ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014

SOFTWARE PATTERNS vs QUALITY ATTRIBUTES (INVESTIGATION APPROACH)

Hassan Almari[1], Clive Boughton[2]

Australian National University (ANU), Research School of Computer Science, Canberra, Australia[1,2]

Abstract: The development of software patterns(SPs)is aimed at providing a reliable and reusable framework for resolving similar problems within distinct contexts. To accomplish this objective competently, it is imperative to document these patterns effectively to facilitate the comprehension of their concepts to users, thereby encouraging their use over and over again. Thus, the documentation of patterns needs to explicitly explain their relationship with the quality attributes (QAs) that they support, or hinder, in order to satisfy the implementation of stakeholders" requirements. The variation in patterns descriptions in contemporary literature renders the explanation of the above relationship complex and difficult to follow. This eventually deters developers from employing patterns or causes them to overlook their QAs. Either of these scenarios may result in significant expense in terms of development time and cost, and/or attaining required system quality. This paper tries to address the aforementioned problem by comparing and analysing six well known software pattern resources, pinpointing the aspects of variation amongst authors descriptions, which lead to different relationships between patterns and QAs, which in fact cause confusion among users. Once the variance concept amongst these six resources in terms of terminology and description has been addressed, we derive a relationship matrix between the software patterns (included in these resources) and the standard ISO-9126 QAs. We believe that this research work is a positive contribution to the enhancement of techniques for documenting software patterns. It further helps improve pattern selection by users via improved prediction of output quality. Thus, to provide a reliable method for maintaining and representing the research work, we have created a database application that identifies the above relationship. This database also includes discrepancies among the documentation approaches of the six resources that we have studied, as well as the variance in pattern categorisations and terminologies. The pattern database should also serve future research endeavours. This research study received a positive response as per the findings of a questionnaire aimed at software professionals and based on the context of the preceding problem. 97 precent of the participants, from six different nations, answering the questionnaire supported this study.

Keywords: Software engineering, Software architectures, Patterns, Quality concepts, Quality analysis and evaluation, Documentation

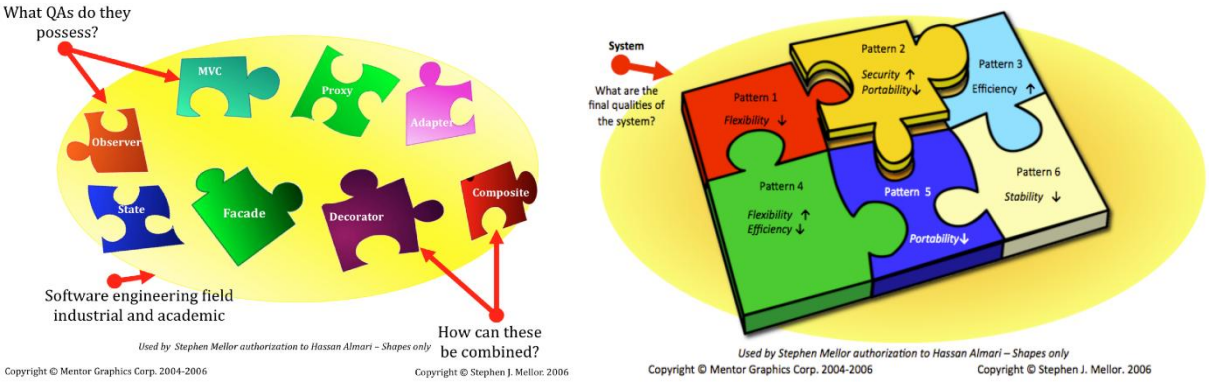
I. INTRODUCTION

Currently, most software pattern resources describe patterns based on the authors "experiences and observations. Some of these resources have pointed explicitly to the relationship between each pattern and their (apparent) quality attributes QAs, i. e., [1], [2] while others do not; i.e., [3] and [4]. However, there are a few works analysing the identification of the relationships between software patterns and quality attributes in a scientific methodology [5], based on measurements and metrics, such as that of the work done by [6] and [7]. The former mentioned work of Kim/Garlan used (Alloy-Analyser) as a tool. They tried to create models using some patterns, and to evaluate some quality characteristics. Their work focused on mapping rules between architectures and models, while maintaining the properties like consistency, style compliance, reliability. Whereas the latter work is concerned about the evaluation of software architecture by metrics, which is applicable to software patterns too. In the work of Dr Zayaraz [7], he did use different available methods within his evaluation framework. For example he applied the rules and principles of the Common Software Measurement International Consortium (COSMIC) Full Function Points with some metrics to measure the basic interaction parameters for some characteristics such as coupling, cohesion, and complexity on different patterns (e.g. Pipes and Filters). Also, he did use Analytical Hierarchy Process (AHP) for comparisons between different pattern structures for specific quality attributes. Both approaches are a good step forward to building a concrete relationship between patterns and QAs based on scientific methods and measurement, not just an observation or experience of the pattern author. More research effort needs to be done to answer some of the questions that have been illustrated in Figures 1a and 1b.

This paper attempts to highlight some important factors that impact pattern usability within the software engineering discipline, that are caused by conflicts between several pattern resources regarding relationships between patterns and their quality attributes (Patt-QAs).

ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014



a b

Fig. 1 Visualizing the Problem Area.

A. Rationale Of The Investigation Approach

This investigation was necessary for two reasons: (1) to emphasize the problem concept, (2) to increase the proposed solution value to pattern users. This accomplished based on three processes shown in Figure 2. The first process is, to highlight the differences between definitions, terminologies and categorisation as factors that challenge identifying relationships between software patterns and QAs (Patt-QAs). Seven analysis steps have been carried out to satisfy the first process as described in Table I.

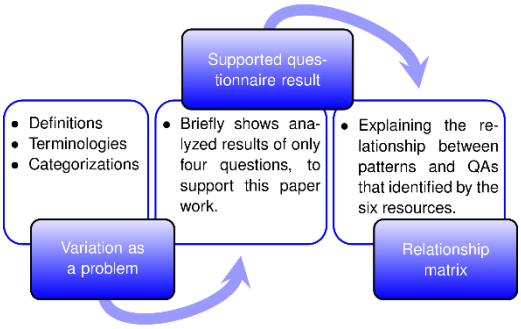


Fig. 2 Investigation approach processes

Second process is, to discuses some survey questionnaire results, which support the existence of the problems described in the first process. Also, it supports our proposed solution to build a database of the relationships between patterns and QAs. Thirdly are, generating a metrics suite designed to express the investigation undertaken for six credible and definitive sources of patterns with respect to their characteristics. The software patterns sources included in this study are:

1. [1] – the Gang of Four (GoF) book,
2. [2] – POSA-V1,
3. [8] – POSA-V2,
4. [9] – POSA-V3,
5. [10] – Software Engineering Institute – Software Architecture in Practice, and
6. [11] – Security Patterns.

The Selection of these sources is based on the authors preliminary research, also by supportive respondent's answers to a questionnaire done in 2012, (by the researchers). Almost half of the respondents identified GoF and POSA books as their reliable, popular and known pattern references. While the [10] and [11] included in this study as important part that tackle architectural and Security patterns, which is valuable to the research main goal.

The rest of the paper is organized as follows: Section 2 discusses the problems associated with differing pattern definitions, terminologies and categorisation, then briefly argues how QA definitions, terminologies, and categorisation cause problems in identifying their relationship with software patterns. Section 3 introduces an example that supports our claims in Section 2. Section 4 lists issues arising from variation on both domain software patterns and QAs. Finally, in section 5. We introduce some important findings from our survey that supports this investigation and our proposed solution, then summarise the database information and structure in section 6. Followed by the conclusions Section 7.

II. PATTERNS AND QUALITY ATTRIBUTES REFINEMENT

To create or describe a pattern we should understand the concept of pattern and follow rules or constraints to document them in the right way. To assess patterns against QAs, we should do the same to the QAs concept. The rest of this section lays out the problems that existed within the concept and rules of creating and documenting patterns within software engineering, that have a direct impact on their utilization and evaluation. Also, this section presents justifications as to why we built a (Patt-QAs) relationships database, and some of the challenges that have been faced during this process.

A. Problems Discovered Within Current Pattern Definitions And Terminologies

Numerous pattern definitions are being suggested for varying contexts. It is therefore difficult to define patterns in commonly acceptable terms. However, it seems

ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014

TABLE I DESCIPTIONS OF THE 7-ANALYSIS STEPS FOR THE TARGETED RECOURSES

Process #	Investigation Steps	Description
1	Pattern Resources Selection	Identifying the most widely and reliable resources within the field of software patterns through concrete literature review, which become the targeted resources for this investigation study.
2	Pattern Categorization Approach	Study and compare all categorization approaches within the selected resources.
3	Pattern Descriptions	Study and compare the description of patterns between targeted resources in the domain of quality attribute relationship. This step includes the investigation of every resource and the way they define and categorize quality attributes in their descriptions.
4	Quality attribute Approach	Selection Identifying one of the best-standardized practices in the field for defining and categorizing the quality attributes through a literature review. Then we use the selected approach for identifying the relationship between patterns and (QAs). Also, we use it for comparisons between different quality attributes categorization schema within the targeted recourses.
5	Creation of the Relationship Matrixes	Based on the pattern descriptions within the targeted resources, and the description of QAs by the selected approach, we built relationship matrix for each resource and a common matrix for all of the resources that identified the relationship between patterns and QAs.
6	Creation of the Quality Attributes Categorization Tables	Based on the information collected from steps 1–5, we created comparisons tables for the QAs classifications, between selected QAs approach and others within the targeted resources.
7	Conflicts and Issues	Based on the investigation steps 1–6, we have identified any relationship conflicts and issues within the descriptions of patterns on targeted resources.

sufficient to say that a pattern is essentially the solution to a problem within a particular domain which can be applied to help resolve similar problems in different contexts within the same domain. The definition of 'context' has evolved over time, for the purpose of this paper/study we believe that Dey's definition is the most appropriate and is probably the most widely accepted.

Dey's defines the 'context' as «any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between (for example) a user and an application, including the user and the application», [12].

The definition of a pattern as described by GoF is «a solution to a problem in a context». This definition, however, was unacceptable to Dick Gabriel [13], who believed that it failed to illustrate the significance of the concept, and may even cause misinterpretation amongst software professionals. Gabriel also believed that many of the existing pattern definitions were indistinct and did not accurately express the implications patterns have. He therefore proposed a new definition, amending an early version by [14]: «Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves».

Likewise, [2], [15], [16], and Gabriel[13], each one have his own pattern definition.

Most of the definitions above share common key points with a few variations. Some are more elaborate than others or include some further important aspects such as forces. Defining the forces that drives and constraints the most appropriate solution to a problem in the form of a pattern is an important step during pattern creation [14].

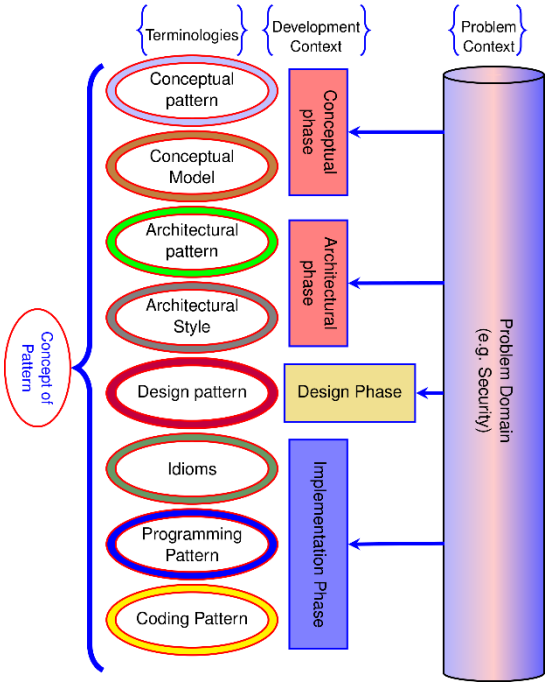


Fig. 3 Terminologies of "Pattern" within software development lifecycle.

Furthermore, having different terminologies and names in real life to explain the same thing, often due to differences in cultural factors or language, is acceptable. However, this is improper in the context of software patterns, as it leads to confusion. It is therefore considered as an absence of standardization, which can cause major challenges, [17]. Therefore this research aims to minimize some of these challenges by explaining the problem area and introducing the (Patt-QAs) database with its benefits and features, (see Section 6).

Terminologies shown in the Figure 3are being used within the current literature. For example, the Architectural-Styles termed by [18] and [10], Architectural-Pattern by [2] and [19], [20] and [15] name it a Conceptual-Model,

and a Conceptual-Pattern by [16]. Many software developers use patterns in different stages of the software development lifecycle. We believe that the *problem context* persists, while the *context of developments* changes as described by Figure 3.

More (redundant) terminology increases the challenge of patterns' usability. It appears to same readers that the terms described in Figure 3 suggest different concepts. But are they?

The philosophy surrounding the conceptual or architectural 'model, style, and pattern' in the aforementioned terms attempts to convey a single idea through various explanations. All of which share the concept, components, restraints, and relationships that focus on a high abstraction level. However, the conceptual models should be explained further through detailed descriptions, in order to be able to move from an architectural context to a design context and so forth.

We believe that all terminologies shown in Figure 3 do have the same concept of pattern, with minor differences, to fit into the various development contexts. Also, less terminology surrounding pattern, and a concrete description of a common formal term, lead to better utilization and understanding of software patterns, which shall minimize the confusion in the midst of its users. So, many existing definitions and terms for the same concept (as illustrated above) was a challenging factor during this study. As a result, and based on this study, the relationship database has been created for all patterns included in the selected resources. All patterns for all levels of the development life cycle are gathered in one place, with an indication of their names, definitions, and categories, to help developers to compare and find relevant information regarding the included patterns and their relations with QAs with little time.

The same discussion above also applies to the design phase as briefly discussed below:

Alexander defines design as «a process of synthesis, a process of putting together things, a process of combination, [14].

According to [2], design patterns depict frequently occurring arrangements of interacting components, thus helping to resolve design dilemmas in a given frame of reference. What this essentially suggests is that a pattern cannot be translated into code, but rather the pattern should be moulded in a way that it provides a solution to the problem.

Whilst, currently software developers can select a pattern as an available code artefact, alter it to match his/her problem context and finally convert the entire package into code. Nonetheless, we agree with [2], that patterns should be highly generic with textual explanations in addition to block and connector diagrams, in order to support higher reusability in multiple contexts and better understanding. However, the textual explanations and the block and connector diagrams should not be arbitrary. Also, should be applied within a common standardized procedure or a framework.

Various definitions (rules), of design patterns that convey the diversity of terminology and description can be noticed by comparing between the definitions of [1], [2], [21]–[23] To conclude, the concept of a repetitive 'structural' pattern theme can be used for describing the architecture, design, and implementation, and what's different then? Is the changed context. So, reducing pattern documentation conflicts, needs more research and standardised procedures, to helps increase the effective use of patterns. Same concept been discussed earlier in architectural level within Figure 3 describtion.

B. Problems Discovered Within Current Pattern Categorisations

Coupled with the expansion of pattern diversity, there is a corresponding rise in the emphasis on the obligation to categorize patterns. To meet this end, a categorization outline is employed to organize the patterns as a collection so as to make them accessible for searching and storing by users. For the purpose of this section we have add POSA-V4 with other resources from section 1A).

The classification approaches for the investigated resources are:

- The first and the second volumes are based on two primary categories: 'pattern' and 'problem' categories. The pattern category is subdivided into 3 types in both volumes, while problem category is organized into 10 types in POSA-V1, and 4 types in POSA-V2.
- POSA-V3 were based on 3 primary categories within the domain of typical resource management lifecycle. These categories were resource acquisition, resource lifecycle and resource release.
- POSA-V4, the patterns were categorized on the basis of 13 technical topics and distributed systems.
- GoF team, however, used a different approach, classifying patterns based on purpose and scope. The 'purpose' has been further sub-classified into creational, structural, and behavioural categories, while 'scope' into categories of classes and objects.
- SEI book by [10] contains architectural styles that are categorized on the basis of respective subjects and relations. [10] describe thirteen different styles, of which the five primary styles are independent components, data flow, data-centre, virtual machine, and the call and return. The primary styles signify the relationships amongst the sub-styles and their respective topics.
- The book on security patterns by [11] comprises pattern categories bearing reference to enterprise and system levels within the security domain, and is related to engineering and operations activities at all levels.

Based on this study we found that the description of the technical topics (POSA-V4) are the same as «technical problems», which shares the same concept of the «problem category» that have been recognized in volumes 1 and 2. For example, the *From Mud To Structure*, have been described as a problem category in POSA-V1, and as a technical topic in POSA-V4.

ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014

From comparing the targeted resources mentioned above, it is clear that there is no common approach for categorising patterns. However, we believe that the 'problem' category as a concept, is shared between many pattern books, although under a variety of names, for example, it is named 'purpose' in GoF book; 'problem' in POSA-V1 and V2 and 'technical topics' in POSA-V4, and as 'main style or related subject' in SEI.

Also, as an example of confusing categorisation schema used in these books is that of the Interpreter pattern, where GoF considers this pattern as a design (behavioural) pattern, but the SEI group consider it an architectural (virtual machine) style. So, what is the Interpreter pattern, and does this affect the reusability of this pattern? Can we use the same pattern, that explained by GoF in the context of a virtual machine, as explained by SEI group, or do we need to adjust it to fit the new context?

This lack of a common classification, particularly for scenarios that are technical, such as software patterns, can end up complicating things for users, researchers and readers. Therefore, when users seek appropriate patterns for resolving certain real-life issues, they are confronted with different guides and classifications for what are essentially the same patterns. Whilst, this can assist the users in employing the patterns in diverse contexts, it may also contribute towards making the reuse factor of patterns more complex, unmanageable, and less efficient. To assist with minimising such confusion, this study provides a database with information regarding 168 pattern (in-total) names and classification, helping developers compare and choose the most appropriate patterns for their problem domain.

C. The Variation Concept As A Problem within QAs

There are many different schools of thought regarding the management of QAs and how they can be addressed effectively such as, ISO, SEI, DoD STD, and IEEE, [24]. Hence, there are challenges that arise when quality has to be defined in the real world. This section tries to demonstrate in brief the difficulties that arose during this study from the QAs documentation variation viewpoint.

, include all variants and relationships with quality attributes.

According to Mitra 2008 and reference therein pertaining to Juran and Gryna (1993), Crosby (1979), IEEE-1061, and ISO-9126, each have their own individual concept of quality. Doctor Ronald [25], argues that there are variations in QA definitions that are acknowledged by both the community and researchers involved. The presence of different concepts of quality amongst different people and communities illustrates that there are variations within the definitions for each QA that may share some characteristics and differ in others. However, small variation within QA definitions could increase the difficulties in defining and evaluating software patterns against them.

Likewise, the terminological variations concept persists with QA categorisations, same as the pattern categorisations issue discussed earlier. So, depending on the domain, people have designed different ways to classify QAs using different approaches. The needs for further research and study increased; however this will not be discussed in this paper. The focus here is to explore the differences in QA categorisations within our six sources and demonstrate the issues elucidated by these differences, which will be discussed in section 3 and 4. However, the relationships database included all the QAs definitions and categorisations for ISO 9126, because this a standards represents a broad agreement of QAs. Also, QAs definitions and categorisations for all targeted resources where applicable is included in the database, to help the users to make their comparisons between different approaches.

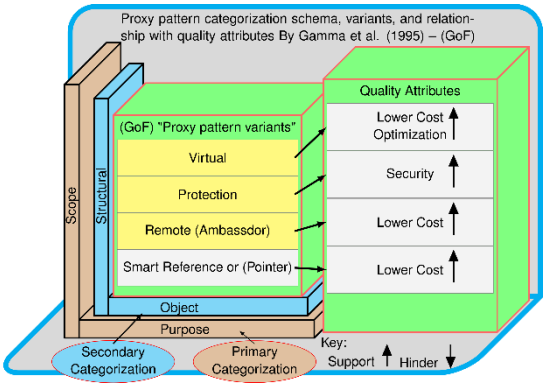


Fig. 4 GoF team approach for classifying, describing Proxy patterns

III. CONFLICT EXAMPLE - (PROXY PATTERN)

This example for illustrative purposes of the issues discussed in Section 2. It is a comparison of the Proxy pattern documentation approach, between the GoF and POSA-V1.This comparison shows some of the differences that we think lead to confusion and that minimize the utilisation of software patterns.

The definition of the Proxy pattern has similarities in both resources. While, POSA-V1 did elaborate further in their description. However, there are more differences within the Proxy pattern such as: (1) their instances or variants, (2) their primary and secondary categorisations, (3) their relationships with QAs. Figure 4 and Figure 5, visualize the above three differences.

The GoF divides Proxy patterns into 4 variants: remote, virtual, protection and smart reference as presented in Figure 4 Contrastingly, the POSA group divide the Proxy pattern into 7 variants, namely remote, protection, cache, synchronization, counting, virtual, and firewall as seen in Figure 5 The common variants between both methods of classification are remote, virtual and protection. The important question being, which QAs are supported or hindered by those variants in both references.

ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014

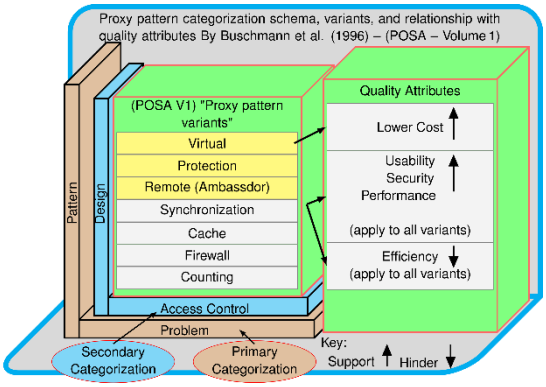


Fig. 5 POSA team approach for classifying, describing Proxy patterns, include all variants and relationships with quality attributes.

Figure 4 shows that all GoF Proxy pattern variants support 'lowering cost' as a QA, and Virtual and Protection patterns supporting optimization and security respectively.

The POSA team on the other hand considered all Proxy pattern variants, including common ones such as Remote, Virtual and Protection patterns, to be supportive of usability, security, and performance. Unlike the GoF scheme, efficiency and lower-cost are supported only by the Virtual pattern. Whereas, efficiency is hindered by all other variants, as shown in POSA team approach, Figure 5

The above divergence in the categorisations schema and relationships between patterns with QAs increase confusion, making it harder to predict outcome quality when utilizing these patterns, as well as reducing pattern usability.

IV. ISSUES DISCOVERED BY THIS STUDY

- There are no specific definitions or categorisations of QAs that are presented by [1]. The approach taken instead focuses on the explanation of how patterns can be used to support claimed QAs. They used their own words and examples to explain QAs in the context of software patterns.
- ISO-9126, POSA Books, and SEI [10], defined QAs with various differences using various vocabularies. Although the concepts of their definitions are largely similar for each QA. However, they do varied in their sentence structuring, terminologies and how many features or constraints are included within their definitions. We believe, that any additional (features or constraints) added to any QA definition should be considered as a prerequisite that needs to be fulfilled, to achieve that QA with all it' s characteristics. As a result, the above variations in the QAs descriptions could have an impact on the overall evaluation process for any system or structure (e.g. patterns), and cause a conflict between development' teams if they use non-common descriptions for the intended requirements (e.g. QAs).
- ISO-9126, POSA Books, and SEI [10] present different QA categories. For example, ISO-9126 and POSA Books, each have 'Reliability' as one of their main categories, but they differ in their sub-categories as illustrated in Figure 6It is clear then that we will experience differences when trying to satisfy or validate the 'Reliability' QA using both approaches. For more information, see the QAs categorisation table in the database.
- One of the biggest causes of confusion and difficulty in traceability is the use of different names for the same patterns or one name for different patterns. For example, GoF team explained Adapter and Decorator patterns as two different patterns, which they are. However, both have been identified as Wrapper pattern. It is neither logical nor user-friendly for the same pattern to have different names or different patterns have the same name, making it hard to identify, trace and apply. It is understandable to have a variety of names if the pattern has individual instances or variants, such as the Proxy variants example discussed earlier. There are other examples of this «documentation problem» where the same pattern has various titles: Publisher-subscriber, Observer and Dependents are all different names for the same pattern. Indeed there are 8 different names described by [11] for Check-Point pattern alone, which are (Policy Definition Point (PDP), Policy Enforcement Point (PEP), Access Verification, Holding off hackers, Validation and Penalization, Make the Punishment fit the Crime, Validation Screen, Pluggable Authentication). However, GoF and POSA books have provided something as a solution to this problem, by introducing «Also Known As» section. Other resources such as [3] and [10], however do not acknowledge alternative names in their work.

Some resources include the same patterns with the same names and definition, but with different QA relationships. For example, in POSA-V1, the Piping and Filtering pattern supports Testability and Exchangeability, whereas SEI book lists it as supporting Maintainability and Usability. Questions therefore arise as to which QAs the pattern truly supports, and how these different conclusions have been reached. Not forgetting that QA relationships seem arbitrary, and the answer most probably lies with the differing experience and observations of the pattern authors, or because there is still a lack of proper methodology to capturing and documenting patterns, as we believe.

TABLE II: METHODS SELECTED DURING THIS ANALYSIS.

Individual analysis methods	*Multi-dimensional analysis methods*
Several types of graphs (e.g. bar chart, pie chart), frequency tables, descriptive statistics, a nonparametric Chi-square, and numerical measurement for the (Likert) type questions.	Several types of graphs (e.g bar charts, scatter plots) , one sample t-test, a cross tabulation with Chi-squares, and descriptive statistics.

ISSN 2393-8021

International Advanced Research Journal in Science, Engineering and Technology
Vol. 1, Issue 1, September 2014

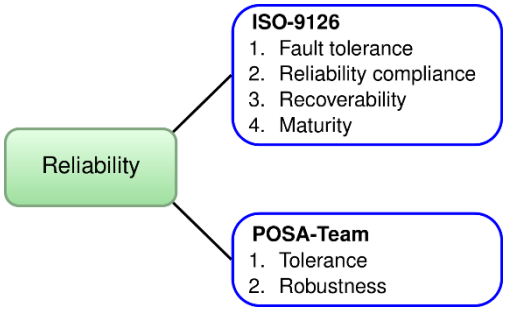


Fig. 6 Reliability as an example of the differences within QAs categorisation.

Using expert knowledge regarding recurring problems to provide feasible solutions to the community relies on good standardized documentation, as recommended by [17], that standardisation helps decrease the challenges facing software development, preventing user confusion. To follow Garlan advise, we used the ISO-9126 model as the reference from which to build the relationship matrices between patterns and QAs, using the information described in all 6 resources studied.

V. PATTERN QUESTIONNAIRE AND ANALYSIS

In this section the researchers report the results of a survey designed to establish the reasons affecting the utilization of software patterns. A (secondary) goal of the survey was to obtain the agreement of survey respondents to some proposed solutions that could help developers with understanding better the effective use of patterns during the process of selecting and deploying them. A high level of confidentiality was applied during gathering and analyzing responds. The following sections outline the process and methods used in the analysis of the responses to the questionnaire. Section 5C shows an important portion of the questionnaire that is related to the scope of this paper. During this analysis the Statistical Package for the Social Sciences (SPSS) tool, was used. This explanation is to facilitate tables and figures notations.

A. ANALYSIS PROCEDURE

The survey was divided into three different sections as follows:
The first section focused on gathering information regarding respondents personal expertise. The second section centred on determining the reasons that affect the usability factor of software patterns during development processes. The last section was aimed at discovering issues that are related to current software patterns documentation, and also, to obtain the respondents' agreement regarding some proposed solutions by the researches.
The analysis procedure was carried out in two steps as follows:

1. One-dimensional Analyses. Each question was analyzed and summarized in the form of graphs and tables where needed.
2. Multi-dimensional Analyses. In this step we analyzed more than one question together (matrix-cross-correlational), to see if there are any relationships or dependences between various factors. The selections of the questions were based on the overall objective of the investigation.

B. JUSTIFICATIONS OF THE METHODS USED DURING THIS ANALYSIS

Several techniques were used to carry out this analysis, the selection of the methods based on best technique that suit the type of questions, such as questions with ordinal scale, t-tests were employed, and for dichotomous variables a pie charts were used etc. However, due to the paper limitations we briefly named the methods that been used for each category, see Table II.

C. RELATED ANALYSIS

In this section we will show the statistical results of the general agreement amongst the questionnaire respondent towards the four mentioned statements that shown in Table \ref{tab:4Q. These questions was proposed as a solutions to some of the issues discussed in sections 2, 3, and 4
Each of the statements responses were of Likert scale, variables are of ordinal scale, so numerical measurements are meaningful. Assigning 1= Strongly Disagree to 5 = Strongly Agree, the neutral option was assigned to 3 as it is value. So, one interesting matter is to see whether there any tendency to «Strongly Agree» or «Strongly Disagree». A one sample right tail t-test will be useful to see the general agreement of the respondents(see Table IV).
So, our hypothesis will be based on the neutral selection (Neutral value = 3), as follows:
Null hypothesis, H_o: $\mu \leq 3$
Alternative hypothesis, H_a: $\mu \geq 3$,
where, μ is the mean score of each of the statements.
Statistical analysis results are:
The overlapping (95 percent CI) error bars on the (Agree=4, option) indicated that most of the respondent agreed with all four statements as illustrated by Figure 7.

TABLE III: THE 4 QUESTIONS – THAT SUPPORTS THIS STUDY.

Please indicate your level of agreement with respect of the following statements:						
Q17	Identifying the relationship between software patterns and quality attributes is very important to software developers and the software engineering field.	○ Strongly Agree	○ Agree	○ Neutral	○ Disagree	○ Strongly Disagree
Q18	Identifying standard quality attribute definitions within current pattern references is a critical for comparing the same patterns against the quality attribute they possess.	○ Strongly Agree	○ Agree	○ Neutral	○ Disagree	○ Strongly Disagree

Q19	Studying relationships between patterns and quality attributes based on the current reliable software pattern references, and creating a database to store these relationships on the basis of standardized quality attribute definitions, is valuable knowledge.	○ Strongly Agree	○ Agree	○ Neutral	○ Disagree	○ Strongly Disagree
Q20	Developing an evaluation model to assess patterns against quality attributes is worthwhile, provided it's not difficult to use.	○ Strongly Agree	○ Agree	○ Neutral	○ Disagree	○ Strongly Disagree

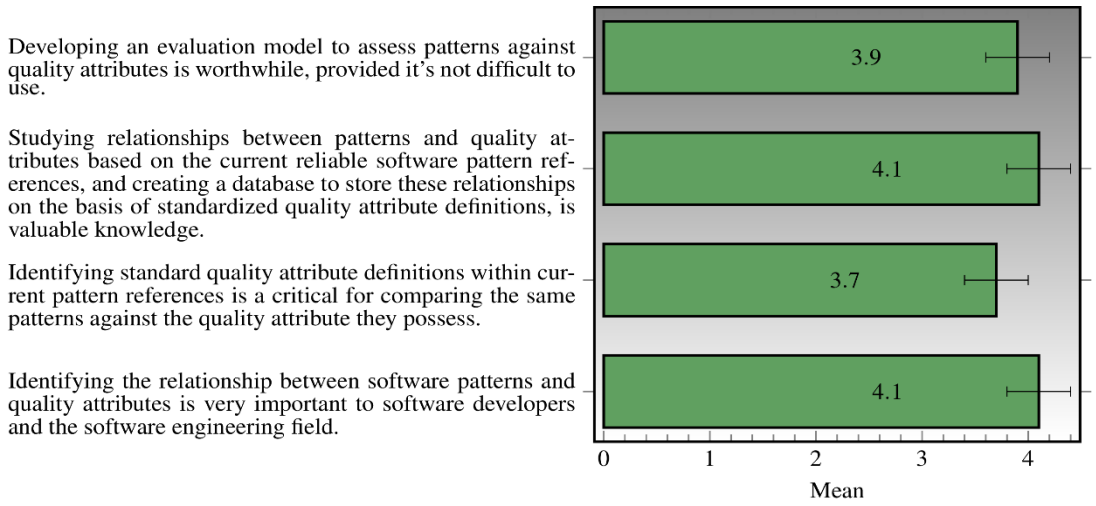


Fig. 7 Distribution of the Mean with Error bars: 95% CI.

TABLE IV: ONE SAMPLE RIGHT TAIL T-TEST.

	Test Value = 3					
					95 % Confidence Interval of the Difference	
	t	df	p-value	Mean Difference	Lower	Upper
Identifying the relationship between software patterns and quality attributes is very important to software developers and the software engineering field.	9.5	33	.000	1.1	.9	1.4
Identifying standard quality attribute definitions within current pattern references is a critical for comparing the same patterns against the quality attribute they possess.	4.3	31	.000	.7	.4	1.0
Studying relationships between patterns and quality attributes based on the current reliable software pattern references, and creating a database to store these relationships on the basis of standardized quality attribute definitions, is valuable knowledge.	6.3	31	.000	1.1	.7	1.4
Developing an evaluation model to assess patterns against quality attributes is worthwhile, provided it' s not difficult to use.	5.7	30	.000	.9	.6	1.2

Aslo, to investigate the respondents agreement significance with 95 percent confidence interval towards the four mentioned statements, one sample (right tailed) t-tests were performed. The test is significant for all of the statements as described in Table IV.

To sum up, this paper presents the work that satisfied part of the respondents' wishes in Q17 and Q19 (see Table III) and to contributes to software patterns community, by identifying the relationships between some existing software patterns and QAs. Also, by developing a database to represent this information in easy way for the users. More research needed to provide solutions to the statements presented in Q18 and Q20 above, (see Table III).

VI. BRIEF DESCRIPTION OF DATABASE OF PATTERNS VS QUALITY ATTRIBUTES RELATIONSHIPS

It is recognized the importance of software patterns and QAs relationships to the software development processes. Investigating, and analyzing of these relations were carried out to help users to locate their desired relationship in short time and easy way through the developed database. There are several tabs, each one have many services. We recommend users to start with the overview tab to understand the overall structure of the database, and to facilitate their navigation process.

In total, we categorised 168 patterns and identified/systematised the known relationships between 120 patterns and 50 QAs within our database. Our database contains these relationships as well as other features such as search functions that can be used to easily find any patterns, conflict relationship or QA. Users can therefore explore each reference included in this study in an individual matrix, or view the pattern categorisation table for an individual resource. Each pattern has a

description table consisting of definitions, alternative names, comments and relationships. A contrast tables of QAs classifications between POSA, SEI and ISO-9126 is also included.

In the future, the description table will needs further updating in order to enhance knowledge about patterns and QAs. Furthermore, they will also include forces, scenarios, quality tactics and quality metrics, as well as other information deemed essential for comprehensive knowledge about software patterns and their QAs relationships. In addition, the database built to be easier to explore as well as navigate through the user-friendly interface and menu. Users are therefore in a position to create, delete or even modify any relation. This database means that all the information on this subject is gathered into one place, providing summaries for numerous resources. The importance of the database comes from its ability to effectively save users time and effort, especially those who are concerned with finding a brief summary about particular patterns. To conclude this section, developing the database was very hard and time consuming, due to all processes involved from investigating to representing the information included. As a result, the database application was produced in such manner that it will be practical to other researchers and analyst. The database could be navigated with a proper access authorization through the researchers.

VII. CONCLUSION

This investigation of the relationship between QAs and software patterns has highlighted two main issues. Firstly, there are differences between pattern documentation within the current literature, which may because of different factors such as, authors experience and the maturity of the patterns in the field of software engineering. Secondly, there isn't concrete approach or process to be followed for describing the relationship between patterns and quality attributes, or for categorizing them in a more sensible formal/verifiable way. Both points above have led to the existence of conflict relationships between patterns and QAs, which decreases the utilization of patterns by users. Our major research objective is to aid software engineering community to see and help overcome the pattern documentation problem that we have identified. Also, to help patterns users to build better software by selecting patterns without ignoring their quality attributes, through visualizing this relationships within presented database, which been identified by several credible resources in the field. We believe that mining software patterns and pointing to any issues within their descriptions is an important step to improve pattern documentation, which already have a major affect on distilling and documenting software artifacts during software development lifecycle as discussed in Sections 1, 2 and 3.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. USA: Addison-Wesley, 1995.

[2] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture - A System of Patterns*, vol. 1. Chichester, UK: John Wiley & Sons, 1996.

[3] D. C. Schmidt and L. Rising, *Design patterns in communications software*, vol. 19. Cambridge University Press, 2001.

[4] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, and R. Stafford, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

[5] R. Freitas, "Scientific Research Methods and Computer Science," 2009.

[6] J. ~S. Kim and D. Garlan, "Analyzing architectural styles with alloy," in *ROSATEA '06: Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis*, 2006, pp. 70–80.

[7] G. Zayaraz, "Quantitative Approaches For Evaluating Software Architectures," Pondicherry Engineering College, Puducherry, India, 2010.

[8] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-Oriented Software Architecture - Patterns for Concurrent and Networked Objects*, vol. 2. Chichester, UK: John Wiley & Sons, 2000.

[9] M. Kircher and P. Jain, *Pattern-Oriented Software Architecture - Patterns for Resource Management*, vol. 3. John Wiley & Sons, 2004.

[10] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 1st ed. USA: Addison Wesley Longman Inc., 1998.

[11] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. Chichester, UK: John Wiley & Sons, 2006.

[12] A. K. Dey, "Understanding and using context," *Pers. ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, 2001.

[13] J. O. Coplien, "A Pattern Definition, http://st-www.cs.illinois.edu/patterns/definition.html." .

[14] C. Alexander, *The Timeless Way of Building*. USA: Oxford University Press, 1979.

[15] M. Fowler, *Analysis patterns: reusable object models*. Addison-Wesley, 1997.

[16] D. Riehle and H. Züllighoven, "Understanding and using patterns in software development," *TAPOS*, vol. 2, no. 1, pp. 3–13, 1996.

[17] D. Garlan, "Software architecture: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, 2000, pp. 91–101.

[18] R. T. Filding, "Architectural Styles and the Design of Network-based Software Architectures," University of California, IRVINE, 2000.

[19] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. USA: Pearson Education, Inc., 2013.

[20] C. Alexander, *Notes on the Synthesis of Form*. Presidents and Fellows of Harvard College, 1964.

[21] P. Wolfgang, *Design patterns for object-oriented software development*. Reading, Mass.: Addison-Wesley, 1994.

[22] J. O. Coplien and D. C. Schmidt, Eds., *Pattern Languages of Program Design*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995.

[23] A. Sherman, K. Brown, and B. Woolf, *The Design Patterns Smalltalk Companion*, vol. Pearson Ed. 1998.

[24] R. ~T. Futrell, D. ~F. Shafer, and L. ~I. Shafer, *Quality Software Project Management*. Upper Saddle River, USA: Prentice-Hall Inc., 2002.

[25] R. Petrasch, "The definition of software quality: a practical approach," in *Proceedings of the 10th International Symposium on Software Reliability Engineering*, 1999, pp. 33–34.

BIOGRAPHIES

Hassan Almari (1971) received his MSE and ME (hons) degrees in software engineering and engineering from the Australian National University - ANU, Canberra, in 2009 and 2010, respectively, and he is currently working toward the PhD degree in software engineering field. He have over 17 years of experience in computer and operational department in RSADF, he is Lieutenant colonel in the RSADF until now. His work experience includes developing and maintaining C3 and

C4I systems. In 1998 he worked for 6 years in the PHI project, to integrate several weapons into one command and control center (C3) with RSADF and Colsa Corporation. His main research interest and current work focused on architectural level, and it is modeling and evaluation techniques.

Clive Boughton (1956) possesses a PhD in Molecular Physics from the Australian National University (ANU). Clive is a professional who possesses over thirty years of practical experience in varying roles as scientist, engineer, software engineer, consultant, academic, and project and company manager. His collective experiences have given him the opportunity to observe and contribute to commerce and defence industries using contemporary techniques, languages and management methods. His extensive industry experience led to him attaining tenure in Computer Science at ANU, where he finalised a four year degree in Software Engineering, and set up a Masters in Software Engineering. His teaching and research interests include requirements elicitation and analysis, project management, quality management and systems/software modelling and architecture. He has supervised several PhD and Masters students surrounding topics in Software Engineering.