The Value of Architecturally Significant Information Extracted from Patterns for Architecture Evaluation: A Controlled Experiment

Muhammad Ali Babar, Barbara Kitchenham, Piyush Maheshwari
National ICT Australia Ltd. and University of New South Wales, Australia
{malibaba, barbara.kitchenham, piyush.maheshwari}@nicta.com.au

Abstract

We have developed an approach to identify and capture architecturally significant information from patterns (ASIP), which can be used to improve architecture design and evaluation. Our experimental goal was to evaluate whether the use of the ASIP improves the quality of scenarios developed to evaluate software architecture. Out of 24 subjects 21 were experienced software engineers who had returned to University for a postgraduate studies and remaining 3 were fourth year undergraduate students. All participants were taking a course in software architecture. The participants were randomly assigned to two groups of equal size. Both groups developed scenarios for architecture evaluation. One group (treatment group) was given ASIP information the other (control group) was not. The outcome variable was the quality of the scenarios produced by each participant working individually. The treatment group participants also completed a post-experiment questionnaire. Our results support the hypothesis that ASIP information assists scenario development in the context of architecture evaluation.

1. Introduction

Software Architecture (SA) evaluation is considered as an effective approach for discovering architectural risks and questionable design decisions early in the software development lifecycle [1]. The architecture community has developed several evaluation techniques and methods, including simulation, checklists and scenarios. Most of the well-known architecture evaluation methods are scenario-based [2], such as the Architecture Tradeoff Analysis Method (ATAM) [1], Performance Assessment of Software Architecture (PASA) [3] and Architecture-Level Maintainability Analysis (ALMA) [4].

Gathering quality-sensitive scenarios is one of the most important activities of scenario-based architecture evaluation methods [5]. The accuracy of the results of architecture evaluation using these methods is largely dependent on the quality (i.e. coverage, relevance, and concreteness) of the scenarios used for evaluating software architecture [6]. A diverse set of stakeholders (such as architects, developers, users, sponsors and others) generally brainstorm scenarios to specify quality attributes required of a system in quality attribute workshops [7]. These approaches to gathering scenarios can be time consuming for participants and may not deliver required coverage of the scenario sets, which may result in the possible sensitivity problem of evaluation methods [8]. Thus, there is need to develop and assess methods to improve the scenario generation activity of architecture evaluation process.

One of the approaches to improving the scenario generation activity is to provide stakeholders with a list of general scenarios that may help them generate concrete scenarios. These general scenarios are usually collected from various sources such as problem domain, experienced stakeholders, previous architecture evaluation exercises, and others. We have found that software patterns'1 documentation implicitly describes general scenarios, which characterize the quality attributes affected by a particular pattern. We have been distilling general scenarios and other architectural artifacts from patterns to support architecture design and evaluation process [9, 10].

The first author of this paper developed a pattern-mining process for extracting general scenarios, quality attributes, and tactics from patterns and recording them in a standard template, which presents these pieces of architecturally significant information in a specified format [11]. We refer to this structured information as Architecturally Significant Information extracted from Patterns or ASIP. In this paper we report a controlled experiment that we designed and ran to assess whether providing stakeholders with the ASIP can help them to develop better quality scenarios.

The reported experiment is a part of a research program that consists of an observational study [10] and two controlled experiments. The research program

---

1 By software patterns we mean architecture and design patterns.
is aimed at evaluating the effectiveness of the pattern-mining process and utility of the ASIP.

2. Background and Motivation

A quality attribute is a non-functional requirement of a software system, e.g., security, flexibility, changeability, portability, and so forth. According to [12], software quality is the degree to which the software possesses a desired combination of attributes. It is widely recognized that SA of large systems put constraints on the achievement of desired set of quality attributes such as variability, flexibility and others [13]. Since SA plays a vital role in achieving quality attributes, it is important to focus on quality related issues during architecture design and to assess the capability of the designed architecture with respect to the desired quality attributes as early as possible.

A scenario is a textual, system independent specification of a quality attribute [13]. Scenarios are considered an effective and useful mechanism for specifying quality attributes. Several approaches use scenarios to encourage disciplined thinking during architecture design and evaluation activities [14, 15]. Scenarios are very flexible. They can be used for systematically reasoning about or evaluating most quality attributes. For example, we can use scenarios that represent failure to reason about the appropriate architectural mechanisms to satisfy availability and reliability requirements; scenarios that represent change requests can be used to examine modifiability; scenarios that represent threats can help select suitable security tactics; or scenarios that represent ease of use can support usability assessment.

A pattern is a known solution to a recurring problem in a particular context [16]. The software architecture of complex and large system is usually developed using several different patterns. Architectures of such systems evolve by successively integrating different patterns that may be described at different levels of abstraction, to deal with particular design problems [17]. One of the main goals of using patterns is to design an architecture with known quality attributes [18]. Each pattern either supports or inhibits one or more quality attributes.

There are several formats for documenting patterns. However, most pattern documentation formats include at least four sections: context, problem, solution and quality consequences. Each pattern’s description contains a large amount of architectural information, e.g., scenarios, quality attributes supported or hindered, tactics, and others. The scenarios informally embedded in a pattern description can be used to characterize the quality attributes achieved by that pattern.

There has been some effort to make these relationships explicit and codify this knowledge [19, 20]. For example, Bass and John associate architectural patterns with usability through scenarios by generating scenarios that characterize usability and presenting architecture patterns that can satisfy those scenarios [19]. Folmer and Bosch [20] also link usability with patterns by decomposing usability attributes into usability properties, which can be characterized by scenarios but they do not provide such scenarios. However, these approaches do not attempt to systematically capture and document the relationships among scenarios, quality attributes, and patterns informally described in pattern documentation. We claim that a disciplined approach to extract and document such relationship from patterns can help software engineers to fully utilize the large amount of architecturally significant information implicitly described in patterns’ documentation.

3. “Pattern Mining” and “ASIP”

The idea of exploring patterns for architecturally significant information arose from our research intended to improve architecture evaluation activities. We found that software patterns are a valuable source of general scenarios, which can be extracted to improve the architecture evaluation process [21]. We later found that each pattern’s description is also an important source of architecturally significant relationships that exist among scenarios, quality attributes, and patterns. We argued that such synergistic relationships should be systematically captured and documented as reusable artifacts to support and improve different activities of architecture design and evaluation process [11].

Initial “pattern-mining” efforts revealed that it is a labour intensive exercise, which relies on the pattern miner’s experience with different classes of patterns (such as architectural patterns [18], design patterns [16], and others) and with the different formats of pattern documentation. Moreover, the extracted information needs to be documented in a way that explicates the relationships among architectural artifacts. We have developed a framework consisting of a process model, guidelines and a set of templates for supporting the pattern-mining process [11].

Different components of the framework support various tasks of capturing and documenting architecturally significant information from patterns. This paper discusses only one component of the framework, a template and information contained with it, which make up the ASIP.
Presentation processing requires complex view navigation, creation and delivery....

Context

Presentation processing requires complex view navigation, creation and delivery....

Problem description

A centralized access point is required for presentation request handling to support the integration of system services. Each view provides its own system services, which results in duplicate code....

Suggested solution

Use a Controller as the initial point of contact for handling a request for services like authentication and authorization, delegating business processing, handling errors.....

Forces

Logic that is best handled in one location is replicated within numerous Views. Decision points exist for retrieving and manipulating data. A user has direct access to each service, which causes complexity.

Available tactics

Servlet front, JSP front, Command and controller, Physical/ logical/Multiplex resource mapping....

Affected Attributes

<table>
<thead>
<tr>
<th>Pattern Name: Front Controller</th>
<th>Pattern Type: Design pattern</th>
</tr>
</thead>
</table>

| Brief description | This pattern holds the common processing logic within the presentation tier. It handles requests and manages content retrieval, security, and navigation, delegating to a Dispatcher...... |
| Context | Presentation processing requires complex view navigation, creation and delivery.... |
| Problem description | A centralized access point is required for presentation request handling to support the integration of system services. Each view provides its own system services, which results in duplicate code.... |
| Suggested solution | Use a Controller as the initial point of contact for handling a request for services like authentication and authorization, delegating business processing, handling errors..... |
| Forces | Logic that is best handled in one location is replicated within numerous Views. Decision points exist for retrieving and manipulating data. A user has direct access to each service, which causes complexity. |
| Available tactics | Servlet front, JSP front, Command and controller, Physical/ logical/Multiplex resource mapping.... |

<table>
<thead>
<tr>
<th>Affected Attributes</th>
<th>Positively</th>
<th>Negatively</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusability, Manageability of security, centralized services.</td>
<td>Reduced efficiency, Another level of indirection.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>General scenarios</th>
<th>S1</th>
<th>Minimize code duplication for system services like data verification, user authentication and others.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S2</td>
<td>System services are managed by a centralized control mechanism, which may invoke them as needed.</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>Introduce a contact point for handling all requests before getting them for processed by core services.</td>
</tr>
</tbody>
</table>

| Usage examples | Logging users’ actions, keep tracking user’s shopping carts |

Table 1 shows the ASIP template that not only helps structure and document the extracted information but also guides during the mining process by identifying the pieces of information that needs to be extracted.

Compared with standard pattern documentation, the ASIP template captures and presents different pieces of a pattern’s description in a succinct format at an abstraction level suitable for architecture processes, where general scenarios are used to characterize required quality attributes and suitable patterns are chosen based on their support for the required quality attributes. Linking pattern, quality attribute, and general scenarios forms a certain type of architecture design knowledge, which should be represented in a manner that improves the usability of such knowledge.

Table 1 demonstrates the ASIP template for representing the ASIP extracted from one of the core J2EE patterns, Front Controller pattern [22], in a manner that makes the relationships among scenarios, quality attributes, and patterns explicit.

Based on our experiences with the ASIP, we believe that instead of reading several pages of pattern documentation to comprehend a pattern’s rationale, the forces section of the ASIP template presents a pattern’s rationale in a precise manner that is easy to understand. The abstract information captured in the ASIP template can be concretized for a specific project. For example, the scenarios extracted from the Front Controller pattern can be concretized to specify modifiability or reusability quality requirements of a project. For instance, the scenario S1 in the template “Minimize code duplication for system services like data verification, user authentication and others.” can be concretized for specifying easily modifiable security policies in this way: “Application should have a centralized place to implement organization’s security policies, which can be modified in two days of one person’s efforts.” From another aspect, assume that this concrete scenario is used to specify the quality attribute of an architecture and that architecture applies the Front Controller pattern, since this concrete scenario is an instance of a general scenario extracted from the Front Controller pattern used in that architecture, this increases an architect’s confidence in that architecture’s ability to satisfy the above-described concrete scenario [10].

We suggest that the ASIP (see Table 1) can be useful during SA design and evaluation in two ways:

1. The ASIP can improve the architecture evaluation process by helping stakeholders develop concrete scenarios for an application based on the general scenarios extracted from the patterns used in the architecture of that application2. The stakeholders do not need to generate scenarios from scratch for every architecture evaluation session. In addition, the general scenarios extracted from patterns can stimulate stakeholders’ thinking resulting in improved quality of concrete scenarios.

Table 1: Architecturally significant information extracted from J2EE Front Controller pattern

---

2 One weakness of the ASIP may be that the stakeholders may develop only those scenarios that would have already been considered by using certain pattern. We will discuss this issue in the discussion section.
The ASIP can help architects/designers identify appropriate patterns for system design by explicating the synergistic relationships among patterns, scenarios and quality attributes and presenting pattern related architecture knowledge in a succinct format at an abstraction level suitable for architecting process.

4. Description of the Experiment

4.1 Research Hypotheses

Informally, our experimental hypothesis is that stakeholders responsible for developing scenarios will be helped by the provision of the ASIP. The context for the experiment is an architecture evaluation task where stakeholders will be developing concrete quality sensitive scenarios that the architecture is intended to address. In this context, appropriate ASIP is information relating to those patterns that are used in the architecture being evaluated.

Like [6], we use a two-stage scenario development process. Firstly, each individual constructs scenarios alone. Secondly, individuals come together in groups to construct a joint set of scenarios. We expect ASIP to have most impact on the first stage of the process when individuals initially start to think about constructing scenarios. Scenarios produced by individuals given appropriate ASIP should include a larger proportion of the most significant issues than scenarios produced by individuals who are not given ASIP. Our proposed outcome measure of scenario quality is designed to capture this aspect of scenario quality.

In the second stage of scenario development (i.e. the group stage), the effect of ASIP may help group members understand each other’s scenarios and come to more informed group decisions about the value of scenarios. However, we have no specific hypothesis concerned with the group activity.

More formally, our null hypothesis is:

H0: The provision of ASIP will not improve the quality of quality-sensitive scenarios prepared for architectural evaluation at an individual level.

Our alternative hypothesis is:

H1: The provision of ASIP will improve the quality of quality-sensitive scenarios produced by individuals. Note this is a one-sided hypothesis.

If ASIP is useful, we would expect to reject H0.

4.2 Experiment Design

The experiment design was a randomized design, which used the same experimental materials for both treatments and assigned the subjects randomly to each treatment. Our experiment design was also a balanced design, i.e. the same number of subjects was assigned to each treatment [23]. Randomization was done by using a sort card method of randomization.

<table>
<thead>
<tr>
<th>Treatments Material</th>
<th>ASIP provided</th>
<th>ASIP not provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiveNet system</td>
<td>12 individuals</td>
<td>12 individuals</td>
</tr>
</tbody>
</table>

Figure 2: Allocation of the participants to treatment and control conditions

Figure 2 shows the experimental design. The participants were asked to construct software change scenarios for the LiveNet system. They were given the software requirements specifications (SRS) of the system, which was well-known to them. Because scenario profiles for architectural evaluation need to be concrete, we decided to select only one quality attribute, modifiability, for developing scenario profiles. However, we believe that the experiment design allows the results to be applicable to selected scenario profiles for other quality attributes as well [6]. The Independent variable manipulated by this study is the information provided to the participants during scenario development activity, with one treatment: ASIP provided, and one control: ASIP not provided. The Dependent variable is quality of scenario profiles developed by the participants with and without ASIP both working individually and in three-person groups.

4.3 Participants

The 24 participants in the study were students of software architecture course offered at the University of New South Wales, Australia. Since this study was incorporated in the syllabus of the course, most of the tasks of the study were part of the course assessment. The students were briefed about the objectives and procedures of the study. They had the option of withholding their results from the study. Written permission was sought from the student to use their data in this study.

Most of the students were post-graduate students with the exception of a few (3) fourth year undergraduate 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 an average age of 27 years. Their working experience typically had a good mix of design, coding, test, maintenance, and technical support activities.
Table 2: Participants’ level of expertise in different software architecture related knowledge

<table>
<thead>
<tr>
<th>Level of expertise</th>
<th>SA Design</th>
<th>SA Styles &amp; patterns</th>
<th>Design patterns</th>
<th>J2EE patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Novice</td>
<td>8</td>
<td>10</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Skilled</td>
<td>11</td>
<td>6</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>Expert</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Most of them had varying level of expertise in architecture design, architecture styles or patterns, design patterns, and J2EE patterns as shown in Table 2. Question on rating their level of expertise in the concepts and technologies used in the course has implications for the self-confidence of the participants. These data were collected on the first day of the course.

4.4 Training

Three 2 hours lectures were dedicated to topics directly related to the experimental study, i.e. architecture styles, design patterns, quality attribute sensitive scenarios and architecture evaluation approaches. During the course, the participants were also introduced to the concept of extracting architectural information from patterns. They undertook class exercises to give them experience of the information extraction and documentation process.

One week before the study, all the participants received detailed information about the system, LiveNet, for which they were supposed to develop change scenarios. One of the authors used LiveNet to create a network of workspaces designed to support various activities of the architecture evaluation process, e.g., architecture presentation, scenarios development, impact analysis, etc. Each workspace had roles, artifacts and different collaborative features. The participants were assigned different roles (e.g., software architect, software engineer, maintainer, etc.) in a few workspaces and asked to interact with various features of the system. A short document describing various features of the 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 review, quality attributes, general and concrete scenarios etc. The duration and format of our training was designed to make the participants representatives of most of the stakeholders involved in real world architecture evaluation, where stakeholders normally receive minimum training in creating scenarios.

4.5 Experimental Material and Instruments

4.5.1 Software requirements specifications

This study used the Software Requirement Specification (SRS) for a web-based collaborative and knowledge management application, LiveNet [24]. LiveNet provides a generic workflow engine and different features to support collaboration among geographically distributed members of a team such as asynchronous chat, discussion forum, document repository, notification, roles, planning tools, task assignment tool, etc. LiveNet enables users to create workspaces and define elements of a particular workspace. LiveNet also supports emergent processes. For further details, please see [24]. We prepared a simplified version of the SRS and a description of the system to provide the participants with as clear a picture of the system as possible.

4.5.2 Patterns used

During the course, we demonstrated the process of extracting and documenting the architecturally significant information from patterns by mining several patterns described at different level of abstraction, e.g. architectural patterns [18], design patterns [16], and framework-based patterns. However, to assess the affect (usefulness) of the extracted information on the quality of the scenarios, we needed to provide the treatment group with the ASIP extracted from the patterns used in the target application, LiveNet. The LiveNet was developed on a J2EE platform and its architecture uses a number of J2EE patterns, such as MVC, View Helper, Front Controller, Session Façade, Data Access Object, etc. The information provided to the treatment group was extracted from the core J2EE patterns [22], which have been used in LiveNet’s architecture. The information was extracted and documented using the ASIP template described in Section 3 (see Table 1).

4.5.3 Measuring quality of scenario profiles

In order to evaluate the usefulness the ASIP during the scenario development activity, we needed to compare the quality of the scenario profiles developed by the treatment group (ASIP provided) and by the control group (ASIP not provided). We used a method of ranking scenario profiles to measure their quality. The actual profile for each individual was re-coded
into a standard format for analysis. The quality of each of the recoded profiles was evaluated by comparison with a “reference profile” constructed from all the recoded profiles [6]. This method depends on the experimenters coding free-format text. To avoid bias, two experimenters recoded the profiles independently and disagreements were discussed and resolved before finalizing the reference profile.

4.5.4 Post-experiment Questionnaire

Each participant in the treatment group completed a questionnaire. The questionnaire asked the participant’s to give their opinion of the usefulness of the ASIP for developing scenarios. This question required the participants to respond by circling a choice on a five point ordinal scale and providing a short explanation of their respective choice.

4.6 Experimental Validity

4.6.1 Threats to internal validity

Internal validity is the degree to which the values of dependent variables can only be attributed to the experimental variables [23].

In order to avoid bias in allocating participants to the treatment groups, we randomized the assignment by using card sort method. We wrote the names of the participants and groups on plain cards. After shuffling the cards, we assigned one card to each group without seeing the individual’s or group’s name on the card.

Another threat to the internal validity of our experiment is the method used to measure the quality of the scenarios developed by the participants. The method was developed and validated for another experiment and various threats to its internal validity have been discussed and addressed in [6]. However, one of the potential threats associated with the method is the skill, knowledge, and bias of reference profile builder. We addressed this issue by having two researchers create reference profile independently.

4.6.2 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 industry, and whether the experimental materials and process are representative of the process and materials used in industrial architecture evaluations [25].

In an industrial evaluation, stakeholders may have a variety of different backgrounds, e.g., software engineering, marketing, management, sales, etc. 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 less than average experience of the application being evaluated.

The participants had limited experience of 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 architecture evaluation or developing scenarios. Thus, the experience of the experimental participants is likely to be similar to that of stakeholders developing scenarios for industrial evaluation.

The SRS used in the experiment is relatively short and simple compared with a typical industrial SRS. However, in industry stakeholders would be given a longer period in which 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 industrial practices for developing scenarios. However, the scenario development process in our experiments was similar to the one used for methods such as ALMA [4] and ATAM [1].

4.7 Experiment Conduct

The experiment was conducted in September 2004 as a part of scenario development workshop for a software architecture course as mentioned in section 5.4. All the 24 participants arrived according to the experimental schedule. There was a 30 minutes briefing session to revise the lecture material on architecture evaluation, generating scenarios, patterns, and the LiveNet. A document describing the LiveNet, architecture review process and example scenarios was also provided.

After the briefing session, all the participants were randomly assigned to 8 groups each consisting of 3 members and their groups were randomly assigned to the treatment group (ASIP provided) and control group (ASIP not provided). Participants were given a
simplified version of requirements for LiveNet. The participants in the treatment group also received *ASIP* extracted from J2EE patterns and documented using the template shown in Table 1. They were encouraged to use the *ASIP* during scenario development exercise.

The participants were asked to develop and rate 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. After that all the participants joined their groups to develop group scenarios for 40 minutes.

Once 40 minutes elapsed, the group scenario profiles were collected and the participants filled a questionnaire to provide their opinion about the usefulness of the *ASIP* for developing scenarios (12 individuals only). The experiment finished with a debriefing. Information collected during the experiment is tracked by an identification code present on the individual scenario profiles, group scenario profiles, and post-session questionnaire. That means data is not anonymous.

5.8 Data Collection

We collected four sets of data: the demographic data about each participant, the individual scenario profiles, the group scenario profiles and the questionnaire filled by treatment group participants to provide their opinion about the usefulness of the *ASIP*.

6. Results and Analysis

6.1 Reference Profile

We gathered 97 scenarios from 32 profiles (24 individuals and 8 groups) for the LiveNet system. We developed a reference profile to rank the scenario profiles developed by the participants. The scenarios were arranged with references to the first occurring equivalent scenario using Microsoft Excel spreadsheets, which helped calculate the frequency of occurrence of each scenario in all the scenario profiles and assign total score to each scenario profile.

The process of developing a reference profile to measure the quality of the scenarios has been extensively documented in [6]. However, we provide a brief description of this process here. To build a reference profile, we identified scenarios that related to the concept and integrated them into a single unique scenario. We noted the frequency for each unique scenario by counting the number of times it had been reported in various profiles. Then, we calculated a score for each scenario profile developed during the experiment by summarizing the frequency of each scenario in the reference scenario profile. Table 3 show the top 10 scenarios of the reference profile.

The method used to measure the quality of scenario profiles requires marking against a reference profile, which may be affected by the subjective judgment of the coder. In order to address this issue, two independent markers performed the task separately. Any disagreement regarding the scenario profile was discussed and resolved before the final marking.

Table 3: Reference profile Top 10 scenarios

<table>
<thead>
<tr>
<th>Reference Scenario Profile</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Support for PDAs and mobile phones</td>
<td>17</td>
</tr>
<tr>
<td>2 User wants to personalize the portal</td>
<td>13</td>
</tr>
<tr>
<td>3 Additional version control capabilities required</td>
<td>13</td>
</tr>
<tr>
<td>4 Modifications in the access control policies can easily be made.</td>
<td>12</td>
</tr>
<tr>
<td>5 Support for different types of DBMSs.</td>
<td>11</td>
</tr>
<tr>
<td>6 Introduce Audio and Video channels for synchronous communication.</td>
<td>10</td>
</tr>
<tr>
<td>7 Improve response time to users’ action.</td>
<td>10</td>
</tr>
<tr>
<td>8 Support topic threads &amp; auto-save features for synchronous discussions.</td>
<td>9</td>
</tr>
<tr>
<td>9 System is easily portable to a new platform.</td>
<td>9</td>
</tr>
<tr>
<td>10 Integrate project planning and management tools.</td>
<td>9</td>
</tr>
</tbody>
</table>

6.2. Statistical Results

The results of the descriptive statistics analysis on the raw scores for individual scenario profiles are presented in Table 4 and Figure 3. The descriptive statistics show that there is significant difference between the scenario profiles created by the participants working with the *ASIP* and without the *ASIP*. The average score of the participants who were given the *ASIP* is greater than the average score of those who were not given the *ASIP*.

Table 4: Summary statistics for individuals

<table>
<thead>
<tr>
<th></th>
<th>Number of participants</th>
<th>Mean</th>
<th>SD</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIP</td>
<td>12</td>
<td>74.50</td>
<td>22.94</td>
<td>6.2</td>
</tr>
<tr>
<td>No ASIP</td>
<td>12</td>
<td>55.42</td>
<td>21.32</td>
<td>6.6</td>
</tr>
</tbody>
</table>

3 Standard Deviation
4 Standard Error
The box plot in Figure 3 shows one outlier in the treatment group and slight non-normality (i.e. skewness) in the control group. Because of the outlier, the assumptions underlying parametric tests like t-test could not be fulfilled [25], we decided to use a non-parametric test. We decided to use Mann-Whitney test to test the difference between the quality of scenario profiles. We converted the data to ranks and performed a one-tailed “t”-test on the ranked data. This analysis showed that the treatment group scenarios were significantly better than the control group (p=0.036), which supports our assertion that the provision of the ASIP helps improve the quality of the scenarios.

Figure 4 indicates that the experience of treatment group participants was less than that of control group participants. Although the median experience of treatment group participants was 2.25 and the median experience of the control group was 4.5, the difference was not significant. Furthermore, as is clear from Figure 5, there is no significant correlation between experience against scenario score (r=-0.009).

6.4 Analysis of the Self-reported Data

With relatively few subjects available for software engineering experiments, it is important to obtain the maximum information from the empirical studies. We believe that it is a good practice to obtain subjective data as well as quantitative data. Subjective data can provide additional information to assist with the interpretation of quantitative results and is particularly appropriate for initial observational studies.

That is why in addition to collecting objective quantitative data as part of the two experiments, we also used a three-part questionnaire to obtain self reported data in all three empirical studies of the research program. The self-reported information was subjective but, for the most part, quantitative (i.e. assessed on an nominal scale at worst). We also obtained more qualitative data from the subjects in response to open-ended questions. Following is the analysis of the two questions of the three-part questionnaire. The first question was asked after the first part of this research (an observation study during which the participants undertook a pattern mining exercise) and the second question was asked after the experiment reported here.

**Predicted value of the ASIP.** When asked how useful the ASIP would be during architecture design
and evaluation, all the respondents seemed convinced of the value of the ASIP during architecture design and evaluation activities. 11 (61%) thought it would be useful, 6 (33%) said very useful and one person (6%) thought extremely useful. One of the respondents commented on usefulness of ASIP for architecting activities in these words “Extracted information presented in templates illustrate clearly the advantages and disadvantages of using a pattern, and provides an easy approach to choose specific pattern for specific requirements”. We consider the responses to this question to correspond to predicted value because at that time the respondents had not used the ASIP for any of the architecting task. However, the responses to the next question confirmed their predictions.

**Usefulness of the ASIP for developing scenarios.**

Once the participants had used the ASIP during scenario development activity (only 12 used the ASIP as part of the treatment group of a controlled experiment), we asked them about the usefulness of the ASIP for developing scenarios. Eleven of the respondents found that the ASIP was a useful support material for constructing concrete scenarios (58% very useful and 34% useful). They based their concrete scenarios on the general scenarios provided by the ASIP. Several participants also explained why they thought that the ASIP was helpful in developing quality-sensitive scenarios. One participant reported that “information in the template is more integrated and informative, which directly targets scenarios. It prompts you in thinking effectively about scenarios”.

Only one person found that the ASIP was not useful in developing scenarios but that participant did not provide any explanation for that.

**7. Discussion**

Our statistical analysis indicates that we can reject the null hypothesis that ASIP does not improve quality of scenarios developed to evaluate an architecture in favour of the alternative hypothesis that ASIP does improve the quality of scenarios. Moreover, we can also be confident that the experimental results are reasonably robust, because any bias caused by experience differences between the experimental groups would favour the control group not the treatment group.

Objective quantitative information is essential to assess the benefits and limitations of technology. However, subjective information provides insights into how acceptable methodological changes and innovations will be to the people who will be expected to employ them. This type of information is essential if process change is to be successful. We gathered self-reported data throughout this research program, one part of questionnaire was served after each part of the study. The participants of the study were convinced of the value of the ASIP during software architecture design and evaluation even before using the ASIP.

After using the ASIP during scenario development task, the majority of the participants believed that the ASIP assisted scenario development. Thus, the subjective opinions of the participants are consistent with the results of the statistical analysis.

One possible drawback of using the ASIP for developing concrete scenario is that stakeholders might limit their effort to concretize the general scenarios provided to them. In this case they would end up with only those scenarios that would have already been considered as sources of the general scenarios (patterns) that are being used in the architecture being reviewed. We have not observed this problem in our experiment. A close analysis of the scenarios developed by the participants revealed that the scenario profiles of the participants, who were given the ASIP, included the concrete scenarios which were based on the general scenarios included in the ASIP but also included several other concrete scenarios as well. However, a better approach might be to provide stakeholders with the general scenarios that are extracted not only from the patterns being used in the architecture to be reviewed but also from patterns not used in that architecture.

Another point worth mentioning is that our experimental results apply to the scenarios produced by individuals with extensive IT experience, limited knowledge of the system being evaluated, and a relatively simple SRS. Before we can be confident that the results will generalize, our experiment needs to be replicated with non-technical participants using more complex SRS documents.

**8. Conclusion and Future Work**

One of our main research goals is to improve the architecting processes by gathering and documenting architecturally significant information in a format that is readily useful in making design decisions with an informed knowledge of the consequences of those decisions. We aim to reduce the time, resources and skill level required to effectively and efficiently design and assess architectures with respect to desired quality attributes. We have proposed a technique of “mining patterns” as one of the steps towards that goal.

This paper reports one part of a research program designed to assess the value of architecturally significant information extracted from pattern (ASIP) to improve architecture design and evaluation.
activities. When we presented our initial work on capturing and documenting general scenarios by “mining patterns” for improving software architecture evaluation process [21], we also delineated some short term goals to refine and empirically validate our idea. One of those goals was to develop a quantifiable means of showing benefits of using the architecturally significant information extracted from pattern.

In this paper, we present the design and results of one of the two experiments of the research program aimed at achieving that goal. Both experiments of this research program are aimed at gathering empirical evidence to support our view of the usefulness of the ASIP for improving architecting activities (e.g. developing quality sensitive scenarios and selecting appropriate patterns to design architecture). The findings of the experiment reported in this paper provide empirical evidence to support one of our assertions that the ASIP is useful for scenario development activity.

We are currently analyzing the results of the final experiment in the program, which investigates whether the ASIP improves the architectural design process. We will report the results of this experiment in a future publication. Future work will also involve designing and implementing a repository of architecture knowledge to support the process of capturing and reusing the ASIP during architecting activities.

Acknowledgement – We thank the participants of this experiment. Cynthia Wang helped us build the reference profile. National ICT Australia is funded through the Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council.

9. References