International Journal of Software Engineering
and Knowledge Engineering
Vol. 27, No. 3 (2017) 475–498
© World Scientific Publishing Company
DOI: 10.1142/S0218194017500176

A Framework for Selecting Architectural Tactics Using Fuzzy Measures

Abdelkareem M. Alashqar[*], Hazem M. El-Bakry[†]
and Ahmad Abo Elfetouh[‡]

Information Systems Department
Faculty of Computers and Information
Mansoura University, Mansoura City 35516, Egypt
[*]*aashgar@yahoo.com*
[†]*helbakry5@yahoo.com*
[‡]*elfetouh@gmail.com*

Received 17 February 2016
Revised 6 March 2016
Accepted 22 April 2016

Software architects cannot avoid the consideration of quality attributes when designing software architecture. Architectural styles such as Layers and Client-Server are often used by architects to describe the overall structure and behavior of software. Although an architectural style affects the achievement of quality attributes, these quality attributes are directly performed by design decisions called architectural tactics. While the implementation of an architectural tactic supports a specific quality attribute, it often enhances or hurts other quality attributes in the software. In this paper, a framework for selecting the most appropriate architectural tactics according to their best achievement of the required levels of quality attributes when developing transaction processing systems is proposed. The proposed framework is based on fuzzy measures using Choquet Integral approach and takes into account the impact of architectural tactics on quality attributes, the preferences of quality attributes and the interactions between them. It can also be used to compare different potential architectures in terms of their supporting of quality attributes. The abilities and the advantages of the proposed framework are clarified via practical experiments using a case study.

Keywords: Quality attributes; software architecture; architectural tactics; fuzzy measures; Choquet Integral; Shiny by RStudio.

1. Introduction

Considering quality attributes such as reliability and efficiency is an inevitable issue when developing software products that support business transactions. Achieving quality attributes must be considered throughout design, implementation, and deployment [1]. Quality attributes are also known as non-functional requirements (NFRs), and sometimes they are more critical than functional requirements [2]. It is important to develop software in conformance with the characteristics of international quality models such as ISO9126 [3]. The defined characteristics of the ISO9126 quality model is appropriate for each type of software [3], and it is considered as the base of most of the current proposed software quality models [4].

Practically, there is a strong relationship between software architecture and quality attributes, because the achievement of quality attributes in software is closely connected with the architecture for that software [5]. Software architect often adopts an architectural style (pattern) such as Layers and Client-Server when designing software architecture. An architectural style is a general determination of a particular high-level modular decomposition of software [6]. Architectural styles support the systematic development of high-quality software with defined functional and NFRs [7]. However, an architectural style can affect a quality attribute positively or negatively [6]. Although there is a general impact of an architectural style on quality attributes, these quality attributes can be directly achieved by implementing architectural design decisions called architectural tactics or strategies (we will call them simply as "tactics" in the rest of this paper). This means that there is a direct relationship between quality attributes and tactics. Moreover, while an implementation of a tactic improves a particular quality attribute it often enhances or hinders other quality attributes within the software architecture. Consequently, the impact of tactics on quality attributes must be evaluated.

Although tactics have been introduced and examined in the literature in abstract fashion without connecting them to specific domain, the concrete implementation of these tactics is often affected by the type of software being developed. Hence, we will focus in this paper on the implementation of tactics on specific type of software which is Transaction Process System (TPS). A TPS is a computerized software that achieves and saves the daily routine transactions essential to support business such as sales order entry and patient record keeping. It also produces information for other types of systems such as Management Information Systems (MIS) [8]. TPS are always user interactive systems in which users perform asynchronous requests for service that retrieves information from database or updates the database information. The database transaction is typically a sequence of operations that is treated as atomic unit which means that all operations of this transaction need to be completed before permanently changing the database content. TPS examples includes interactive banking systems, booking systems, information systems and web applications such as e-commerce systems [2].

Selecting the most appropriate set of tactics is a challenging process since it needs an evaluation method that considers some important input parameters related to software being developed such as the impact of implementing tactics on quality attributes as stated previously, the preferences of quality attributes as determined by requirements engineers, in addition to the interactions (relationships) between these quality attributes. The selection process of a tactic among several tactics based on some quality attributes acts as a Multi-Criteria Decision-Making (MCDM) process. Based on the notion of MCDM as presented in [9], we can say that tactics act as a set of alternatives and quality attributes act as a set of criteria. The aggregation functions such as Arithmetic Mean (AM) and Weighted Arithmetic Mean (WAM) that combines several inputs to produce single output can be used in the MCDM process [10]. However, the consideration of preferences (priorities) between quality attributes cannot be performed by simple aggregation functions such as AM. Additionally, while WAM and the Analytical Hierarchical Process (AHP) [11] have the ability of considering the priorities of criteria in the MCDM process, they do not consider the interactions between these criteria. Such interactions can be performed by fuzzy measures using Choquet Integral which is a type of MCDM aggregation methods [9] that recently used in the field of evaluating software quality.

The main contribution of this paper is to propose a framework for selecting the most appropriate architectural tactics when developing TPS. Our proposed framework provides software architects and developers of TPS with the ability of defining the required quality attributes and other parameters such as their preferences as defined by software users. Based on this, the framework will help in selecting the best candidate tactics that will achieve software users' quality expectations. Furthermore, it can also be used in evaluating architectural design alternatives in terms of their supporting the best tactics before realizing them. This earlier architectural evaluation of software will reduce the development cost overheads caused by unwilling rework. The benefits and abilities of our proposed framework will be clarified using experimental results of a case study.

The rest of this paper is organized as follows: Section 2 introduces related work. Section 3 describes our proposed research methodology by designing a framework for selecting the most appropriate tactics when developing TPS. It also explains the benefits of applying Choquet Integral approach in the selection process. The quantitative evaluation of the impact of tactics on quality attributes for TPS is provided in Sec. 4. A mapping of the relationships of quality attributes between some different sources is examined in Sec. 5, where most of the results are drawn based on current literature. Experiments and results that show the benefits of our framework are drawn in Sec. 6. Section 7 finishes the paper with the conclusion and future work.

2. Related Work

In this section, we provide a discussion of tactics in addition to the current techniques used in the selection process of appropriate tactics when designing software architecture.

A refined list of tactics, their classifications and their role on the achievement of quality attributes are found in [1]. In this source, the authors have defined seven categories of tactics which are availability, modifiability, performance, usability, testability, interoperability and security. However, other types of tactics are also found in literature. For instance, the authors in [12] have defined a list of statics pertinent to safety, and the authors in [13] have defined tactics relevant to reliability. Generally, each category of tactics devoted to a particular quality attribute is often represented as a hierarchy of tactics such as the availability tactics that depicted

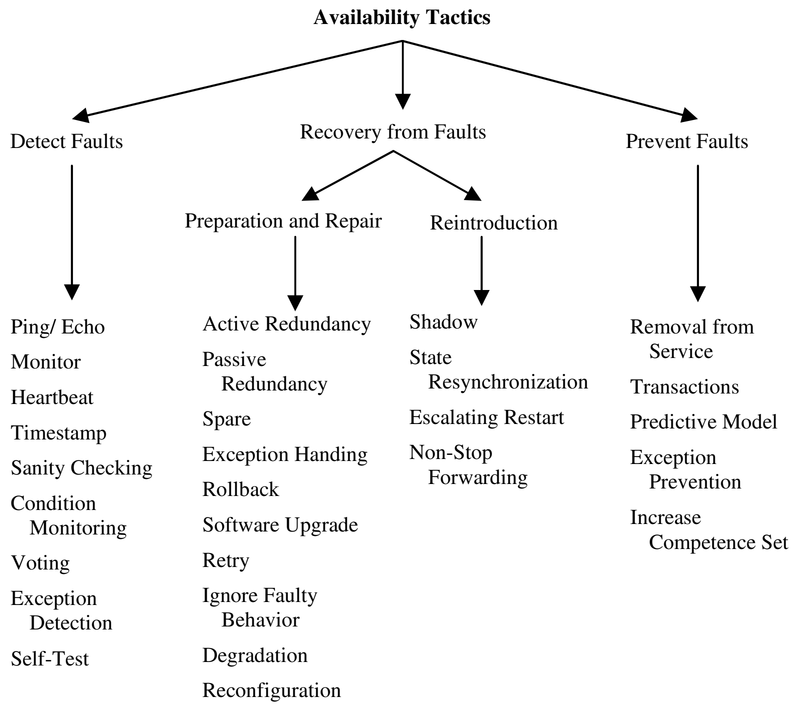


Fig. 1. Hierarchy of availability tactics.

in Fig. 1. Although, tactics have been often introduced abstractly in the literature without linking them to specific type of software, we will examine tactics in our proposed selection process when developing TPS that support business operations.

To our knowledge, there is no significantly comprehensive framework for selecting tactics found in the literature. Architects often depend on their understanding of how the current architecture will meet the required quality requirements when selecting the best tactics. The authors in [1] introduced two main approaches for selecting tactics which are model-based and expert-based. In model-based approach, the architect first enumerates quality parameters of the model, then for each parameter a list of tactics is enumerated. However, an apparent shortcoming related to this approach is that some quality attributes such as usability cannot be architecturally modeled. In expert-based approach, the architect reviews the current architecture or asks some experts to define a list of tactics that will meet the required quality attributes. However, the selection list of tactics in this approach depends on specialists' experiences and it is qualitative in nature.

While the impact of tactics in the process of architectural design is considered an important step in the selection process, most of the previous work focused on the impact of tactics on architectural styles. However, limited works have examined the impact of tactics on quality attributes. With regard to architectural styles and tactics, the authors in [6] developed a model for the interaction of styles and tactics to help software architects to understand the impact of these tactics on the overall system structure. They followed the in-depth analysis approach to study the relationships between tactics and architectural styles. Extensions to this work [14, 15] have been performed by some researchers who provide quantitative evaluation of the impact of architectural styles on quality attributes by studying the relationships between architectural styles, tactics and quality attributes.

The impact of tactics on quality attributes was studied by [16] where the authors have provided evaluation of the impact of some tactics on some quality attributes of embedded systems using nine-point scale approach. Narrower results were introduced in [14] where the authors have showed that some security tactics have negative impact on some performance tactics and vice versa.

For more accurate and convenient results of the selection process of tactics, the interactions among quality attributes should be considered. Nonetheless, the interactions between quality attributes have often been examined separately in some research studies such as [17, 18]. The interactions among quality attributes can always be considered using fuzzy measures. Some existing research work such as [19, 20] have applied fuzzy measures using Choquet Integral approach as an MCDM aggregation method for evaluating quality attributes without considering neither architectural styles nor tactics.

Our work differs from previous work in that it proposes a new framework for selecting the best tactics when developing TPS, since according to our knowledge, no such framework exists. Furthermore, the proposed framework adopts fuzzy measures techniques using Choquet Integral method. Choquet Integral is considered an effective quantitative MCDM approach because it considers the priorities of criteria as well as the interaction between them.

3. Research Methodology

The main goal of this paper is to introduce a framework for selecting the best architectural tactics in order to implement them in the potential software architecture of TPS. To provide more accurate and more convenient results, we believe that the selection process will depend on the following important types of data:

(1) quality attributes and their preferences as determined by software users.
(2) the interactions between quality attributes.
(3) the impact of tactics on quality attributes.
(4) supplementary information (e.g. rank of tactics) as defined by decision maker.

These four types of data must be provided as quantitative data inputs to the selection process of our proposed framework. The first type of data concerns the required quality attributes and their preferences as defined by software users. Practically, requirements engineers and software architects work with users to perform this.

The second type of data concerns the interactions between quality attributes. Ideally, it is not convenient to achieve a quality attribute without considering its effect on other set of quality attributes that will be totally achieved in the software being developed. This is because of the existence of interactions between these quality attributes, and hence these interactions must be defined. Regarding the relationships between quality attributes, most of research work have stated that an achievement of one quality attribute may improve or hinder other quality attributes in the software. The interactions among quality attributes will be studied in more details in Sec. 5 of this paper.

The third type of data is related to the potential tactics that will be adopted in software architecture to achieve the required quality attributes, in addition to tactics' impacts on these quality attributes. These impacts must be quantitatively evaluated and we will perform this in the next section of this paper.

The fourth type of the above data is about the additional information that may be provided by the decision maker in order to add more flexibility when selecting an appropriate method among the variety of Choquet Integral methods. We will discuss these methods in the following section of the paper.

3.1. *Fuzzy measures using Choquet Integral approach*

Based on the abovementioned four types of data that will be considered as main input parameters supplied to our proposed framework for selecting the best tactics, the simple AM aggregation function cannot be used because it does not take into account the preferences or the interactions between quality attributes. Moreover, while WAM and AHP consider the preferences of quality attributes, they do not consider the interactions between these quality attributes. So the powerful method that overcomes these shortcomings is Choquet Integral method which is a type of fuzzy measures technique. Choquet Integral method has been described by some researchers. To illustrate its work, if we have a set of criteria $X = \{x_1, x_2, x_3\}$, then the fuzzy measures (also called capacity) for X is represented as follows [9, 21]:

$$\mu(\{\phi\}), \mu(\{1\}), \mu(\{2\}), \mu(\{3\}), \mu(\{1,2\}), \mu(\{1,3\}), \mu(\{2,3\}) \quad \text{and} \quad \mu(\{1,2,3\}).$$

As seen, fuzzy measures differs from other traditional aggregation functions in that it permits for assigning weights for the subsets of criteria. For example, if we have $A \subseteq X$, then the number $\mu(A)$ is considered the weight or the importance of A. The Choquet Integral with respect to a fuzzy measure μ is calculated as follows [9, 21]:

$$C_\mu(x) = \sum_{i=1}^{n} x_{(i)} \left[\mu(\{j | x_j \geq x_{(i)}\}) - \mu(\{j | x_j \geq x_{(i+1)}\})\right]. \tag{1}$$

Choquet Integral approach can also be used to calculate the Shapley value which represents the specific contribution score of each criterion within a set of criteria in the MCDM process. It is important to denote that, the sum of Shapley values of criteria must be equal to 1. Shapley values help the decision maker to get insight into the importance of each quality attribute in the software being developed. Moreover, Choquet Integral method has the ability of computing the interactions behavior (interaction index) of groups of criteria within a set of criteria. A k-additive notion is often linked with interaction index where k ranges from 1 to the total number of criteria. When k equals 2, the interactions are limited to each pair of criteria, whereas the interactions between all of subsets of criteria can be considered when k value is maximized to the total number of criteria [22].

A more important issue regarding our work is the existence of variety of Choquet Integral methods. The authors in [21] have discussed some of these methods. They argued that most of Choquet Integral methods for capacity identification proposed in the literature can be stated as optimization problems. They differ according to their objective function and the preferential information they require as inputs. A discussion for some of these methods is provided next [21].

3.1.1. *Least-Squares based approaches*

Least-Squares (LS) approaches are one of the first proposed approaches which are classified as a generalization of linear regression. LS approaches often require additional knowledge about the desired overall evaluations of the available alternatives provided by decision maker. The aim of these methods is to minimize the average quadratic distance between the overall utilities computed by means of the Choquet Integral and the desired overall scores provided by the decision maker. Heuristic Least Mean Squares (HLMS) is a variant of LS approach used to avoid the use of quadratic solvers. HLMS is based on a gradient approach starting from a decision maker defined capacity μ which is called the initial capacity representing the DM's prior idea.

3.1.2. *Maximum split approach*

This approach is based on Linear Programming (LP). The main idea of this approach is to maximize the minimal difference between the overall utilities of alternatives that have been ranked by the decision maker. This ranking of the alternatives is called partial weak order.

3.1.3. *Minimum variance and minimum distance approaches*

The main idea of the Minimum Variance (MV) method is to support the least specific capacity (if it is found) compatible with the initial preferences of the decision maker. This method is considered an equivalent to maximum entropy approach. The Minimum Distance (MD) approach is used in the absence of clear requirements and practically implemented using the principle of MD, where three quadratic distances should be studied.

The aforementioned Choquet Integral methods add more importance to our proposed selection process of tactics especially when there is more supplementary information need to be provided by a decision maker such as the overall rank and the overall utilities of tactics. For instance, if two tactics T_1 and T_2 that are evaluated based on six quality attributes have the same mean value, the decision maker can rank these tactics as $T_1 > T_2$ which means that T_1 is more favored to the decision maker than T_2. Or the decision maker prefers one tactic over the other by assigning a value, called overall utility, which is greater or lower than the mean value within specific threshold.

Additionally, the existence of various Choquet Integral methods permits for more flexibility in selecting an appropriate one of them in the MCDM process based on the available data. For example, least squares approaches such as LS and HLSC are used only when overall utilities are provided by the decision maker. In contrast, other approaches such as LP, MV and MD are used in the absence of the overall utilities of the alternatives. Instead, such methods need ranking of available alternatives which is called partial weak order over these alternatives.

Moreover, when software users cannot provide with specific preferences of quality attributes, Choquet Integral methods such as LP, MV and MD help them in providing information related to the preorder of these quality attributes (Shapely Preorder). They can define whether two or more quality attributes have the same importance and whether a quality attribute is more important or less important than another.

3.2. *A framework for selecting the most appropriate tactics*

Based on the data needed for the selection process of tactics in addition to the ability of Choquet Integral method, we will introduce the steps of our proposed framework for selecting the most appropriate tactics as depicted in Fig. 2. As seen from this figure, the framework encompasses the following three main steps (see Fig. 2).

3.2.1. *Quality attributes parameters*

The objective of this step is to prepare quality attributes parameters. Two main outcomes are produced; the first is the quality attributes and their parameters as defined by requirements engineers. These data are used as main parameter for the next step of deriving the potential tactics. The second output is the types of the interactions between quality attributes which will be used as main parameter of the computation process of the final step of our proposed framework. The interactions between quality attributes will be defined in Sec. 5 of this paper.

3.2.2. *Tactics and their impacts on quality attributes*

This step aims to prepare all parameters that concerns tactics. Firstly, a comprehensive checklist of tactics are derived based on the defined quality attributes prepared in step one. The content of this checklist is determined based on the architect's understanding and intuition of how the required quality attributes will be met in the potential software architecture. Because of the close relationship between tactics and

April 26, 2017 10:52:45am WSPC/117-IJSEKE 1750017

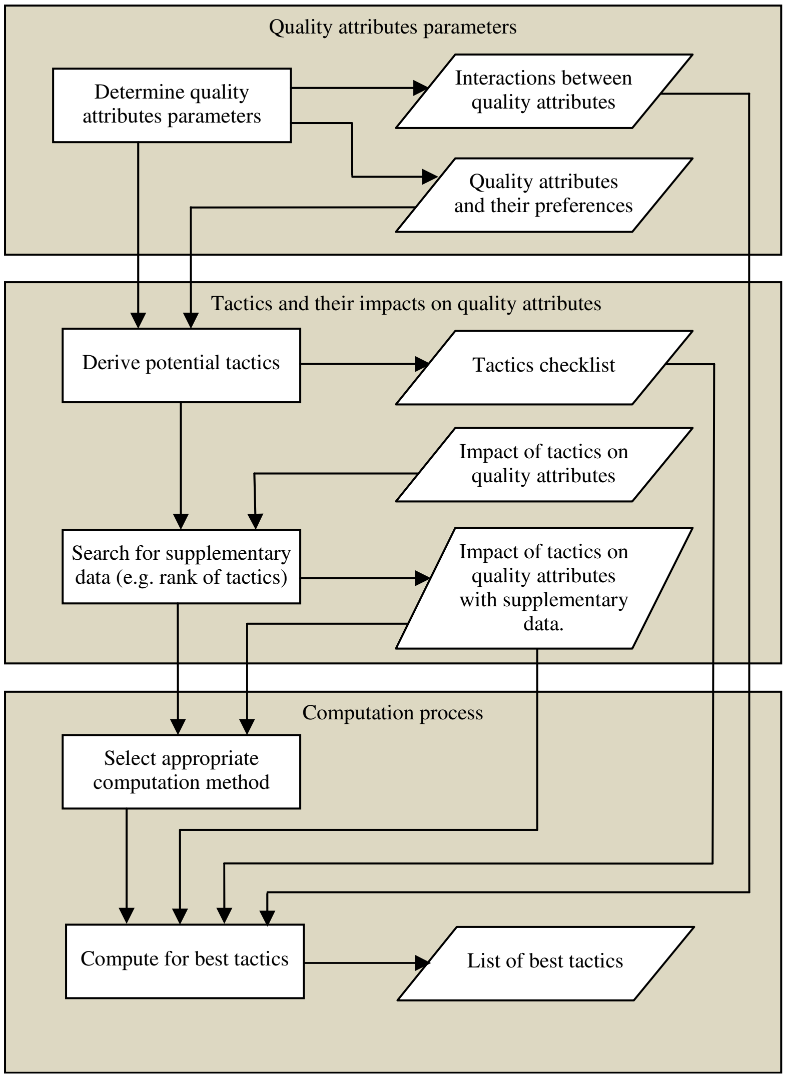

Quality attributes parameters
Determine quality attributes parameters
Interactions between quality attributes
Quality attributes and their preferences
Tactics and their impacts on quality attributes
Derive potential tactics
Tactics checklist
Impact of tactics on quality attributes
Search for supplementary data (e.g. rank of tactics)
Impact of tactics on quality attributes with supplementary data.
Computation process
Select appropriate computation method
Compute for best tactics
List of best tactics

Fig. 2. The main components of the framework for selecting the most appropriate tactics.

quality attributes, the impact of tactics on quality attributes will be adequately evaluated. These impact data are considered as main parameter of the overall selection process. Additionally, it will guide the architect (or a decision maker) to provide supplementary data such as the overall rank of tactics, in order to be used in the Choquet Integral computation process in the next step of the proposed framework. This will produce more comprehensive data which is considered as main parameter to the next step. We will provide evaluation of the impact of tactics on quality parameters with relevant to TPS in the next section of this paper.

April 26, 2017 10:52:46am WSPC/117-IJSEKE 1750017

3.2.3. *Computation process*

This step aimed at selecting the best tactics that will be implemented in the potential software architecture while developing TPS. The main input parameters of this step are quality attributes and their interactions, the tactics checklist in addition to the comprehensive data of the impact of tactics on quality attributes. These input parameters have been produced in the previous two steps.

4. Impact of Tactics on Quality Attributes

For the main purpose of our proposed framework for selecting the most appropriate tactics, we provide in this section the evaluation of the impact of several tactics on some quality attributes when developing TPS. To perform this, each tactic must be analyzed against each quality attribute. The affected quality attributes is limited to those proposed by the ISO9126 quality model, which are functionality, reliability, usability, efficiency, maintainability and portability. Most of previous work regarding the effect of tactics in software architecture have been drawn based on researchers' own understanding of these tactics such as [6, 14, 15, 23]. However, the authors in [16] have carried out the impact results based on experts and their own evaluation in the domain of embedded systems.

Since the evaluation of the impact of tactics on quality attributes depends on experience and objective analysis, we have developed a research survey and sent it to three experts who are working in academia and have considerable industrial experience in the field of developing TPS. Each participant has been asked to evaluate the impact of each tactic related to availability, performance and modifiability on the six quality attributes of the ISO9126 quality model when developing TPS. In addition, the participants have been asked to provide justification for their evaluations. The evaluation method was explained carefully to each participant. Moreover, supplementary data including descriptions of the ISO9126 quality attributes and descriptions of the examined tactics were provided to the participants. We have requested that the participants will use a five-point scale for the impact evaluation, where "+2" means a tactic strongly improves a quality attribute, "+1" means some improvement, "−2" means a tactic strongly hurts a quality attribute, "−1" means some hurt, and "0" means a tactic neither improve nor hurts a quality attribute. Since the survey is time consuming because of the large number of evaluation judgments, each participant has been taken 10 days in order to have sufficient time at home to judge for the evaluations.

Tables 1–3 show the impact of 48 tactics relevant to performance, availability and modifiability respectively on the six quality attributes of the ISO9126 model in addition to the inter-rater reliability (level of agreement) between the participants' judgments as shown in the rightmost column of each one of these tables.

We have used the Kendall's Coefficient of Concordance (KCC) provided by irr package [25] to compute the inter-rater reliability (level of agreement) between raters

Table 1. Impact of performance tactics on ISO9126-based quality attributes.

Tactics/Attributes	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Inter-rater reliability
Control resource demand							
Manage sampling rate	0	+2	0	+2	0	0	0.67
Limit event response	0	+1	−1	+2	−1	−1	0.67
Prioritize events	−1	+1	0	+2	−1	−1	0.97
Reduce overhead	0	+2	+1	+2	−1	−1	0.76
Bound execution times	−2	0	0	+2	0	0	0.76
Increase resource Efficiency	+1	+1	+1	+2	−1	0	0.86
Manage resources							
Increase resources	0	+1	+1	+2	0	0	0.84
Introduce concurrency	+1	−1	0	+2	−1	−1	0.82
Maintain multiple copies of computations	0	0	0	+2	−2	−1	0.84
Maintain multiple copies of data	0	+1	0	+2	−2	−1	0.95
Bound queue sizes	−1	−1	0	+2	−1	0	0.53
Schedule resources	−1	−1	0	+2	−2	−1	0.63

Table 2. Impact of availability tactics on ISO9126-based quality attributes.

Tactics/Attributes	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Inter-rater reliability
Detect faults							
Ping/Echo	0	+2	0	0	−1	−1	0.92
Monitor	−1	+2	0	−1	−1	−1	0.64
Heartbeat	0	+2	0	−1	−1	−1	0.73
Time stamp	0	+2	0	−1	−1	0	0.78
Sanity checking	+1	+2	+1	−1	0	0	0.81
Condition monitoring	+1	+2	+1	−1	0	0	0.80
Voting	0	+2	0	−1	−1	−1	0.90
Exceptions detection	+1	+2	+1	−1	−1	−1	0.87
Selftest	+1	+2	+1	−1	0	0	0.95
Preparation and repair							
Active redundancy	−1	+2	0	−2	−1	−1	0.65
Passive redundancy	−1	+2	0	−1	−1	−1	0.68
Spare	0	+2	−1	−1	0	0	0.66
Exception handling	+1	+2	+1	−1	−1	−1	0.78
Rollback	−2	+2	0	−1	−2	0	0.67
Software upgrade	+1	+2	+1	−1	−1	−1	0.68
Retry	−1	+2	−1	−2	−1	0	0.52
Ignore faulty behavior	0	+2	0	−1	0	0	0.67
Degradation	−2	+2	+1	0	−1	0	0.60
Reconfiguration	−2	+2	−1	−1	−1	−1	0.58
Reintroduction tactics							
Shadow	−2	+2	0	−1	−1	−1	0.67
State resynchronization	−1	+2	0	−1	−1	0	0.55
Escalating restart	−2	+2	0	0	−1	−1	0.65
Nonstop forwarding	+1	+2	0	0	−1	−1	0.61
Prevent Fault							
Removal from service	−2	+2	−1	0	−1	0	0.63
Transactions	+1	+2	0	0	−2	0	0.62
Predictive model	−1	+2	0	−1	−1	−1	0.71
Exception prevention	0	+2	0	−2	−1	0	0.65
Increase competence set	−1	+2	−1	−1	−1	−1	0.61

Table 3. Impact of modifiability tactics on ISO9126-based quality attributes.

Tactics/Attributes	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability	Inter-rater reliability
Reduce size of module							
Split module	+2	0	0	−1	+2	+1	0.76
Increase cohesion							
Increase semantic coherence	+2	+1	0	−1	+2	+2	0.73
Reduce coupling							
Encapsulate	+1	+1	0	0	+2	0	0.81
Use an intermediary	0	+1	+1	0	+2	+2	0.85
Restrict dependencies	+1	0	0	+1	+2	+1	0.76
Refactor	+1	0	+1	0	+2	0	0.83
Abstract common services	0	+1	+1	0	+2	+1	0.72
Defer binding							
Defer binding	0	−1	0	−1	+2	+1	0.83

(survey participants). KCC is considered an appropriate method of inter-rater reliability for our evaluations because it can be used to quantify the extent of agreement among three raters or more with respect to their ordinal ranking of the same group of subjects (tactics) [24, 25]. KCC produces an agreement value called w which ranges from 0 (no agreement) to 1 (complete agreement). The level of agreement value between 0.40 and 0.75 may represent fair to good agreement beyond chance, and it may represent an excellent agreement if it is greater than 0.75 [26]. Rater interchangeability can be guaranteed if inter-rater reliability is high. In this case, the raters can be used interchangeably without the researcher worries about the evaluation being affected by a rater factor. Moreover, the evaluated subjects can be used with confidence without asking what rater produced them [24]. The inter-rater reliability between our survey's participants (evaluators) is considerably high for most of the examined tactics as shown in Tables 1–3.

5. Interactions Between Quality Attributes

The interactions between quality attributes have been studied by different researchers from different views such as literature, academic and industry [27]. Most of previous works regarding the relationships between quality attributes have defined that the type of a relationship between two quality attributes can be positive "+", negative "−" or neutral "0". The positive relationship means that an improvement of one quality attribute will also improve the other. The negative relationship means that an enhancement of one quality attribute will hurt the other. Whereas the neutral relationship means that there is no relationship between the two quality attributes. In the case of negative relationship, a conflict happens between two attributes because it is very hard to achieve both of them simultaneously. Hence, it is necessary to make a trade-off in order to find a suitable balance between the two attributes. Whereas the positive relationships indicate that the two attribute are synergetic because they have much in common and are achieved in similar ways [28].

With the context of ISO9126 quality attributes, we have surveyed the literature and found relationships from six sources as shown in Table 4. Where "P" indicates a positive value, "N" indicates a negative value, and "0" indicates a neutral value. The empty cells of the table indicate that there is no assigned value found in the surveyed sources. Note that the number of relationships for each pair of the six quality attributes of ISO9126 model is calculated as $N * (N - 1)/2$ which equals to 15.

We have discovered that not all of the surveyed quality attributes examined in the sources that listed in Table 4 have identical classifications (names) to those proposed by the ISO9126 model. So, we have adopted a policy for deciding if there is equivalence between two quality attributes that have different classifications. Let x be a quality attribute introduced in IOS9126 quality model, and y be a quality attribute listed in the surveyed sources that has different classification to x, if y has similar meaning to x or its sub-characteristics, then we decided that y is equivalent to x. However, this equivalence policy might be inaccurate because some of the surveyed

Table 4. Mapping table to define ISO9126-based quality attributes relationships based on six different sources.

			Interaction types as defined in literature						Final
ID	Quality attribute	Versus Quality attribute	[17]	[18]	[27]	[29]	[30]	[31]	value
1	**Functionality**	**Reliability**	**P**			**P**			**P**
2	**Functionality**	**Usability**		**P**		**P**			**P**
3	**Functionality**	**Efficiency**	**N**	**N**		**N**	**N**		**N**
4	**Functionality**	**Maintainability**	**P**			**P**		**P**	**P**
5	Functionality	Portability	P	P	N	0			P
6	**Reliability**	**Usability**				**P**	**P**	**P**	**P**
7	Reliability	Efficiency				**0**			**0**
8	Reliability	Maintainability	N			P	N	N	N
9	*Reliability*	*Portability*			*P*	*0*			*P*
10	**Usability**	**Efficiency**	**N**		**N**	**N**	**N**		**N**
11	*Usability*	*Maintainability*	*P*			*0*			*P*
12	*Usability*	*Portability*		*N*		*0*			*N*
13	**Efficiency**	**Maintainability**	**N**			**N**			**N**
14	**Efficiency**	**Portability**	**N**	**N**		**N**			**N**
15	Maintainability	Portability	P			P	N		P

sources do not provide definitions for the examined attributes. Based on our policy of equivalence, we have considered the following:

- Each of "correctness" examined in [17, 30, 31], and "interoperability" examined in [18] are equivalent to functionality.
- The "performance" examined in [18] is equivalent to efficiency.
- The "flexibility" examined in [17, 31] is equivalent to maintainability.

However, most of the surveyed sources have examined only some of the 15 relationships. We have noticed that only one pair relationship, {reliability, efficiency}, has been determined by only one investigated source. Additionally, we have noticed that the other 14 surveyed sources have agreed completely on only eight pairs' relationships (highlighted by bold text in Table 4). For three of the remaining six pairs, we have judged on their relationships considering the majority as follows:

- A positive "P" value is assigned to the pairs {functionality, portability} and {maintainability, portability}.
- A negative "N" value is assigned to the pairs {reliability, maintainability}.

Still the majority cannot be applied to three remaining cases (highlighted in italic text in Table 4), because each of them has only two different values. Since each one of them has a neutral "0" relationship in addition of considering that a relationship must exist, we have judged on their relationships as follows:

- A positive "P" value is assigned to the pairs {reliability, portability} and {usability, maintainability}.
- A negative "N" value is assigned to the pair {usability, portability}.

The rightmost column of Table 4 shows the final assigned value for each pair relationship of the 15 relationships regarding the ISO9126-based quality attributes.

With respect to the interaction index concept of Choquet Integral method, if two quality attributes have a positive relationship (synergic) then it is said that they act in complementary way, whereas if they have a negative relationship (conflicting) then it is said that they act in substitutive way. The rational of this is that the software architect's preferences will increase if both of synergic quality attributes are achieved while these preferences will not increase even if both of the conflicting quality attributes are satisfied.

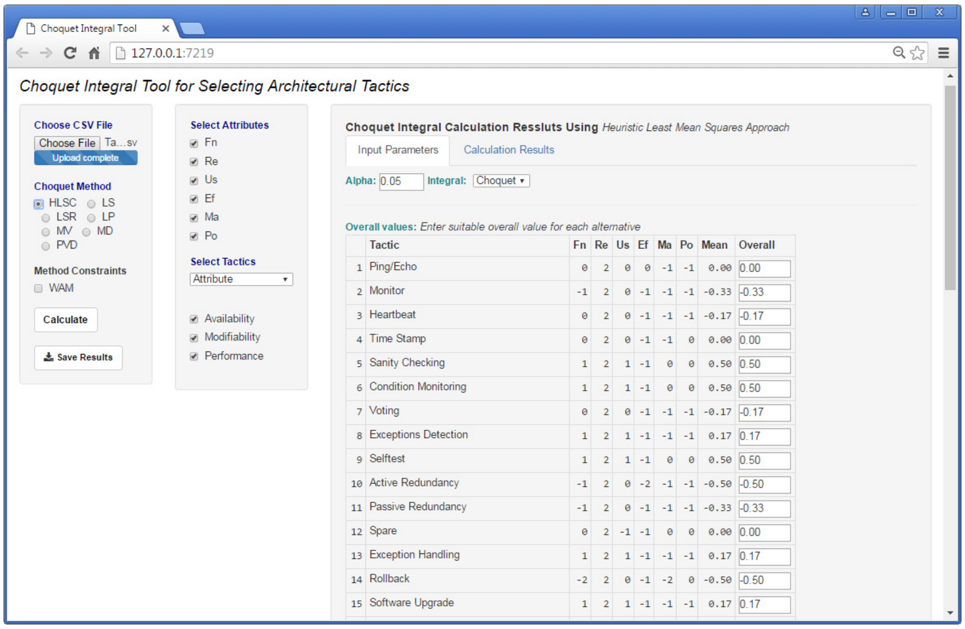
6. Experiments and Results

This section provides an experimentation of the advantages and the abilities of our proposed framework for selecting the most appropriate tactics when developing TPS using the following case study:

A medical information system [2] *maintains information about patients suffering from mental health problems and the treatments they have received. Majority of patients do not require devoted hospital treatment but need to attend specialist clinics frequently where they can meet a doctor who has detailed knowledge of their problems. The clinics may be held in local medical practices or community centers as well as in hospital, so the system should facilitate the patient attendance in any place. Although the system has been designed with a centralized database of patient information in hospital it also has been designed to run on clinics. For such design, users can connect with centralized database using secure network or can access a copy on their local database. A clinic also can exchange data with other clinical information system. Patient information confidentiality is an issue in the system.*

6.1. *Supporting tool*

We developed a user interactive tool using RStudio Shiny Web Application Framework for R [32]. This tool utilizes the Choquet Integral approaches discussed in Sec. 3 of this paper, in order to be used in our selection process of tactics depending on given parameters. It is mentioned here that these approaches are essentially provided as functions by the Kappalab package [33] where each function takes predetermined input parameters (arguments) in order to create a type of Mobius capacity (fuzzy measure). Using this tool, the data related to tactics such as their impact on quality attributes can be read from stored files of type Comma Separated Values (CSV). Where the rows of the CSV file represent the tactics and columns represent the quality attributes. The decision maker can use our developed tool easily for selecting the suitable Choquet Integral method, uploading data related to quality attributes and tactics, and entering any supplementary data as shown in Fig. 3. As seen from the figure, the main screen of the tool includes two tabbed sub-windows, the first is named "Input Parameters" to permit for entering input parameters such



Choquet Integral Tool for Selecting Architectural Tactics

Choose CSV File Ta...sv

Choquet Method [✓ HLSC] LS LSR LP MV MD PVD

Method Constraints WAM

Select Attributes ✓ Fn ✓ Re ✓ Us ✓ Ef ✓ Ma ✓ Po

Select Tactics Attribute

✓ Availability ✓ Modifiability ✓ Performance

Choquet Integral Calculation Ressluts Using *Heuristic Least Mean Squares Approach*

Alpha: 0.05 **Integral:** Choquet

Overall values: *Enter suitable overall value for each alternative*

	Tactic	Fn	Re	Us	Ef	Ma	Po	Mean	Overall
1	Ping/Echo	0	2	0	0	-1	-1	0.00	0.00
2	Monitor	-1	2	0	-1	-1	-1	-0.33	-0.33
3	Heartbeat	0	2	0	-1	-1	-1	-0.17	-0.17
4	Time Stamp	0	2	0	-1	-1	0	0.00	0.00
5	Sanity Checking	1	2	1	-1	0	0	0.50	0.50
6	Condition Monitoring	1	2	1	-1	0	0	0.50	0.50
7	Voting	0	2	0	-1	-1	-1	-0.17	-0.17
8	Exceptions Detection	1	2	1	-1	-1	-1	0.17	0.17
9	Selftest	1	2	1	-1	0	0	0.50	0.50
10	Active Redundancy	-1	2	0	-2	-1	-1	-0.50	-0.50
11	Passive Redundancy	-1	2	0	-1	-1	-1	-0.33	-0.33
12	Spare	0	2	-1	-1	0	0	0.00	0.00
13	Exception Handling	1	2	1	-1	-1	-1	0.17	0.17
14	Rollback	-2	2	0	-1	-2	0	-0.50	-0.50
15	Software Upgrade	1	2	1	-1	-1	-1	0.17	0.17

Fig. 3. Main screenshot of the Choquet Integral tool for selecting tactics.

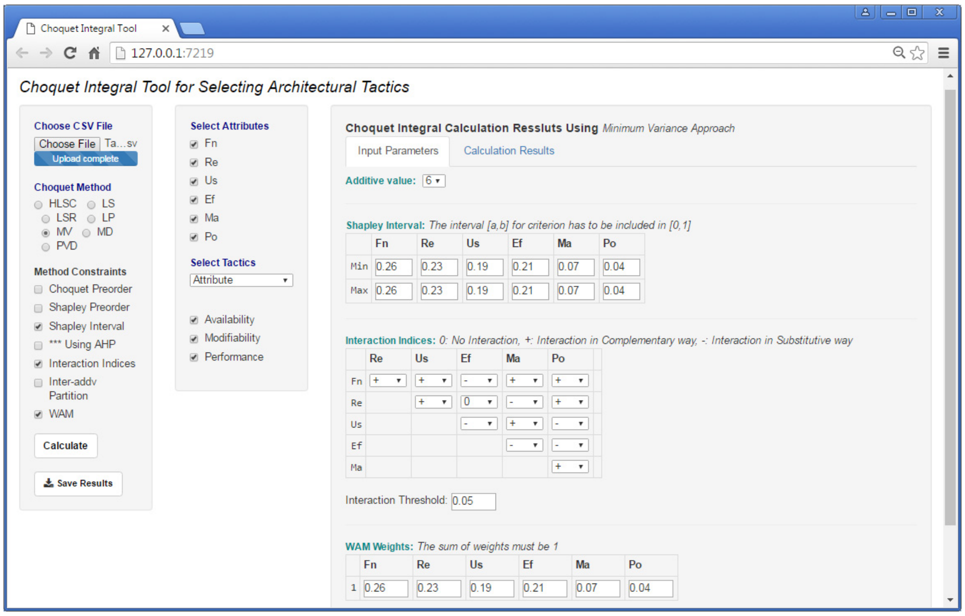
as the preferences of quality attributes, and the second is named "Calculation Results" for presenting the computation results. The tool also has an ability of storing the experimental results into a file of type CSV.

6.2. *Quality attributes and their preferences for the case study*

On the assumption that the requirements engineer and the software architect who work with software users, determined that the required software quality attributes are functionality, reliability, usability, efficiency, maintainability and portability, and the assigned Shapley values (preferences) for these quality attributes are 0.26, 0.23, 0.19, 0.21, 0.07 and 0.04, respectively. Additionally, the determined types of interactions between these quality attributes are identical to those stated in Table 4. Figure 4 shows the entry window of the Shapley values and the interactions between quality attributes as well. Note that each quality attribute is presented by our tool as shown in Fig. 4 using its first two characters because of space limitation of the user interface screen.

6.3. *Determine candidate tactics*

Based on the given information provided in the above step in addition to the domain understanding of the proposed software, we have selected availability, performance and modifiability tactics as candidate tactics so that they will be ranked with the


Choquet Integral Tool

127.0.0.1:7219

Choquet Integral Tool for Selecting Architectural Tactics

Choose CSV File
Choose File Ta...sv
Upload complete

Choquet Method
HLSC [ ] LS [ ] LSR [ ] LP [ ] MV [●] MD [ ] PVD [ ]

Method Constraints
- [ ] Choquet Preorder
- [ ] Shapley Preorder
- [✓] Shapley Interval
- [ ] *** Using AHP
- [✓] Interaction Indices
- [ ] Inter-addv Partition
- [✓] WAM

Calculate

Save Results

Select Attributes
- [✓] Fn
- [✓] Re
- [✓] Us
- [✓] Ef
- [✓] Ma
- [✓] Po

Select Tactics
Attribute

- [✓] Availability
- [✓] Modifiability
- [✓] Performance

Choquet Integral Calculation Ressluts Using *Minimum Variance Approach*

Input Parameters | Calculation Results

Additive value: 6

Shapley Interval: *The interval [a,b] for criterion has to be included in [0,1]*

	Fn	Re	Us	Ef	Ma	Po
Min	0.26	0.23	0.19	0.21	0.07	0.04
Max	0.26	0.23	0.19	0.21	0.07	0.04

Interaction Indices: *0: No Interaction, +: Interaction in Complementary way, -: Interaction in Substitutive way*

	Re	Us	Ef	Ma	Po
Fn	+	+	-	+	+
Re		+	0	-	+
Us			-	+	-
Ef				-	-
Ma					+

Interaction Threshold: 0.05

WAM Weights: *The sum of weights must be 1*

	Fn	Re	Us	Ef	Ma	Po
1	0.26	0.23	0.19	0.21	0.07	0.04

Fig. 4. The selected Choquet Integral method and the input parameters used for the case study.

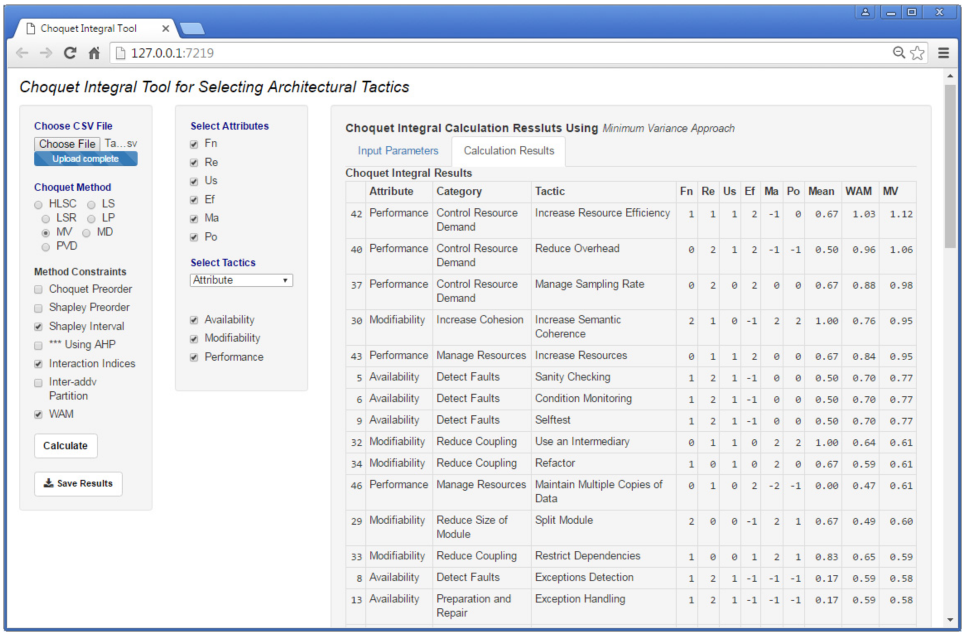
assistance of our developed tool. This rank gives the architect an insight into the most appropriate tactics that will be implemented in the potential software for the case study. The selected categories of tactics using our developed tool are shown in Fig. 4.

6.4. *Select Choquet Integral method and determine any additional parameters*

Based on the information provided in the two previous steps, one method from LP, MV or MD can be selected. Note that, as mentioned before, these methods are suitable when there are no overall utilities provided for the experimented alternatives (tactics). However, providing overall utility for each tactic when there is large number of tactics, such as in this case is considered a tedious work. Hence, the MV method was selected for our experiments where the k-additive value was set to 6 in order to get the best performance. Furthermore, a value of 0.05 was assigned to the interaction threshold indifference as shown in Fig. 4. This threshold indicates the absolute value of the interaction index that will be considered with significance from zero [33].

6.5. *Compute for the best tactics*

The computation results can be performed by pressing the "Calculate" button from the application tool window. As shown in Fig. 5, the tactics are presented in



Choquet Integral Tool for Selecting Architectural Tactics

Choose CSV File
Choose File Ta…sv
Upload complete

Choquet Method
HLSC LS LSR LP [✓ MV] MD PVD

Method Constraints
- [ ] Choquet Preorder
- [ ] Shapley Preorder
- [x] Shapley Interval
- [ ] *** Using AHP
- [x] Interaction Indices
- [ ] Inter-addv Partition
- [x] WAM

Calculate

Save Results

Select Attributes
- [x] Fn
- [x] Re
- [x] Us
- [x] Ef
- [x] Ma
- [x] Po

Select Tactics
Attribute
- [x] Availability
- [x] Modifiability
- [x] Performance

Choquet Integral Calculation Ressluts Using *Minimum Variance Approach*

Input Parameters | Calculation Results

Choquet Integral Results

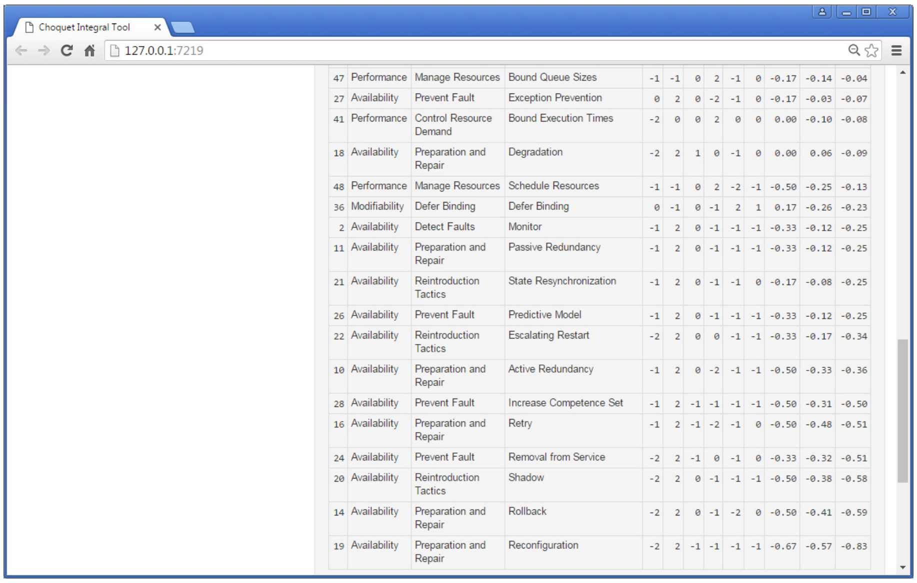
	Attribute	Category	Tactic	Fn	Re	Us	Ef	Ma	Po	Mean	WAM	MV
42	Performance	Control Resource Demand	Increase Resource Efficiency	1	1	1	2	-1	0	0.67	1.03	1.12
40	Performance	Control Resource Demand	Reduce Overhead	0	2	1	2	-1	-1	0.50	0.96	1.06
37	Performance	Control Resource Demand	Manage Sampling Rate	0	2	0	2	0	0	0.67	0.88	0.98
30	Modifiability	Increase Cohesion	Increase Semantic Coherence	2	1	0	-1	2	2	1.00	0.76	0.95
43	Performance	Manage Resources	Increase Resources	0	1	1	2	0	0	0.67	0.84	0.95
5	Availability	Detect Faults	Sanity Checking	1	2	1	-1	0	0	0.50	0.70	0.77
6	Availability	Detect Faults	Condition Monitoring	1	2	1	-1	0	0	0.50	0.70	0.77
9	Availability	Detect Faults	Selftest	1	2	1	-1	0	0	0.50	0.70	0.77
32	Modifiability	Reduce Coupling	Use an Intermediary	0	1	1	0	2	2	1.00	0.64	0.61
34	Modifiability	Reduce Coupling	Refactor	1	0	1	0	2	0	0.67	0.59	0.61
46	Performance	Manage Resources	Maintain Multiple Copies of Data	0	1	0	2	-2	-1	0.00	0.47	0.61
29	Modifiability	Reduce Size of Module	Split Module	2	0	0	-1	2	1	0.67	0.49	0.60
33	Modifiability	Reduce Coupling	Restrict Dependencies	1	0	0	1	2	1	0.83	0.65	0.59
8	Availability	Detect Faults	Exceptions Detection	1	2	1	-1	-1	-1	0.17	0.59	0.58
13	Availability	Preparation and Repair	Exception Handling	1	2	1	-1	-1	-1	0.17	0.59	0.58

Fig. 5. Best ranked tactics according to their MV values.

descending order based on their MV values, i.e. the tactics that got the highest scores come first. In other words, these tactics are considered the best candidate tactics for the proposed software architecture. For example, "Increase Resource Efficiency", "Reduce Overhead" and "Manage Sampling Rate" performance tactics are the best three candidate tactics, while "Selftest" availability tactic comes in the eighth and "Restrict Dependencies" modifiability tactic comes in the thirteenth. It is also noted that all of these tactics positively support the potential software architecture. In contrast, "Shadow", "Rollback" and "Reconfiguration" availability tactics got the worst rank and they negatively support the potential architecture as shown in Fig. 6.

The results also provide us with insight in how Choquet Integral results differ from AM and WAM as shown in Fig. 7 which visually shows this difference for the top 20 ranked tactics with respect to MV computation. It is clear that there is a significant difference between WAM results and MV results although the two methods were tested against the same candidate tactics, same quality attributes and their preferences. This emphasizes the role of the interactions among quality attributes in the computation process. Figure 7 also shows that there is a big difference between the results of WAM and MV on the one hand and the results of AM on the other hand because the latter does not consider the preferences among quality attributes and the interactions between them as well.

To ensure these beneficial insights, we have repeated the previous computations using the same input parameters but using different preferences between quality



47	Performance	Manage Resources	Bound Queue Sizes	-1	-1	0	2	-1	0	-0.17	-0.14	-0.04
27	Availability	Prevent Fault	Exception Prevention	0	2	0	-2	-1	0	-0.17	-0.03	-0.07
41	Performance	Control Resource Demand	Bound Execution Times	-2	0	0	2	0	0	0.00	-0.10	-0.08
18	Availability	Preparation and Repair	Degradation	-2	2	1	0	-1	0	0.00	0.06	-0.09
48	Performance	Manage Resources	Schedule Resources	-1	-1	0	2	-2	-1	-0.50	-0.25	-0.13
36	Modifiability	Defer Binding	Defer Binding	0	-1	0	-1	2	1	0.17	-0.26	-0.23
2	Availability	Detect Faults	Monitor	-1	2	0	-1	-1	-1	-0.33	-0.12	-0.25
11	Availability	Preparation and Repair	Passive Redundancy	-1	2	0	-1	-1	-1	-0.33	-0.12	-0.25
21	Availability	Reintroduction Tactics	State Resynchronization	-1	2	0	-1	-1	0	-0.17	-0.08	-0.25
26	Availability	Prevent Fault	Predictive Model	-1	2	0	-1	-1	-1	-0.33	-0.12	-0.25
22	Availability	Reintroduction Tactics	Escalating Restart	-2	2	0	0	-1	-1	-0.33	-0.17	-0.34
10	Availability	Preparation and Repair	Active Redundancy	-1	2	0	-2	-1	-1	-0.50	-0.33	-0.36
28	Availability	Prevent Fault	Increase Competence Set	-1	2	-1	-1	-1	-1	-0.50	-0.31	-0.50
16	Availability	Preparation and Repair	Retry	-1	2	-1	-2	-1	0	-0.50	-0.48	-0.51
24	Availability	Prevent Fault	Removal from Service	-2	2	-1	0	-1	0	-0.33	-0.32	-0.51
20	Availability	Reintroduction Tactics	Shadow	-2	2	0	-1	-1	-1	-0.50	-0.38	-0.58
14	Availability	Preparation and Repair	Rollback	-2	2	0	-1	-2	0	-0.50	-0.41	-0.59
19	Availability	Preparation and Repair	Reconfiguration	-2	2	-1	-1	-1	-1	-0.67	-0.57	-0.83

Fig. 6. Worst ranked tactics according to MV values.

attributes. The new results are shown in Fig. 8 where the preferences of the same quality attributes have been changed to 0.19, 0.25, 0.18, 0.14, 0.13 and 0.11, respectively.

It is important to denote that these results of best tactics will change based on the given data of input parameters such as the candidate tactics, quality attributes

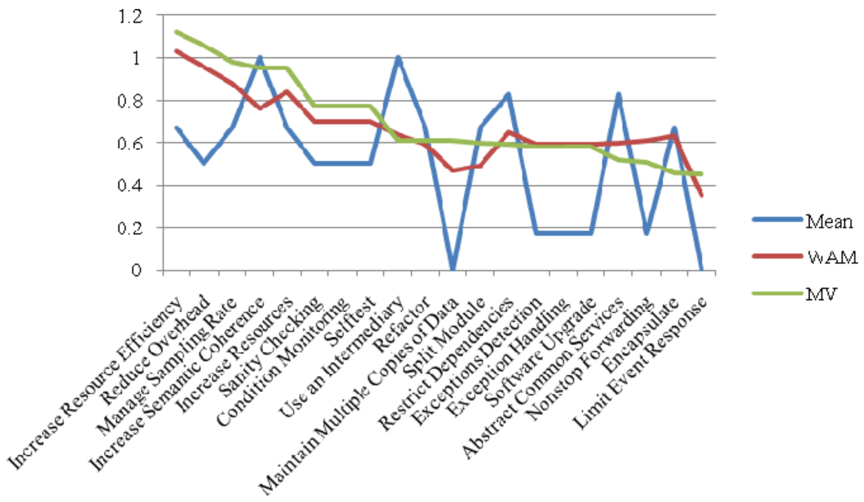


Fig. 7. Difference between Mean, WAM and MV values for 20 tactics based on specific inputs.

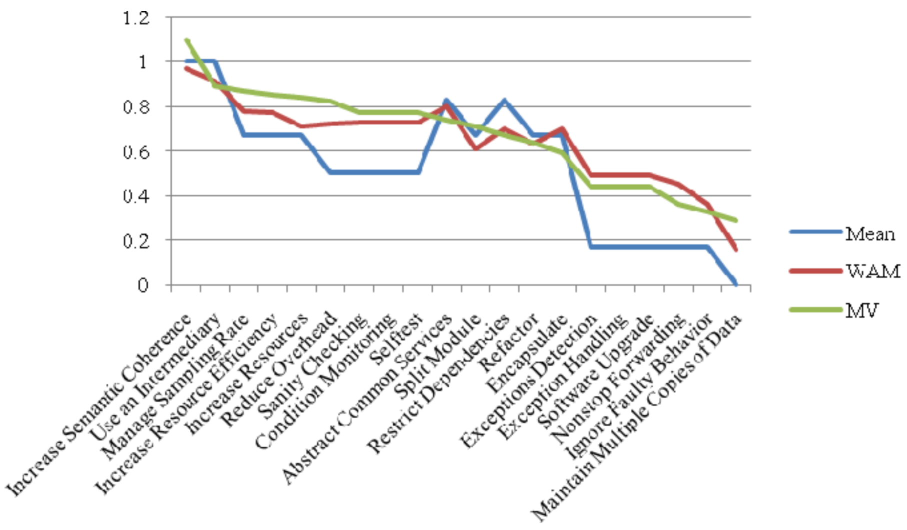


Fig. 8. Difference between Mean, WAM and MV values for 20 tactics based on specific inputs.

under focus in addition to their constraints such as the their preferences and interactions, in addition to the selected Choquet Integral method.

Additionally, some tactics that have low level of agreement between experts with respect to the impact of tactics on quality attributes (as defined in Sec. 4 of this paper) may affect the accuracy of the experimental results.

6.6. *How our proposed framework can be used in comparing different potential software architectures*

Our proposed framework has also the ability of comparing some different potential architectures dedicated for the software being developed. To explain this, suppose that software users of the aforementioned case study want to achieve functionality, reliability, usability and efficiency quality attributes. And also suppose that users in this situation cannot provide specific values for the preferences of these quality attributes but they need that functionality and usability have the same importance, and reliability and efficiency have the same importance as well. However, they need that reliability is more important than usability. Based on these users' quality expectations, the LP method can be selected to compute the Choquet Integral values for the dedicated tactics. Table 5 shows the LP values for some selected tactics related to the following subcategories: "Detect Faults" that is pertinent to availability, "Control Resource Demand" that is pertinent to performance, and "Increase Cohesion" and "Reduce Coupling" that are relevant to modifiability. Table 5 also shows three alternatives of architectures which are *Architecture* 1, *Architecture* 2 and *Architecture* 3 in addition to the tactics that actually realized in each one of them. By comparing the sum of LP values for each alternative, we can conclude that

Table 5. Comparison of three potential architectures based on their achievement of tactics.

Tactic	LP	Architecture 1	Architecture 2	Architecture 3
Detect Faults				
Ping/Echo	0		√	
Monitor	−1	√		
Heartbeat	−0.17			
Time stamp	−0.17			√
Sanity checking	0.67	√		√
Condition monitoring	0.67			
Voting	−0.17		√	
Exceptions detection	0.67	√		
Selftest	0.67			√
Control Resource Demand				
Manage sampling rate	0.64		√	
Limit event response	−0.09			√
Prioritize events	−0.09	√	√	
Reduce overhead	0.64			
Bound execution times	−0.81			√
Increase resource efficiency	1.28		√	√
Increase Cohesion				
Increase semantic coherence	−0.17	√		
Reduce Coupling				
Encapsulate	0			
Use an intermediary	0			
Restrict dependencies	0.28	√	√	
Refactor	0			
Abstract common services	0			
Sum of LP values		0.36	1.94	1.55

Architecture 2 is the best one for the proposed software because it has got the largest sum of LP values as shown in Table 5.

7. Conclusion and Future Work

In this paper, a framework for selecting the most appropriate tactics when developing transaction processing systems has been introduced. Such framework helps requirements engineers, software architects and software developers in selecting the best tactics based on some input parameters. These parameters include the impact of tactics on quality attributes, the required levels and priorities of quality attributes as determined by users and the interactions among these quality attributes. Three experts have been selected to judge the impact of tactics related to performance, availability and modifiability on the ISO 9126-based quality attributes. The level of agreement between experts has been statistically tested where the results show that most of tactics have high level of agreement. Furthermore, the proposed framework adopts fuzzy measures using Choquet Integral as a powerful MCDM quantitative evaluation method. An interactive software tool has been developed in order to clarify the capabilities and benefits of the proposed framework using a case study. An extension to this work can be achieved to approve the Choquet Integral results by domain experts. A broad work is needed for extending the impact results of tactics on quality attributes so that they can be used as important historical data in the evaluation process of software architecture such as the one adopted by the proposed framework of this paper. The impact of tactics on quality attributes in other application domains such as real-time systems can be also investigated in future work.

References

1. L. Bass, P. Clements and R. Kazman, *Software Architecture in Practice*, 3rd Ed. (Addison-Wesley, 2013).
2. I. Sommerville, *Software Engineering*, 9th Ed. (Addison-Wesley, 2011).
3. ISO/IEC 9126-1, Software engineering, Product quality, Part 1: Quality model, 2001.
4. D. Gade, The Evaluation of Software Quality, University of Nebraska-Lincoln, Industrial and Management Systems Engineering, Dissertations and Student Research, 2013.
5. R. Kazman and L. Bass, Toward deriving software architectures from quality attributes, No. CMU/SEI-94-TR-10, Carnegie-Mellon University, Pittsburgh, Software Engineering Institute, 1994.
6. N. B. Harrison and P. Avgeriou, How do architecture patterns and tactics interact? A model and annotation, *J. Syst. Softw.* **83**(10) (2010) 1735–1758.
7. F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns* (John Wiley & Sons, 1996).
8. K. C. Laudon and J. P. Laudon, *Management Information Systems: Managing The Digital Firm* (Pearson Education, 2006).
9. V. Torra and Y. Narukawa, *Modeling Decisions: Information Fusion and Aggregation Operators* (Springer-Verlag, 2007).
10. G. Beliakov, A. Pradera and T. Calvo, *Aggregation Functions: A Guide for Practitioners* (Springer-Verlag, 2007).
11. T. L. Saaty, Decision making with the analytic hierarchy process, *Int. J. Serv. Sci.* **1**(1) (2008) 83–98.
12. W. Wu, and T. Kelly, Safety tactics for software architecture design, *Proc. 28th Ann. Int. Computer Software and Applications Conf.*, 2004, pp. 368–375.
13. P. Trivedi, A. K. Dubey and S. Pachori, Reliability tactics, in *3rd Int. Conf. Electronics Computer Technology*, 2011, pp. 167–169.
14. M. Kassab, G. El-Boussaidi and H. Mili, A quantitative evaluation of the impact of architectural patterns on quality requirements, in *Software Engineering Research, Management and Applications* (Springer, 2012), pp. 173–184.
15. S. Tahmasebipour and S. M. Babamir, Ranking of common architectural styles based on availability, security and performance quality attributes, *J. Comput. Secur.* **1**(2) (2014) 83–93.
16. S. Al-Daajeh and M. Svahnberg, Balancing Dependability Quality Attributes Relationships for Increased Embedded Systems Dependability, Master Thesis, School of Engineering, Blekinge Institute of Technology, 2009.
17. J. McCall, Quality factors, *Encyclopaedia of Software Engineering* (John Wiley & Sons, 1994), pp. 958–969.
18. B. Boehm and I. Hoh, Identifying quality requirement conflicts, *IEEE Softw.* **13**(2) (1996) 25–36.
19. V. Pasrija, S. Kumar and P. R. Srivastava, Assessment of software quality: Choquet integral approach, *Procedia Technol.* **6** (2012) 153–162.

20. H. Zulzalil, A. A. A. Ghani, M. H. Selamat and R. Mahmod, Using fuzzy integral to evaluate the web-based applications, *Malaysian Software Engineering Interest Group*, 2011.
21. M. Grabisch, I. Kojadinovic and P. Meyer, A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the Kappalab R package, *Eur. J. Oper. Res.* **186**(2) (2008) 766–785.
22. M. Grabisch, k-Order additive discrete fuzzy measures and their representation, *Fuzzy Sets Syst.* **92**(2) (1997) 167–189.
23. N. B. Harrison and P. Avgeriou, Incorporating fault tolerance tactics in software architecture patterns, in *Proc. RISE/EFTS Joint Int. Workshop on Software Engineering for Resilient Systems*, 2008, pp. 9–18.
24. K. L. Gwet, *Handbook of Inter-rater Reliability: The Definitive Guide to Measuring the Extent of Agreement among Raters*, 4th edn. (Advanced Analytics, 2014).
25. M. Gamer, J. Lemon, I. Fellows and P. Singh, Package irr: Various Coefficients of Interrater Reliability and Agreement. R package version 0.84, 2012, Internet resource: https://cran.r-project.org/web/packages/irr/index.html.
26. J. L. Fleiss, B. Levin and M. C. Paik, *Statistical Methods for Rates and Proportions*, 3rd edn. (John Wiley & Sons, 2003).
27. M. Svahnberg and K. Henningsson, Consolidating different views of quality attribute relationships, in *ICSE Workshop on Software Quality*, 2009, pp. 46–50.
28. P. Berander, L. O. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson and P. Tomaszewski, Software quality attributes and trade-offs, Technical report, Blekinge Institute of Technology, 2005.
29. H. Zulzalil, A. A. A. Ghani, M. H. Selamat and R. Mahmod, A case study to identify quality attributes relationships for web-based applications, *Int. J. Comput. Sci. Netw. Secur.* **8**(11) (2008) 215–220.
30. K. Henningsson, Trade offs and Conflicts between Quality Attributes, Master's thesis, Blekinge Institute of Technology, 2001.
31. M. S. Deutsch and R. R. Willis, *Software Quality Engineering — A Total Technical and Management Approach* (Prentice Hall, 1988).
32. RStudio, Inc., Shiny Web Application Framework for R, 2015, http://shiny.rstudio.com/.
33. M. Grabisch, I. Kojadinovic and P. Meyer, Package kappalab: Non-Additive Measure and Integral Manipulation Functions, 2015, Internet resource https://cran.r-project.org/web/packages/kappalab/index.htm