. 2001, pp. 42–50.
[14] S. Biffl and M. Halling, "Investigating the defect detection effectiveness and cost benefit of nominal inspection teams," *IEEE Trans. Softw. Eng.*, vol. 29, no. 5, pp. 385–397, May 2003.
[15] S. Biffl, D. Winkler, and M. A. Babar, "Impact of experience and team size on the quality of scenarios for architecture evaluation," in *Proc. 12th Int. Conf. Eval. Assessment Softw. Eng.*, 2008, pp. 1–10.
[16] S. Biffl, P. Grunbacher, and M. Halling, "A family of experiments to investigate the effects of groupware for software inspection," *Automat. Softw. Eng.*, vol. 13, no. 3, pp. 373–394, 2006.
[17] R. P. Biuk-Aghai and I. T. Hawryszkiewycz, "Analysis of virtual workspaces," in *Proc. Int. Symp. Database Appl. Non-Traditional Environ.*, 1999, pp. 325–332.
[18] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Reading, MA, USA: Addison-Wesley, 2002.
[19] M. E. Fagan, "Design and code inspections to reduce errors in program development," *IBM Syst. J.*, vol. 15, no. 3, pp. 182–211, 1976.
[20] T. Gilb and D. Graham, *Software Inspection*. Reading, MA, USA: Addison-Wesley, 1993.
[21] A. Jedlitschka and D. Pfahl, "Reporting guidelines for controlled experiments in software engineering," in *Proc. Int. Symp. Empirical Softw. Eng.*, 2005, pp. 95–104.
[22] B. E. John and L. Bass, "Usability and software architecture," *Behav. Inf. Technol.*, vol. 20, no. 5, pp. 329–338, 2001.
[23] P. Johnson and D. Tjahjono, "Does every inspection really need a meeting?" *Empirical Softw. Eng. J.*, vol. 3, no. 1, pp. 9–35, 1998.
[24] P. Johnson and D. Tjahjono, "Assessing software review meetings: A controlled experimental study using CSRS," in *Proc. 19th Int. Conf. Softw. Eng.*, 1997, pp. 118–127.
[25] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," *IEEE Softw.*, vol. 13, no. 6, pp. 47–55, Nov. 1996.
[26] R. Kazman, M. Barbacci, M. Klein, and S. J. Carriere, "Experience with performing architecture tradeoff analysis," in *Proc. 21st Int. Conf. Softw. Eng.*, May 1999, pp. 54–63.
[27] R. Kazman, L. Bass, G. Abowd, and M. Webb, "SAAM: A method for analyzing the properties of software architectures," in *Proc. 16th Int. Conf. Softw. Eng.*, May 1994, pp. 81–90.
[28] R. Kazman, S. J. Carrière, and S. G. Woods, "Toward a discipline of scenario-based architectural engineering," *Ann. Softw. Eng.*, vol. 9, nos. 1–2, pp. 5–33, 2000.
[29] B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, F. Kurniawati, M. Staples, H. Zhang, and L. Zhu, "Evaluating guidelines for reporting empirical software engineering studies," *Empirical Softw. Eng. J.*, vol. 13, no. 1, pp. 219–221, 2008.
[30] J. Knodel and M. Naab, "Software architecture evaluation in practice: Retrospective on more than 50 architecture evaluations in industry," in *Proc. IEEE/IFIP Conf. Softw. Archit. (WICSA)*, Apr. 2014, pp. 115–124.
[31] O. Laitenberger and J. DeBaud, "An encompassing life cycle centric survey of software inspection," *J. Syst. Softw.*, vol. 50, no. 1, pp. 5–31, 2000.
[32] L. P. W. Land, R. Jeffery, and C. Sauer, "Validating the defect detection performance advantage of group designs for software reviews: Report of a laboratory experiment using program code," in *Proc. 6th Eur. Softw. Eng. Conf. 5th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 1997, pp. 294–309.
[33] N. Lassing, P. Bengtsson, J. Bosch, and H. V. Vliet, "Experiences with ALMA: Architecture-level modifiability analysis," *J. Syst. Softw.*, vol. 61, no. 1, pp. 47–57, 2002.
[34] N. Lassing, D. Rijsenbrij, and H. van Vliet, "How well can we predict changes at architecture design time?" *J. Syst. Softw.*, vol. 65, no. 2, pp. 141–153, 2003.
[35] N. Lassing, D. Rijsenbrij, and H. V. Vliet, "On software architecture analysis of flexibility, complexity of changes: Size isn't everything," in *Proc. 2nd Nordic Softw. Archit. Workshop*, 1999, pp. 1–6.
[36] A. Liu, L. Bass, and M. Klein, "Analyzing enterprise javabeans systems using quality attribute design," Softw. Eng. Inst., Carnegie Mellon Univ., Tech. Rep. CMU/SEI-2001-TN-025, Oct. 2001.
[37] J. Miller, M. Wood, and M. Roper, "Further experiences with scenarios and checklists," *Empirical Softw. Eng. J.*, vol. 3, no. 1, pp. 37–64, 1998.
[38] S. Misra, L. Fernández, and R. Colomo-Palacios, "A simplified model for software inspection," *J. Softw., Evol. Process*, vol. 26, no. 12, pp. 1297–1315, 2014.
[39] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Commun. ACM*, vol. 15, no. 12, pp. 1053–1058, 1972.
[40] D. L. Parnas and D. M. Weiss, "Active design reviews: Principles and practices," *J. Syst. Softw.*, vol. 7, no. 4, pp. 259–265, 1987.
[41] A. A. Porter and P. M. Johnson, "Assessing software review meetings: Results of a comparative analysis of two experimental studies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 3, pp. 129–145, Mar. 1997.
[42] C. B. Seaman and V. R. Basili, "Communication and organization: An empirical study of discussion in inspection meetings," *IEEE Trans. Softw. Eng.*, vol. 24, no. 7, pp. 559–572, Jul. 1998.
[43] L. G. Votta, Jr., "Does every inspection need a meeting?" *ACM Softw. Eng. Notes*, vol. 18, no. 5, pp. 107–114, 1993.
[44] L. G. Williams and C. U. Smith, "PASA: A method for the performance assessment of software architecture," in *Proc. 3rd Workshop Softw. Perform.*, 2002, pp. 179–189.
[45] D. Winkler, S. Biffl, and B. Thurnher, "Investigating the impact of active guidance on design inspection," in *Proc. 6th Int. Conf. Product Focused Softw. Process Improvement*, 2005, pp. 458–473.
[46] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer, 2000.

M. ALI BABAR received the Ph.D. degree from the University of New South Wales, Australia. He is a Professor of software engineering with the School of Computer Science, University of Adelaide, Australia. His research is to develop and/or rigorously evaluate approaches and tools for supporting the design, analysis, and evolution of complex and dependable software intensive system and services that meet both the functional and non-functional requirements as derived from the quality goals specified by the stakeholders.

HAIFENG SHEN received the Ph.D. degree from Griffith University, Australia. He is an Associate Professor and the Discipline Leader of Information Technology with Peter Faber Business School, Australian Catholic University, Australia. His research is on human-centred software and systems including human computer interaction, software engineering, ubiquitous computing, mobile devices and applications, and social and collaborative computing.

STEFAN BIFFL received the Ph.D. degree from the Vienna University of Technology, Austria. He is an Associate Professor of software engineering with the Institute of Information Systems Engineering, Technische Universität Wien. His research interests include software and systems engineering process improvement, model quality assurance, value-based software engineering, empirical software engineering, collaboration in software and systems engineering, and software quality management and software engineering process improvement.

DIETMAR WINKLER received the Ph.D. degree from the Vienna University of Technology, Austria. He is a Senior Researcher with the Research Group for Quality Software Engineering, Institute for Information Systems Engineering, Vienna University of Technology. His research interests include software/systems engineering, software processes and software process improvement, empirical and value-based software engineering, and model-driven software and systems development.