Requirements Eng (2010) 15:215–233
DOI 10.1007/s00766-010-0099-3

RE'09 SPECIAL ISSUE

A controlled experiment to assess the impact of system architectures on new system requirements

Remo Ferrari · James A. Miller · Nazim H. Madhavji

Received: 19 October 2009 / Accepted: 1 February 2010 / Published online: 6 March 2010

Abstract While much research attention has been paid to transitioning from requirements to software architectures, relatively little attention has been paid to how new requirements are affected by an existing system architecture. Specifically, no scientific studies have been conducted on the "characteristic" differences between the newly elicited requirements gathered in the presence or absence of an existing software architecture. This paper describes an exploratory controlled study investigating such requirements characteristics. We identify a multitude of characteristics (e.g., end-user focus, technological focus, and importance) that were affected by the presence or absence of an SA, together with the extent of this effect. Furthermore, we identify the specific aspects of the architecture that had an impact on the characteristics. The study results have implications for RE process engineering, post-requirements analysis, requirements engineering tools, traceability management, and future empirical work in RE based on several emergent hypotheses resultant from this study.

Keywords Requirements engineering · Software architecture · Empirical study · Controlled study · Software engineering · Software process

A preliminary version of this paper was published in [29].

R. Ferrari (✉) · J. A. Miller · N. H. Madhavji
Department of Computer Science,
University of Western Ontario, London,
ON N6A 5B7, Canada
e-mail: rnferrar@uwo.ca

N. H. Madhavji
e-mail: madhavji@csd.uwo.ca

1 Introduction

While much research attention has been paid to transitioning from requirements to system architectures (SA)[1] [39], relatively little attention has been paid to how new requirements are affected by an existing SA. Indeed, it was over a decade ago, in a panel session [38], when several concerns and thoughts expressed the need to consider SA during requirements engineering (RE), for example: "We still do not have a clear understanding of the role of software architecture in requirements engineering" [38]; "Software architecture must be considered during requirements engineering to ensure that requirements are valid, complete, consistent, feasible, etc." [31]. Also, SWEBOK [17]—the software engineering body of knowledge—for example, does not describe any practices to deal with this issue.

Thus, to explore this matter further, given its thin baseline, we first conducted a survey [28] of 17 experienced RE and SA researchers and practitioners from North America and Europe. We found that the average rating of the importance of considering existing architecture when engineering new requirements was 4.5 (on a 5-point Likert scale)—implying that the respondents strongly agreed with this concept. Despite this, several respondents noted in the qualitative part of the survey that, in actual practice, many organizations neglect this consideration, or perform analysis only on existing high-level features (i.e., the requirements) of the current system, and not on the system's architecture.

Although there is curiosity in the RE community about the impact of SA on RE, and that there is a dichotomy

[1] For the rest of the paper, the acronym SA refers to System (or Software) Architecture as a software artefact.

between theory and practice, to the best of our knowledge, no scientific studies have ever been conducted to investigate this issue. Thus, we still do not truly know the "characteristics" of the newly elicited requirements in terms of how they are affected by the presence or absence of a SA in the RE process.

For example, first, a general question is: to what extent are new requirements affected by the existing SA? Also, the extent to which they are affected, what are the characteristics of this effect? For instance, to what degree are the requirements "user needs" focused, "technological needs" focused, or "architecturally focused"? etc. There are a number of such questions to which the RE research and practice community has no specific answers.

Having a grounded body of knowledge on these issues could benefit RE practice in a number of ways. For example, it could help in determining:

- when in the RE process one should examine the SA to ensure fitness of the new requirements;
- when in a product's lifecycle it is advantageous not to be influenced by the existing SA;
- the extent to which the system's requirements are misaligned with the business goals; and
- the requirements characteristics that should be tweaked in order to bring them back in line with the business goals.

Ultimately, such investigations are aimed at increasing the general quality and relevance of the system, improving RE processes, and at improving business efficiency and profitability.

Motivated by these issues, we conducted an exploratory, controlled, study to characterize the differences in the newly elicited requirements in the presence or absence of the SA. The study involved two types of groups. One type of group (the SA-group) received the SA of an existing (banking) system; whereas the second type of group (non-SA-group) did not receive the SA of this system. Both types of groups received the same initial requirements for this system, and they were both asked to enhance the system's requirements given the same problem description (or project goals).

This paper describes this empirical study and its findings in quantitative terms. For example, specific biases of various requirements characteristics in the presence/absence of the existing SA are given and interpreted. These findings constitute new knowledge and are the chief contribution of the paper. The paper also describes the implications of the findings for both RE practice (e.g., RE process engineering, post-requirements analysis, traceability management, etc.) and RE research (e.g., seven emergent hypotheses, and RE tools).

This paper is a significantly enhanced version of [29]. The additions to this paper include:

- The investigation of a new research question regarding specific aspects of the SA that affected the requirements.
- A significantly expanded related work section.
- More information given on the data analysis conducted.
- Extended implication section including new hypotheses for further empirical studies.
- Elaboration of empirical study procedures employed.
- Appendix describing data collection instrument used.

The rest of the paper is structured as follows: Sect. 2 describes related work; Sect. 3 describes the empirical study; Sect. 4 analyzes the data, presents the results, and makes interpretations; Section 5 discusses the implications of the findings; and Section 6 concludes the paper.

2 Related work

In this section on related work, we focus on two key aspects: (1) observations, commentary, and empirical work on the role of SA in RE and (2) recent technological-based research on requirements evolution. Subsection 2.3 concludes with a reflection on the current state of research described in Subsects. 2.1 and 2.2. Other aspects that are related (for example, technology to transition from RE to SA, or empirical studies focused on requirements-oriented issues while architecting) are omitted here because they are not as relevant as the two aspects identified earlier. The reader can refer to [13] for a thorough discussion of related works focused on the transition from RE to SA.

2.1 Role of SA in RE

As early as 1994, a panel session at a RE conference was held to deliberate on the role of SA in RE [38]. This marks perhaps the first attempt by the RE research community to recognize this relationship.[2] The consensus in this panel session was that this relationship is an important one but was little understood.

In this same session, Jackson [19] gave four key reasons as to why RE and architecting are best treated as interweaving processes. First, RE can be "very tricky" in that, often, it can be simpler to start by building the system right away, even if only in outline. Second, evaluating possible system designs early can help gain an important understanding of which requirements might not be feasible, saving time, and money. Third, requirements can sometimes be reasonably embedded in system design, eliminating the need for formal specification during RE. Finally,

[2] Related workshops, such as STRAW 2001 and 2003 [39] focused mainly on transitioning from RE to SA and not on the role of SA in RE.

there is evidence that successful developers are those who are able to move relatively more freely between stages (i.e., RE, architecting, design, testing, etc.) within the development cycle.

Shortly thereafter, in 1995 [11], El-Emam and Madhavji found four factors for RE success in information systems that deal with architecture or the system (the first being relevant for this study): the *adequacy of diagnosis* of the existing system (which includes SA); the *fit between the architecture and the way users work*; the *fit between the recommended requirements solution and the strategic orientation of the organization*; and the *fit between the recommended solution and the technical orientation of the organization*.

Subsequently, hints can be found in the pedagogical literature [24] promoting the need to consider the existing system in the RE process. More recently, in 2000, Nuseibeh and Easterbrook [31] stated that we needed a "better understanding of the impact of software architectural choices on the prioritization and evolution of requirements." In [30], Nuseibeh describes the "twin-peaks" model, which captures the iterative relationship between RE and architecting. An important aspect of this model is that the architecting process can and should feed back into the RE process (as well as vice versa).

In a recent study [28], Miller, Ferrari, and Madhavji investigated the different types of effects an SA has on requirements decisions. They identify and quantify four principal ways in which a previous architecture can affect evolving requirements work, i.e., as an enabler (30%), as a constraint (25%), as an influence (6%), or the null case (39%). This means that approximately 60% of the decisions were affected by the architecture, highlighting the impact an existing architecture has on RE.

While these are some of the key works highlighting the role of SA in RE, the body of knowledge on this topic is fairly thin overall and has basically remained static.

2.2 Requirements evolution

An area of research that is related to our work is requirements evolution, in particular from the viewpoint of methods, notations, and tools development. In the following subsection, we highlight recent research in this area from prominent RE literature sources. Because our study is focused on both the *absence* and *presence* of SA in RE, we include research that does not, explicitly or implicitly, consider the existing SA in requirements evolution.

In [41], the authors present a method for requirements engineers and project managers to perform software evolution in the domain of embedded systems. The method's primary purpose is to aid in systematic reasoning on the identification of volatile requirements and planning changes to the architecture. The method is composed of four phases. The first phase is preparation for volatility analysis and is meant to establish the timeframe restricting the current volatility analysis and identifying the types of components that will be involved in the changes. The second phase is environmental change anticipation where the primary tasks are to identify and characterize changes that may occur in the system's environment within the identified timeframe. Specifically, the analyst needs to identify actors, roles, external events, and environmental facts that could cause changes. The third phase is the actual change impact analysis, which is composed of identifying the adaptation needs, such as identifying features to be affected and estimating their business impact. The result of this phase is a prioritized list of adaption needs that should be included for implementation. The final phase is the product evolution planning, where the analysts establish when and how the previously high-priority adaptation needs are to be introduced into the system. The result of the method is a plan for product evolution based on the high-priority adaptation needs.

In [12], the authors present a framework that defines challenges for RE caused by co-evolution and also show which and how existing requirements technologies address the identified challenges. Their framework is structured around five dimensions which each correspond to a RE-related issue regarding co-evolution. These dimensions were determined from their experience in three industrial evolution projects. To summarize, the five dimensions are: (1) understanding the consistency relationship between RE-related artifacts and other co-evolving artifacts from outside RE (e.g., design, testing, code,), (2) formalizing notations to express evolution requirements, (3) elicitation of evolution requirements, (4) propagating identified changes to processes outside RE, and (5) verifying the relationships between the proposed changed system entities. The authors conclude that no particular existing technology addresses all of the above-mentioned dimensions of co-evolution and therefore a research gap exists in this area.

In [20], the authors propose the use of a domain analysis approach to identify and document current and future requirements in an application domain. The author's primary motivation for this approach is that defining a long-term strategy for software product evolution is an extremely difficult task because the requirements and future trends must be anticipated in advance. They argue that a domain analysis technique can be used for this anticipation of future requirements, while recognizing the problem that ongoing domain analysis for evolutionary purposes can be costly (in terms of time and cost). Thus, they propose to use an instantiation of the PuLSE-CDA (customized domain analysis) method [4], which aims to overcome this problem by systematically coordinating domain analysis effort with

necessary product evolution activities. The primary goal to facilitate a cost-effective approach is to only model a sub-domain where only the key future changes are modeled, in order to reduce excess modeling of irrelevant information. In short, the main steps are to : (1) analyze existing change requests from maintenance and marketing, in addition to analyzing existing application domain knowledge, which provides an initial list of sub-domain candidates, (2) map the identified candidates to logical software components, (3) model and refine each logical component's relevant data attributes and processes in which the components are involved. The output of this method is a map of inter-related domain models (and logical components) that are a subset of the overall application domain. Each component can then be implemented and integrated with the existing system.

In [6], the authors investigate requirements evolution from the perspective of scenarios. Specifically, they derive a scenario evolution taxonomy from the investigation of twelve case studies spanning over 200 scenarios; each of these studies comprised the analysis of a software project during its evolutionary phase. The authors state, based on the findings from the case studies, that the main challenges in scenario evolution are in understanding and managing the relationships between scenarios; an individual scenario can often be related to many other scenarios and in the projects examined in the case study, there was minimal technological support for this problem. The resultant scenario evolution taxonomy then describes the classification and formal heuristics for semi-automated detection of scenario relationships, as well as an initial suggestion for a formal notation that can be used for scenario relationship representations.

In [14], the authors propose a simulation model to help project managers and requirements analysts understand how requirements volatility impacts a given software development project. The model is built on results from an empirical survey administered to software project managers and developers, where more than 50 parameters (such as number of requirements change requests per release, requirements defects detection rate during design, percentage of perceived job size increase due to requirements change) were derived from the survey data. Based on this theoretical model, the authors designed a software simulator that can be used by developers to input project parameters that are related to requirements volatility and determine the potential impact for a given project of changing requirements. The simulator was used on two industrial projects to explore the relationship between requirements volatility and its impact on software projects. The results of these case studies show that there were significant simulated cost, schedule, and quality impacts due to requirements volatility.

In [36], the authors tackle the problem of requirements evolution with a formal requirements specification modeling approach and tool. Currently, this approach works on specifications modeled using i* [43]. The aim of the approach is the precise definition of the change requirements, and the approach does this by facilitating the modeling of specific gaps between the current requirements specification and the target specification. The approach uses a generic gap typology where each gap is associated to a predefined type of requirements change (such as add actor and remove feature), and these are then associated with gap operators which perform the actual transformations in the i* model. The approach and tool was validated, and the authors estimate that approximately 50% time was saved eliciting change requirements using this approach vs. a manual approach.

In [7], the author's raise the problem that in large software projects, the number of changes and enhancements requested for inclusion in the next release often exceeds the resources available to implement those changes. Therefore, technological support is needed to incorporate the multitude of factors (such as approval for finance for the change, development time, and human expertise required) that influence these possible changes into an improved set of information for the purpose of facilitating better decision making. The authors propose the use of influence diagrams, which are an extension to Bayesian nets [9], to formalize the combining of the different change factors to address the requirements change problem. Influence diagrams, as argued by the authors, are suitable because they model both decision-making trees along with random chance events. The combination of these two dimensions adequately covers both fixed and volatile project factors that can influence decisions to implement certain requirements changes.

2.3 Reflection on research

Having discussed the current knowledge pertaining to the role of an SA in RE and requirements evolution, in this subsection, we reflect on the current state of research in these areas. As discussed In Sect. 2.1, as early as 1994, researchers discussed the importance of the role of an SA in RE [38]. A few other works have commented on this issue since then [11, 28, 31]. However, beyond these works there has been, to the best of our knowledge, little research conducted in the area of the role of an SA in RE. Despite the sparse research in this specific issue, there is a wide range of technological-based research conducted in the area of requirements evolution, as described in Sect. 2.2, which are meant to improve the RE process in the context of an evolving system. However, the work is often focused solely on the RE process; downstream activities such as

architecting, coding, testing are treated as black-box processes, and there is thus a lack of explicit recognition of the interaction of RE with SA as highlighted as being important, for example, in Nuseibeh's Twin-Peaks Model [30]. Furthermore, there is a lack of empirical evidence regarding the different requirements characteristics, and how these characteristics are impacted by the presence or absence of an SA during systems evolution. The empirical study presented in this paper is meant to present detailed quantitative findings on the impact of an SA on requirements characteristics, which can then be fed back into technological research as described in Sect. 2.2.

Though the importance of conducting empirical studies in software engineering (SE) has been recognized [40, 42], Shaw's analysis [37] of research papers presented at a prominent 2002 SE conference suggests that only 12% were submitted in the category of "Design, evaluation or analysis of a particular instance" and 0% in the category of "Feasibility study or exploration". In [13], we presented our own analysis of published papers. In the fields of RE and SA, since the year 2000, only approximately 15% of the published papers were in the above-mentioned categories, suggesting that studies such as the work described in this paper are currently rather rare. Our work is meant to help in filling this research gap.

3 The study

We now describe the core parts of the study. Section 3.1 describes the research paradigm used, GQM [2], to state the goal, questions, and metrics for this study. Section 3.2 describes the study design. Section 3.3 describes the study hypothesis. Section 3.4 describes the participants. Section 3.5 describes the RE project. Section 3.6 describes the data collection procedures. Finally, Sect. 3.7 describes threats to the study.

3.1 Goal, research questions, and metrics

This study followed the Goal-Question-Metric (GQM) paradigm, which helps in ensuring that measurements taken in the study are aimed at answering specific research questions which, in turn, help in achieving the overall goal of the study [2].

The overall goal of this investigation was:

To better understand the characteristics of requirements elicited in the absence or presence of a SA.

In order to obtain a quantitative insight into the elicited requirements, we decomposed the notion of a requirement into specific, measurable characteristics. Table 1 defines these characteristics, which are rooted in three sources: those which are prominent in the literature (such as requirement type); those which would intuitively be of interest to industry (such as cost); and those which relate to an architecture (such as architectural relevance [3]). Five researchers subsequently validated these characteristics, and their associated metrics, as an acceptable set of variables for the study.

Linking the overall goal and the characteristics described in Table 1 is the following question aimed at achieving the goal:

Q1 Which requirements characteristics were affected, and to what extent, by the presence or absence of the SA?

Our objective now is to determine whether or not the SA (an *independent* variable) has an impact on the requirements characteristics (the *dependent* variables). We can accomplish this by comparing the requirements sets elicited by two different types of study groups, one group which is comprised of teams that have access to the SA while doing RE (the SA-groups) and the other set of teams do not have access to the SA (the non-SA-groups). To complement and probe deeper into the findings from Q1, we raise the following research question.

Q2 Which specific aspects of the SA affected the requirements?

In this question, we are examining the specific aspects of the architecture and how they affected the requirements characteristics that exhibited significant differences (i.e., findings from Q1). To investigate this question, the requirements analysts teams, during the RE process, had to explicitly state when an architectural aspect affected their RE work. We then take the intersection between the identified affected architectural aspects and the requirements characteristics with significant differences from Q1.

3.2 Hypothesis

Because this study was undertaken without a significant underlying theory (on how requirements characteristics are affected by software architectures) on which to build a priori hypotheses, this study is best described as an exploratory study [35]. Contrary to a dogmatic viewpoint, hypothesis testing *can* be done in exploratory studies but is not meant to confirm existing scientific theory as in a purely experimental design. Rather, the results of the hypothesis testing here provide initial indications on which future experimental research can be conducted [35]. As a result, the only hypothesis that will be discussed in this paper are the "null hypothesis" which is necessary for null hypothesis statistical testing (NHST). The null hypothesis for a given metric M, where M is a requirements characteristic described in Table 1, can be generally stated as:

Table 1 Requirement characteristics

1	Focus on cost	The degree to which the cost factor, concerning the system's content, is prominent in the requirement
2	Focus on time	The degree to which the development time factor, concerning the system's content, is prominent in the requirement
3	Focus on quality	The degree to which the quality factor, concerning the system's content, is prominent in the requirement
	Note: a requirement can be prominent in one or more of the above-mentioned three characteristics	
4	Focus on user's needs	The degree to which the requirement is focused on the needs of the end-users. End-user issues include different ways of accessing the system, end-user features provided, usability requirements, etc.
5	Focus on client's needs	The degree to which the requirement is focused on the needs of the client (i.e., the needs of the organization itself.) Note the difference from (4) earlier
6	Focus on technological needs	The degree to which the requirement is focused on technological needs. Technological issues include the "back-end" server, choice of algorithms and data types, interface specifications, communication protocols, data access mechanisms, etc. that are important in terms of ensuring that the system will be technically sound
7	Testability	The degree to which it can be shown that a requirement can be tested against
8	Implementability -	The level of effort required to implement a requirement
9	Importance -	The degree to which the success of the system depends on the implementation of a requirement
10	Architectural relevance -	The degree to which the requirement will have an impact on the architecture, e.g., architectural-driver. Note: not all "important" requirements are architecturally relevant, e.g., a common but basic requirement
Scale: Scale for characteristics (1–10): These were measured using a 7-point Likert scale. The scale was the same for all characteristics and is defined as follows: 7—Very high, 6—High, 5—Moderately high, 4—Neither high nor low, 3—Moderately low, 2—Low, 1—Very low.		
11	Level of abstraction	The breadth of impact of a requirement. Does the requirement affect a module, a component, an entire sub-system, etc.
Scale for 11: This was measured with a 7-point ordinal scale; this scale was used because there is an ordering in the levels of abstraction (high to low for example), but this ordering does not denote specific, discrete values, with equal intervals between them [32]		
12	Type of requirement	This categorizes the requirement into functional, non-functional, and business quality, where there can be more than one category for non-functional and business qualities. Examples of categories in this scale include functional, standards, legal, performance, and safety

H_0 the presence of a SA has no impact on M.

If NHST leads to the rejection of this hypothesis for any metric M then we can say that "characteristic M is affected by the SA".

3.3 Study design

This is a *post-test* only control group design experiment [22]. This type of design involves administering a treatment (i.e., SA documentation) to one type of groups (in this case, the SA-groups), with observations taken only at the end following the treatment. These observations are contrasted against the non-SA-groups that did not receive the treatment. It is from this contrast that the results of this study are drawn. A visual depiction of this design is given later; where the O represents observation, X represents treatment, and R represents random assignment. This study design was used because it falls in the category of *strong designs* [22] and alleviates many internal validity threats in a multiple group design.

SA-Groups R -------X ---------- O

Non-SA Groups R ----------O

The specific type of experimental design that our study falls under is a nested ANOVA design [35]. This design is used when there is one measurement variable (i.e., requirements characteristic) and two or more nominal variables (i.e., categorical variables). In our study, there are two nominal variables: (1) the type of study group (SA-groups vs. non-SA-groups) and (2) the different requirements teams. These nominal variables are nested, meaning that each requirements team belongs to only one category of the higher-level nominal variable (i.e., the SA-groups vs. non-SA-groups). We used this design

because although our *unit of observation* is the requirements teams, the *unit of analysis* was the individual requirements since we were interested in the differences in the requirements characteristics and not differences in teams. The subsequent analysis (see Sect. 4.1) supports this study design and reconciles the difference between *the unit of analysis* and *unit of observation*.

3.4 Participants

We used convenience sampling [22] to involve 25 final-year RE students in the study. In order to conduct the study involving students, we received consent from the ethics board at the University of Western Ontario. The threat of using students as participants is discussed in the external threats to validity section (Sect. 3.6.2). The participants were randomly assigned to groups of two with one group of 3, making a total of 12 groups. The groups were then randomly divided into two types: the so-called architecture (SA) groups and non-architecture (non-SA) groups.

To ensure that the participants had sufficient knowledge to conduct the project, they were given theory knowledge in RE and two pre-requisite requirements projects prior to the project to learn and familiarize themselves with RE practices such as elicitation, analysis, negotiation, validation, and prioritization. The assessment of the pre-requisite projects and RE knowledge indicated a satisfactory level of attainment to conduct the investigative project. Subsequent to this, SA learning sessions were given to the SA groups so that they could perform architectural analysis required for the project. These sessions focused on understanding architectural documentation and the ADD/ATAM methods for architecture design and assessment [3].

3.5 The RE project

Each of the 12 groups was given the same set of requirements elicitation tasks for upgrading a software infrastructure for a fictitious bank:

1. To add support for *Interac* service to the existing system.
2. To create a new wireless banking application that would allow customers to carry out basic banking transactions through their cell phones or PDAs.
3. To reduce the operational cost of the telephone banking system.
4. To increase modifiability in the web banking system.

These tasks were chosen since they constituted a sizeable and complex RE project that would still be feasible within the constraints of a University course. We held numerous peer-review sessions with a total of six experts to validate these four tasks with respect to their appropriateness in giving a project that met both pedagogical and study needs.

Both types of groups, SA and non-SA, were given the requirements for the existing system. These pre-existing requirements were baselined from a previous project [13]. The requirements elicitation process and techniques followed are described in [24].

Also baselined was the pre-existing architectural document (developed using the ADD method [3]) from the same previous project [13], given to each of the SA-groups only. This document included architectural information such as numerous different tactics, quality attribute scenarios, decomposition views, user/layer views, class views, component and connector views, deployment views, work assignment views, sequence diagrams and state charts. The RE project, including logging of the elicited requirements, took 2 months to complete.

3.6 Data collection

The data collected for analysis were the ratings for the requirements characteristics (defined in Table 1) for the elicited requirements (to answer Q1—see Sect. 3.1) and the list of architectural aspects that affected the SA-group during the RE process (to answer Q2—see Sect. 3.1). In the following two subsections, we discuss these two disparate sources of data.

3.6.1 Requirements characteristics ratings

The primary source of data is the requirements ratings for the requirements characteristics (defined in Table 1), where an instrument was designed for this purpose (see "Appendix"). Three external researchers rated the requirements using this instrument. This process involved examining the title, description, and rationale for each requirement and giving a rating on the appropriate scale of each and every characteristic.

Since the data was subjective and different raters could measure constructs in different ways, an inter-rater agreement method from [15] was used to establish rating reliability. Basically, the ratings for each characteristic for each requirement were assigned to two researchers. The two researchers independently performed the ratings. Following this, if necessary, the researchers harmonized their ratings to finalize a rating. In particular, if there was a difference of more than one in the ratings, a third researcher also performed the rating to reconcile the difference. All ratings were conducted blindly, i.e., without knowledge of which group authored which requirements. The researchers used for the rating procedure all had 3–10 years experience in RE, and the minimum academic level was a Ph.D. candidate. Prior to the ratings collected

Table 2 Sample rating of requirements

Requirement id	Focus on cost	Focus on time	Focus on quality	Focus on user's needs	Focus on client's needs	Focus on tech needs	Testability	Implementability	Importance	Arch relevance
R1	4, 4, ***4***	4, 4, ***4***	4, 4, ***4***	2, 1, ***1.5***	1, 1, ***1***	7, 7, ***7***	7, 6, ***6.5***	7, 7, ***7***	5, 4, ***4.5***	2, 3, ***2.5***
R2	4, 3, ***3.5***	4, 3, ***3.5***	4, 4, ***4***	7, 6, ***6.5***	1, 2, ***1.5***	1, 1, ***1***	7, 7, ***7***	4, 5, ***4.5***	5, 4, ***4.5***	4, 4, ***4***

for this study, a pilot rating session was conducted on a subset of the requirements with all researchers involved, the purpose of which was to train the raters on the rating procedure. Table 2 shows the ratings of two example requirements, R1 and R2.

R1 When performing an Interac transaction, if the primary server is busy, it will send a 'Server busy' response signal followed by a secondary server IP. The client machine should then have to re-establish connection with the secondary server and perform a second request. This results in less demand on the primary server.

R2 The customer shall be able to add or remove companies from their profile to which bills are being paid using their mobile application (cell phones, PDA's, etc.).

The first two numbers in each column represent ratings given to the requirement for the given characteristic by two independent raters (see Table 1 for explanation of scale). The third value in bold italic is the final, agreed upon value. One example interpretation of this instance of ratings is that R1 is high on technological focus (7 out of 7) but low on user-focus (1.5 out of 7) and R2 is high on user-focus (6.5 out of 7) but low on technological focus (1 out of 7). Likewise, interpretations can be made about other characteristics and their ratings. Because the scale for the characteristics Abstraction and Requirement Type are different, we omit them in the example for simplicity.

3.6.2 Architectural aspects

We used a tool from [28] that had the dual purpose of supporting the subjects' project and of recording relevant data for this study. For example, the tool maintained a pervasive list of both the original requirements (as well as their evolution) and new requirements introduced by each team. Also, the tool recorded data from the SA-groups (only) about the specific parts of the architecture that had an impact on the requirements. To help ensure the quality of the data gathered in this tool, regular meetings with each project team were held to clear up issues and to monitor the progress.

3.7 Threats to validity

We classify threats into those *internal* and those *external* to the project, as well as *construct* and *conclusion* validity. We focus here only on those considered relevant to our study (see Table 3). Description of other types of threats can be found in [22].

3.7.1 Internal validity

Internal validity deals with whether we can infer that a relationship between two variables is causal, and not due to any confounding factors [22]. There are numerous specific types of internal validity threats [22], we discuss here only the threats that applied to our study and the procedures we employed to contain the threat.

Differential selection: This is when possible characteristics of the subjects may, by chance, differ between the two types of groups and possibly affect the quality of the data. In our study, such a characteristic is the participants' SE educational and industrial-experience backgrounds; participants with differing SE background could possibly perform differently in the project. To identify any such

Table 3 Results of nested ANOVA testing

Requirement characteristic	Independent factor	Degrees of freedom	*F*-value	Significance
Focus on user needs	SA	2	267.038	0.000
	Team	10	2.914	0.001
Focus on technology needs	SA	2	335.914	0.000
	Team	10	2.688	0.003
Architectural relevance	SA	2	2148.621	0.000
	Team	10	1.233	0.266
Importance	SA	2	3244.578	0.000
	Team	10	1.830	0.052

possible outlier participants, prior to the study, each participant was interviewed about their background experience so that any subjects with prior SE industry experience could be identified. None had any such experience. This, coupled with the knowledge that every participant was a full-time computer science student and had taken similar software engineering courses for specialization, ensures that they had undergone similar SE training. Furthermore, weekly review sessions were used to identify any obvious outliers during the project, which we did not find.

Differential mortality: This occurs when a physical or mental change occurs to participants during study that is not "equal" between the two types of study groups. This threat existed in our study because of the duration of the participants' project (see Sect. 3.5), which lasted approximately 2 months. To contain this threat, the researchers reviewed and assessed weekly submissions of work and collected data. Additionally, weekly motivation meetings were held to further monitor the participants' progress. At the conclusion of the study, all initial participants remained in the study and no effects of the differential mortality threat were observed.

Researcher bias: This occurs when the researcher, knowingly or unknowingly, influences the outcome of the study. This threat exists in our study because of the subjective nature of the requirements characteristics ratings (see Sect. 3.6.1). To mitigate this threat, multiple researchers and domain experts, and an "open" process (with no direct investment in study), were used in the study processes. These are recognized techniques for dealing with *researcher bias* [22].

3.7.2 External validity

External validity refers to the degree to which the results of a study can be generalized across a population, time or place [22]. Population validity can exist when generalizing to industry; the reason for using students in our study was the *availability sampling* technique. It would have been extremely difficult (if not impossible) to conduct this first-time controlled study in industry. The use of students should not diminish the results of this study, as important results have been found in other SE studies when student-based studies have been conducted (e.g., in requirements triage [5]; code inspection [8]; and lead-time impact assessment [18]). We do acknowledge the threat in generalizing to experienced requirements engineers; however, there is no evidence suggesting that the results could not be generalizable to, at the very least, novice requirements engineers in industry [18]. Regardless, *exploratory* studies such as this are an important first step toward determining initial results on a particular research issue that can provide the groundwork for future studies in wider contexts.

3.7.3 Construct validity

Construct validity refers to the extent to which a measurement corresponds to theoretical concepts (i.e., constructs) concerning the phenomenon under study [22]. In this study, the constructs were the requirements characteristics. These were measured by an instrument created and used by external researchers (see Sect. 3.6.1). We held numerous peer-review sessions with a total of six experts to validate the measurement instrument with respect to the theoretical constructs we wanted to investigate (see Sect. 3.1).

3.7.4 Conclusion validity

Conclusion validity is the degree to which conclusions we make based on our findings are reasonable [22]. There are three ways in which conclusion validity can be improved in a quantitative-based study: statistical power, reliability, and proper implementation of study methods. In our study, statistical power (or lack thereof) is not an issue, as our statistical tests are performed on ratings from approximately 900 requirements which were elicited by the 12 RE teams. Also, the study design and statistical tests (see Sects. 3.3, 4.1, respectively) accounted for the difference between the *unit of analysis* (RE teams) and *unit of observation* (requirements). As discussed in Sect. 3.6.1, we used multiple researchers to rate each requirement in order to achieve a reliability of the rankings. Lastly, the researchers performing the ratings were trained prior to the actual rankings to ensure they properly carried out their task.

4 Data analysis, results and interpretations

We now describe the analysis of the data gathered, the findings, and their interpretation. The implications of the findings are described in Sect. 5.

4.1 Data analysis

The SA-groups collectively produced 443 newly elicited requirements; whereas the non-SA-groups collectively produced 458 newly elicited requirements. Note that, normally, in a controlled experiment, the analysis of data would be conducted on the randomized construct (in our case, the teams). However, we are primarily interested in exploring whether there are significant differences in the "characteristics" of the requirements elicited by the teams (SA vs. non-SA) and not simply in the teams themselves. Thus, the analysis we have conducted acknowledges the team randomization and "takes into account the extent to which outcomes (i.e., characteristic ratings of the

requirements) differ across all the teams "independent of the treatment effect" (i.e., ignoring the presence of SA for the treatment group. This is an established procedure [35]". Specifically, we conducted separate statistical analysis that incorporated the possible effect of the different teams on the characteristic ratings. That is, we performed a two-way nested ANOVA[3] with the presence of SA as a *fixed* variable and the teams as a *random* variable. This statistical analysis corresponds to the experimental design used for this study (see Sect. 3.3) and is used to test the hypotheses (see Sect. 3.2). The results of the two-way nested ANOVA are presented and discussed in the next section. We qualitatively analyze identified architectural aspects that affected the SA-groups' RE process and link them as sources for the differences reported in the preceding analysis.

4.2 Results and interpretations

We now describe the results and interpretations from the data analysis that was performed. In the following subsection, we discuss the results from the hypothesis testing (answer for Q1—"which requirements characteristics were affected"—see Sect. 3.1). We then discuss more detailed results regarding each requirement characteristic. Lastly, we present the findings from Q2 (see Sect. 3.1) where we qualitatively link the SA aspects that were determined to have affected the RE process with the requirements characteristics that showed statistical significance in the preceding analysis.

4.2.1 Requirements characteristics hypothesis testing

Recall that in Sect. 4.1, we mentioned the two-way nested ANOVA which tests for the effect, on the requirements characteristics, of both the SA and being in a different requirements team. Here, we now present and interpret the results from this two-way nested ANOVA. From Table 3, we see the characteristics that showed statistically significant difference due to the presence/absence of an SA when controlling for the "team effect" (*focus on user needs*, *focus on technology needs*, *architectural relevance* and *importance—all significant at* $p = 0.000$). This means that there is virtually no possibility that these results were due to chance. The Table also shows that there was a statistically significant effect from the teams for the characteristics *focus on user needs* and *focus on technology needs* ($p = 0.001$ and 0.003, respectively). The characteristics *architectural relevance* and *importance* did not show a statistical significant difference for the *team* variable. The characteristics in this Table were the only characteristics that showed a significant difference for either the SA or team variable.[4]

What this means is that both the presence/absence of the SA and being on a different team had an effect on the user and technology focus characteristics. Because this study was more technically oriented, we did not collect more specific data on why being in a different requirements teams led to an effect on these particular requirements characteristics. However, there are some intuitive explanations for why this interesting phenomenon occurred. It is possible that different individual personal interests and capabilities played a role in changing the "flavor" of the requirements. For example, if the teams had individuals who preferred downstream processes and were thus more "solution" driven (e.g., designing, testing, coding.), then these teams' requirements would have a more technological bias regardless of the presence/absence of an SA. Conversely, if the teams had individuals who preferred the more human aspect of SE (e.g., RE, human–computer interaction.), then their requirements would necessarily be more user-focused. The role of human personality and capability is outside the scope of this study; however, it could prove to be an interesting avenue for future empirical research and is the focus of one of our emerging hypotheses in Sect. 5.5.

Irrespective of the team effect exhibited above for the two requirements characteristics, we have demonstrated that the SA does have a significant effect on the requirements characteristics and so we discuss the results for the rest of this section at the requirements level which is the focus of this paper.

4.2.2 Detailed requirements characteristics results

We now discuss in more detail the results and their interpretations pertaining to each of the requirements characteristics (described in Table 1). Recall, from Table 1 in Sect. 3.1, that 10 of the 12 characteristics (e.g., *focus on cost*, *focus on time*, *focus on quality*.) were measurable on the Likert scale; whereas the remaining 2 (*level of abstraction* and *type of requirement*) were measurable on the ordinal and nominal scales.

4.2.2.1 Technological needs versus user needs On average, the SA-groups scored higher for *focus on technological needs* (4.12 vs. 3.42, Table 4); whereas the non-SA-groups scored higher for *focus on user needs* (3.65 vs. 3.26). There

[3] The 2-way nested ANOVA testing was done using SPSS 16.0 from SPSS Inc. (http://www.spss.com).

[4] The ANOVA test can only be conducted on the Likert-based requirements characteristics and not the ordinal-based characteristics (*abstraction* and *type of requirement*). These ordinal-based characteristics are analyzed separately in Sect. 4.2.2.5.

Table 4 Mean scores of the two types of groups for selected qualities

	Group	Focus on user needs	Focus on tech. needs	Arch. relevance	Imp.
Mean	SA	3.26	4.12	4.59	5.63
	Non-SA	3.65	3.42	4.12	5.28
Cohen's effect size		Large	Large	Large	Large

The Cohen's effect size indicates the difference between the two types of groups is "large" [35], meaning that there is not only a statistical difference between the two groups, but that the difference is substantial for real-world application of the results (e.g., making a business decision). What this means in terms of the real-world RE processes and products is discussed in the next subsection

was thus a trade-off between the characteristics of the requirements from the two types of groups. Usually, when focus on technological needs was high (scoring a 5, 6 or 7 on the Likert scale), focus on user needs was low (scoring a 1, 2 or 3) and vice versa. The notion of this trade-off is supported by a follow-up test that was conducted, Spearman's rho test, which showed a statistically significant ($p = 0.007$) inverse correlation between the two characteristics. From the perspective of RE processes, this data suggests that the SA-groups had *technological needs* higher than *user needs* in 52.9% of their requirements compared to 41.6% for the non-SA-groups. Likewise, the non-SA-groups had *user needs* higher than *technological needs* in 46% of their requirements compared to 37% for the SA-groups. Table 5 characterizes the bias toward technological and user needs for each type of group.

The surface-level reasoning for this difference could be that the SA-groups oriented themselves toward the technological arrangement of the system; whereas the non-SA-group oriented themselves toward the user perspective of the system. However, the fact that the SA-group, either knowingly or unknowingly, "shortchanged" the user-oriented requirements is rather surprising.

What this means is that the potential benefactors of these requirements (i.e., the various stakeholders) are not having their needs fully met, which could then lead to negative feedback later in downstream processes or when the given product is released, resulting in lower customer satisfaction, poorer product quality, development rework, etc.

4.2.2.2 Architectural relevance Another characteristic where the two types of groups scored differently is *architectural relevance* (see Table 4), (i.e., the degree to which a requirement will affect the architecture [3]). The mean ratings were 4.59: SA-groups vs. 4.12: non-SA-groups. Examining the data more closely, the SA-groups had more requirements (56) [13%] that scored 7—extremely high than the non-SA-groups (23)[5%]. Conversely, the non-SA-groups had more requirements that scored 1–3 (extremely low to slightly low) (163) [37%] than the SA-groups (161) [27%].

The reason for this variance could be that having access to the system's architecture, the SA-groups are better poised to question the architectural relevance of an elicited requirement in their decision making.

Ultimately, they seem to be have selected more architecturally relevant requirements from their base-set in their solution strategy than have the non-SA-groups.

4.2.2.3 Importance The two types of groups also scored differently with regard to the level of importance of the requirements they produced (see Table 4), namely, the degree to which the success of the system depends on the implementation of a requirement. Upon closer examination, the difference for this characteristic is similar to that for *architectural relevance* in that the SA-groups had more requirements that scored 6 or 7 (quite high and extremely high) was 255 (59%) than the non-SA-groups (216) [50%]. The non-SA-groups had more requirements that scored 1–3 (extremely low to slightly low) (62) [14%] than the SA-groups (27) [6%]. Both types of groups scored closely for 4—neither high nor low (42 [10%] for the SA-groups, 46 [11%] for the non-SA-groups) and 5—slightly high (106 [25%] and 109 [25%]).

This shows that the SA-groups were better able to elicit the kinds of requirements that would be important to the success of the system than the non-SA-groups. This result is surprising because *importance* of a requirement for system success is not influenced only by its technological or user orientation. Other influencing factors include return on investment, cost, implementability, resource consumption, etc. Rather, these orientations are simply orthogonal. One would thus not expect a statistically significant difference between the SA and non-SA groups. The cause of the difference thus calls for further investigation.

4.2.2.4 Categories with no difference The means of the ratings for the requirements from the SA-groups and non-SA-groups were similar in a number of categories: *focus on time*, *cost*, and *quality*; *implementability*; and *focus on client needs*. *Testability* was higher for the SA-groups at $p = 0.06$, close to a statistically significant result, and so it is a characteristic of interest for future studies. Thus, for these six characteristics, there is no evidence to support the rejection of the null hypothesis (*the presence of a SA has no impact on characteristic*—see Sect. 3.1).

Table 5 Focus on technological needs versus user needs in the two types of groups

	SA-groups (%)	Non-SA-groups (%)
Higher focus on user needs	37	46
Equal focus	10	14
Higher focus on technological needs	53	42

What this means is that the characteristics of the requirements gathered in the presence or absence of the architecture by the respective two types of groups were statistically not different in so far as these six particular characteristics are concerned.

4.2.2.5 Abstraction and type Two of the requirements characteristics (see Table 1), *requirement type* and *level of abstraction*, were rated on a nominal and an ordinal scale, respectively.[5]

For the characteristic *level of abstraction*, the SA-groups had more requirements that scored high values (5 and 6) (71 [16%] vs. 38 [9%]), which denote requirements having a greater breadth of impact (sub-system level, system level, inter-system level). The non-SA-groups had more requirements with lower scores (0 and 1) (58 [13%] vs. 37 [9%]), indicating a small breadth of impact (module or component level). The NHST reveals that the difference in frequency counts is statistically significant ($p = 0.001$) so we conclude that there is evidence to support the rejection of the null hypothesis for this requirements characteristic.

An inference of the SA-groups' higher score on the *level of abstraction* quality over non-SA-groups' is that, not having access to the architecture, the non-SA-groups were dealing with requirements at a functional level which are dealt with at an individual component's level. On the other hand, the SA-groups' requirements are more cross-cutting across the architecture, since they elicited requirements that dealt with the integration of new sub-systems and components with the existing SA.

For rating the characteristic *requirement type*, requirements were analyzed according to a priori nominal categories from standard RE literature [24]. Examples of these categories include *usability*, *performance*, *delivery*, *reliability*, etc. The dataset shows that the SA-groups produced more implementation (62 vs. 50) and interoperability (51 vs. 30) requirements; whereas the non-SA-groups produced more usability (51 vs. 20) and functional (99 vs. 81) requirements. NHSTs for this characteristic (*type of requirement*) show that the differences between the two types of groups are statistically significant ($p = 0.013$) so we conclude that there is evidence to support the rejection of the null hypothesis for this requirements characteristic.

The observation here is consistent with our earlier observation (see Sect. 4.2.2.1) that the SA-groups were more focused on back-end technical issues; whereas the non-SA-groups were more focused on user issues.

4.2.3 SA aspects vs. requirements characteristics

To answer Q2 (see Sect. 3.1), we can now link aspects of the SA identified by the SA-groups as affecting the specific requirements characteristics that showed a statistically significant difference (see Sect. 4.2.1). First, requirements of the SA-groups will be divided according to whether they were affected by the SA.[6] Thus, we now have three sets of data: (1) those of the SA-groups which *were* affected by SA, (2) those of the SA-groups which were *not* affected by SA, and (3) the requirements of the non-SA-groups. This division will be used to show that the differences measured between the two types of groups are a direct result of the requirements being affected by the architecture, that is, the presence or absence SA is the "cause" of the specific types of differences observed between the two types of groups.

To confirm that the six requirements characteristics identified in Q1 (see Sect. 3.1):

- focus on user needs, focus on technological needs, architectural relevance, and importance
- type and level of abstraction.

were indeed affected *because* of the SA, we now qualitatively examine the differences in the three sets of data identified previously (1, 2 and 3) with respect to these six requirements characteristics.

Subsection 4.2.3.1 discusses these six requirements characteristics and the differences with respect to them in the three identified groups, and Subsect. 4.2.3.2 relates specific architectural aspects and their impact on the above-mentioned six requirement characteristics.

4.2.3.1 Causal impact of SA on requirements characteristics Earlier findings (see Sect. 4.2.2) indicated that requirements from the SA-groups generally scored lower with regard to *focus on user needs* and higher with regard to *focus on technological needs*, *architectural relevance*, and *importance*. Table 6 shows this along with the statistical significance of this new distribution.

On average, when the architecture was not affecting a requirement (188 requirements), both the SA and non-SA

[5] Because of the scale types used for these two attributes, the Cohen Effect Size tests cannot be applied.

[6] In total, there were 148 requirements that were *enabled* by the architecture, 126 requirements that were *constrained* and 51 requirements that were *influenced*. However, since we found no statistically significant differences between enabled, constrained and influenced requirements they will simply be grouped as *affected requirements*.

Table 6 Characteristics of requirements and the architecture's effect

Group	Role of SA	Number of requirement's.	Focus on user needs	Focus on tech. needs	Arch. rel.	Importance
Non-SA	Not affected	458	3.63	3.42	4.12	5.28
SA	Not affected	188	3.56	3.69	4.24	5.37
	Affected	255	3.1	4.4	4.8	5.8
NHST *p*-value		Chi-square	0.046	0.001	0.001	0.003

groups scored similarly for three characteristics: *focus on user needs*, *architectural relevance*, and *importance*. For these three characteristics, it can now be seen that the differences in means that were observed earlier between the two types of groups (SA-groups: 3.26, 4.59, and 5.63 vs. Non-SA-groups: 3.65, 4.12, 5.28, respectively—see Table 4) were caused almost entirely by *affected requirements* (255 in Table 6). That is, we see "decreased" architectural effect on the characteristic *focus on user needs*, and "increased" effect on the characteristics *architectural relevance*, and *importance*. For *focus on technological needs*, the difference in means reported earlier (SA-groups: 4.12 vs. Non-SA-groups: 3.42 in Table 5) was not caused entirely by *affected requirements* but was certainly augmented by them (SA-group not affected: 3.69 vs. SA-group affected: 4.4 in Table 6).

For the characteristics *level of abstraction* and *type*, we did not find any causal link between (1) the differences between the two groups (see Sect. 4.2.2) and (2) the SA.

4.2.3.2 Architectural source of the impact In the previous section, we saw which particular requirements characteristics were affected by the SA (see Table 6). However, we do not know as yet which particular aspects of the SA were the causes of those effects—this is the focus in this subsection. Determining specific aspects of the SA affecting particular requirements characteristics can help during future elicitation of requirements, for example, in being vigilant about any architectural implications on development cost, system quality and schedule and, accordingly, negotiate the requirements with the stakeholders.

In Table 7, we can see that four specific SA aspects affected requirements that exhibited the properties in Table 6: *existing hardware*, *non-functional characteristics* (*same sub-system*), *non-functional characteristics* (*different sub-system*), and *architectural patterns*. Requirements affected by one of these four architectural aspects showed lower mean values for *focus on user needs* and higher mean values of *focus on technological needs*, *architectural relevance*, and *importance*. This suggests that these four architectural aspects had a substantial and consistent impact on the requirements elicited by the SA-groups.

Modifiability was another SA aspect that affected requirements exhibiting three of the four characteristics (i.e., scored higher means than the groups not affected by SA): *focus on technological needs* (*4.26*), *architectural relevance* (*5.12*), and *importance* (*5.98*). However, unlike the four SA aspects described earlier, these requirements scored *higher* for *focus on user needs* than did requirements which were not affected by the architecture (3.95 vs. 3.56). Without further data and analysis, it is difficult to discern the extent of the impact of *modifiability* on *focus on user needs*.

Table 7 The source of architectural effects and the requirements characteristics affected

Group	SA role	SA aspect	Requirements characteristics				
			Number of requirement's	Focus on user needs	Focus on tech. needs	Arch. relevance	Importance
Non-SA	None	N/A	458	3.63	3.42	4.12	5.28
SA	None	N/A	188	3.56	3.69	4.24	5.37
	Affected	Existing hardware	9	3.00	5.67	5.11	5.89
		NF characteristics (same sub-system)	53	1.94	5.32	4.77	5.85
		NF characteristics (different sub-system)	100	3.29	4.55	4.88	5.93
		Arch. Patterns	25	2.88	3.80	4.54	6.25
		Modifiability	42	3.95	4.26	5.12	5.98

The numbers in this table do not equal the totals in Table 6 because Table 6 includes requirements affected by all nine identified architectural aspects, whereas Table 7 contains only those five aspects that had an impact on requirements with differing characteristics

4.3 Summary of results

The following are the key results of the study:

- Given access to the architecture, the analysts tend to elicit approximately 10% more technologically oriented requirements; without access to the architecture, they tend to elicit approximately 10% more user needs–oriented requirements.
- For a given type of group (SA or non-SA), there is an inverse relationship within the set of elicited requirements between the characteristics technological needs and user needs; as the quantity of one increases the other decreases (e.g., requirements focused on user needs have less focus on technology needs—46 vs. 37%, respectively).
- Given access to the architecture, the analysts tend to elicit 10% more architecturally relevant and 10% more important requirements.
- Given access to the architecture, the analysts elicited approximately 7% more abstract requirements (i.e., those with cross-cutting concerns across system requirements) than analysts without access to the architecture.
- Given access to the architecture, the analysts elicited more *implementation* (62 vs. 50 requirements) and *interoperability* (51 vs. 30) requirements.
- Without access to the architecture, the analysts elicited more requirements of type *usability* (51 vs. 20) and *functionality* (99 vs. 81).
- Specific architectural aspects were identified that affected the requirements characteristics: *Existing hardware*, *NF characteristics* (*same sub-system*), *NF characteristics* (*different sub-system*), *architectural patterns*, and *modifiability*.

Until now, there was no scientific data on the above-mentioned issues. This can therefore be considered an important step toward building a grounded theory on the impact of SA (or non-SA) on RE. While the findings may be interesting, it is rather unfortunate that the various SA and non-SA groups had long disbanded and therefore not accessible by the time the data analysis was completed. Thus, we could not obtain their perspective of our inferences. The next section discusses example implications of our findings.

5 Implications

We discuss the implications of the described findings on such issues as RE process engineering, post-requirements analysis, RE tools, traceability management, and further empirical work in RE.

5.1 RE process engineering

The findings described in Sect. 4.3 raise some interesting questions, such as should the SA always be used in the RE process as promoted by the literature [27, 38], and as strongly supported by our survey results (see Sect. 1). Could there be some conditions when it would be advisable *not* to use SA in RE? Of course, the considerations behind these questions are such factors as project costs, time-to-market, system quality, profitability, innovation, sustainability, and human factors.

We have identified three key cases which merit consideration in the design of RE processes: (1) new innovative system; (2) mature system; and (3) system with user and technology balance.

When evolving a relatively new product, the business strategy might be to focus more on innovative features (so as to attract a large customer-base) than on refining technical system attributes such as reliability, security, performance (so as to stabilize the system) [34]. In this scenario, management can determine whether the cost/benefit of the RE process would be better if the influence of the existing SA on elicited requirements was *omitted* or *minimized*, if not entirely in the RE process then at least at the outset of the RE process.

Likewise, in a mature product, with a large-dependent customer-base, it would be imperative that new requirements do not destabilize system quality and, accordingly, it would be advisable to use SA in RE. For example, we saw that involving SA in RE led to more global or architecturally relevant requirements (see Sect. 4.2.2.2), whereas, absence of SA in RE in this scenario could lead to myopic requirements. In turn, this could lead to increased development backtracking to fix architectural problems or duplication of features (both functional and non-functional) across the system [23].

In the cases where there is a need for a user- *and* technology-focused requirements, we need to design the RE process to have mechanisms built into ensure that the characteristics of the requirements are not lop-sided in favor of SA and against user-focus (or vice versa). For example, in Sect. 4.2.2.1, we saw that involving SA in the RE process short-changed user-focused requirements. Consequently, not having vigilance about the impact of SA on RE could result in system qualities that may not satisfy diverse stakeholders' interests.

These are but specific examples. There are many project and organizational factors that could influence when to utilize the SA in a RE process, for example: size and competency of the development team, development paradigm used (e.g., agile vs. iterative), familiarity of SA by the development team, budget, etc. Therefore, in deploying the RE process, these multitude of factors

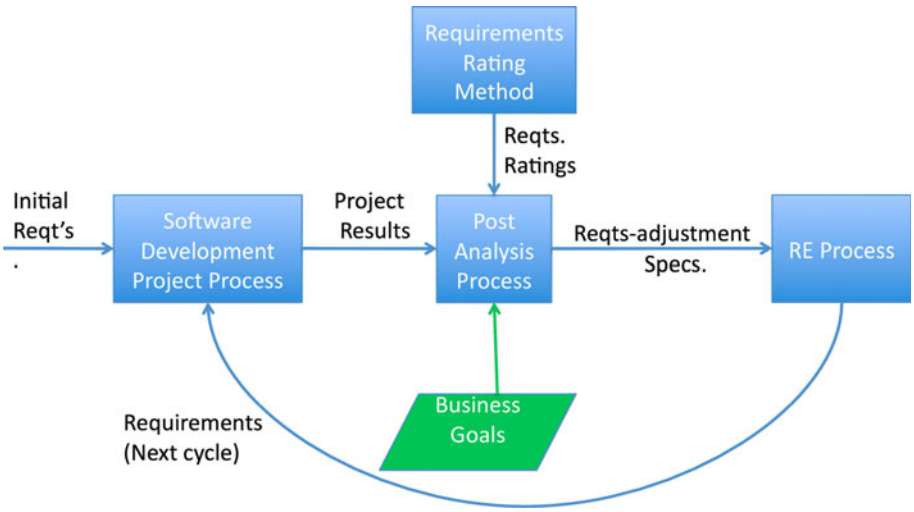

Fig. 1 Integration of requirements rating method in software development process

should be considered when planning the inclusion of the SA in the RE process.

5.2 Post-requirements analysis

As permitted by a project-specific situation, it is prudent to examine the requirements being elicited – as a post-RE or post-project exercise – to detect any biases counteracting business goals, which could then be adjusted in the subsequent elicitation efforts. For example, by integrating the rating (see Table 2 in Sect. 3.6.1) into post-requirements (or post-development) analysis, a project could gather (release-based or timed) quantitative data on requirements characteristics. This would create a history of the "flavor" of the system, as dictated by the characteristics of the requirements (e.g., trends on quality attribute biases, user needs focus, technological focus.). By analyzing such trends, management can assess, periodically, whether the evolving system is aligned with the current and future organizational goals. For example, a trend heavily in favor of "user-focus" coupled with heightened architectural defects could indicate inadequate consideration of SA during the RE (and development) processes. Based on such analysis, tweaking the requirements and the associated RE processes can help align the requirements characteristics to the system and business goals. Figure 1 shows an example process model of how the requirements rating method could be integrated in a software development process.

Unlike other implications in this paper, this one is rooted in the empirical procedures of the study and not the findings. There is precedence for this idea. For example, in software effort estimation [21], the collection and analysis procedures of software defects, size and effort data has been integrated into a software estimation tool. Likewise, in software process improvement [10], the authors propose the use of historical, exploratory research studies to identify process improvement opportunities in industrial projects. Finally, in software maintenance [33], the analysis of software metrics (such as development effort, defaults, and component changes) in past releases of a software project is used to automatically identify components most likely to cause rework.

5.3 RE tools

The realization that specific characteristics of new requirements are affected by particular aspects of existing system architectures (as depicted in Table 7) opens up possibilities for a new generation of RE tools. In particular, requirements management tools (such as DOORS and Requisite pro) and goal-oriented modeling tools (such as *i** [43] and KAOS [25]) could be enhanced with a product-centric knowledge-base that accumulates, over a span of many releases, how different aspects of the evolving system architecture affect requirements characteristics (e.g., as shown for one evolutionary iteration in Table 7). Such tools thus would have a characteristic of becoming more and more mature over time while enhancing the elicitation, analysis, and reasoning capabilities of the requirements engineers.

5.4 Traceability management

Recent work in [16] encourages value-based requirements traceability, as opposed to early research which attempted at all encompassing traceability (implemented in tools such as DOORS and Requisite Pro) which is known to be wasteful in terms of future use, and thus has not gained wide acceptance in practice [1]. In this paper, we show how empirical studies can lead to discovery of targeted product

dependencies and relationships (e.g., between SA and requirements) that can be worth tracking during software evolution.

In Table 6, for example, we see that 255 requirements (58%) were affected by the SA. Also, Table 7 indicates the particular SA aspects that affected the requirements with particular characteristics. Furthermore, previous research [28] has already shown the different effect types of SA on requirements (e.g., SA as a constraint on new requirements or as an enabler of new requirements). Thus, by collating these different pieces of information, it would be possible to create *targeted* traceable links. For example, by linking those architectural aspects (e.g., components with particular non-functional characteristics) that are historically known to constrain certain types of requirements, it could help in speeding up detection and analysis of *new* risky requirements that are in conflict with the baseline architecture. We can see that empirically derived targeted links would (a) reduce the burden considerably in making the select-few links in the first place and (b) render invaluable information upon usage of traceability tools during RE.

5.5 Further empirical studies in RE

Based on the findings of the exploratory study, we raise the following example emergent hypotheses:

H1 Requirements elicited from a RE process that involve analysis of a current architecture will be more technologically focused than a RE process that does not include such analysis.

This hypothesis emerges from the finding in Sect. 4.2.2.1 and improves upon the null hypothesis used in this study by providing a direction of the effect (i.e., SA analysis implies *more* technological focus). While the finding on technology focus vs. users-need focus may seem intuitive, further testing of this hypothesis in different domains and project contexts would not only confirm whether these results are generalizable across different population and settings (see Sect. 3.6.2) but would also indicate the variance in *extents* across them. This could help tune the RE process specific to the domain concerned.

H2 Requirements elicited from a RE process that do not analyze the current architecture will be more user-focused than a RE process that does not include such analysis.

In Sect. 4.2.2.1, the results show that requirements elicited in the absence of an existing architecture are more user-focused than requirements in the presence of an existing architecture. The motivation for this hypothesis follows from H1 above.

H3 Requirements elicited when the current architecture is analyzed are considered more important for system success than without such analysis.

This hypothesis is rooted in the finding from Sect. 4.2.2.3. This result is surprising and seems more based on factors outside of technology and user needs and is therefore difficult to discern whether this would hold across other project domains and business contexts.

H4 Requirements elicited when the current architecture is analyzed are more architecturally relevant than requirements without such analysis.

This hypothesis emerges from the finding in Sect. 4.2.2.4, and the motivation for this hypothesis follows from H1 above.

H5 An RE process that does not include analysis of current architecture will output more innovative requirements.

A requirement characteristic that was not measured in this study was innovation. This was due to the fact that the project domain was banking which is not a new domain, and thus it would be difficult to measure innovation in such a system. However, it is an important characteristic that we initially did want to investigate. Although this hypothesis is not directly derived from the findings, we include it here because this characteristic should be investigated if the domain permits. It seems that, by intuition, this characteristic could be affected (most likely constrained) by the analysis of a current architecture. The investigation of this hypothesis complements recent research effort [26] in improving the process of eliciting innovative requirements.

H6 A requirements elicitation team with motivation and expertise in system solution is more likely to elicit requirements that have technological bias regardless of the absence or presence of an existing system architecture.

This hypothesis emerges from the finding in Sect. 4.2.1 where it was shown that the simply being in a different requirements team had a significant effect on the technological bias of the requirements. One possible explanation for this phenomena occurring is that the motivation and expertise of the team members was more solution-oriented and accordingly biased the requirements.

H7 A requirements elicitation team with motivation and expertise in a system's context (e.g., human–computer interaction and end-user satisfaction) is more likely to elicit requirements that user-focused regardless of the absence or presence of an existing system architecture.

This hypothesis emerges from the finding in Sect. 4.2.1, and the motivation for this hypothesis follows from H6 above.

To test hypotheses H1–H5, controlled experiments with the same type of project setup as described in this paper (see Sect. 3) would need to be designed. The study design for testing H6–H7 would be more difficult in randomized controlled experiment, and thus, a quasi-experiment design would be more suitable for these cases.

6 Conclusions

While the role of software architecture (SA) in requirements engineering (RE) has been discussed several times over the past decade [19, 27, 38], no scientific studies have ever been conducted to explore the quantitative relationship between SA and requirements. In this paper, we describe a controlled study, involving 12 teams, investigating the characteristics of new requirements in the presence or absence of an existing SA.

In a nutshell, we found that of the 12 requirements characteristics identified (see Table 1), the following were significantly affected by the presence or absence of the SA (see Sect. 4.3): *focus on technological needs*, *architectural relevance*, *importance*, *level of abstraction*, *requirement type*, and *focus on user needs*. We did not find SA effects on the remaining characteristics (see Sect. 4.2.2.4). We then probed the results further from the perspective of the specific architectural aspects that affected these characteristics and found five such aspects (*Existing hardware*, *NF characteristics* (*same sub-system*), *NF characteristics* (*different sub-system*), *architectural patterns*, and *modifiability*). From these findings, we discuss potentially useful implications in the areas of RE process engineering, requirements alignment with business goals, RE tools, traceability management, and future empirical work based on four emergent hypotheses from this study.

While these findings contribute new scientific knowledge concerning SA-requirements interaction, let us not forget that this was only one exploratory controlled study on a particular domain (Banking system). Significant as it is, we advise caution when making business or project decisions based on the findings of this foundational study alone! Rather, we encourage other researchers to conduct confirmatory studies in other domains and contexts to help build grounded theory on the impact of SA on RE.

Though our study, in principle, can be replicated in both industry and academia, each of these contexts has its own limitations to consider when planning replications of the study. Conducting controlled studies in industry would be extremely difficult, if not impossible, due to the unavailability of equivalent projects and the inability to impose research control (i.e., presence or absence of SA in the RE process). More likely, the chances are better to conduct controlled studies and perform hypothesis testing in academia using qualified students as participants. This, however, does have the issue of being able to generalize the results to industrial contexts. Despite this threat, studies such as these are still a critical stepping-stone to conducting "case studies" in industry. Case studies in industry would be invaluable for providing an industrial perspective on the impact of SA on RE. These studies can be carried out by analyzing the existing requirements of projects where the RE processes have either involved SA or not. In particular, data from different projects with varying levels of complexity, and in different domains, would help solidify the body of knowledge in this research area.

Acknowledgments This work was, in part, supported by Natural Science and Engineering Research Council (NSERC) of Canada. We are also grateful to the study participants, and to the researchers who conducted the ratings of the requirements. Lastly, we are thankful to the anonymous reviewers for their helpful suggestions.

Appendix: requirements ratings data collection instrument

The requirements ratings data collection instrument was administered to each requirements rater to collect the requirements rating data (see Sect. 3.6.1). The instrument was operationalized through an MS Excel Spreadsheet file, and the organization of the spreadsheet is organized from the structure of Table 8 below.

Table 8 Requirements rating data entry template

Requirements information				Requirements characteristics					
Requirement. ID	Title	Description	Rationale	Cost	Time	Quality	User needs	Tech. needs	… (cont'd.)
R1									
Rater comments									
R2									
Rater comments									
R3									
Rater comments									

Essentially, each requirement takes up two rows of this table. The first row is where the information pertaining to the requirement is given to the raters and where they enter the ratings for the different requirements characteristics. Specifically, for each requirement, there are four pieces of information given to the reviewer: requirements ID, a title, a description, and a rationale. The requirement ID is a numerical value that uniquely identifies the requirement. The title explicitly indicates what part of the system the requirement is referring to (Tele-Banking, Wireless Banking, Web Banking or Interac.) The description is the requirement itself, and the rationale provides additional reasoning as to why the requirement is necessary. These four pieces of information are given in the first four columns of the Table. The next twelve columns[7] are where the rater enters their rating for the particular requirements characteristic given in the column header. The raters filled out this part of the instrument with reference to the list of requirements characteristics, their definitions, and the scales to use for each characteristic (see Table 1). In the second row for a given requirement, the rater can optionally leave any comments regarding their specific ratings for a particular requirement characteristic entry.

Note that in order to remove possible *researcher bias* during the ratings process, the Table does not contain any information that can associate given requirements with specific teams that elicited the requirements and whether they had access to the existing SA during their RE project.

The results of each individual rater's assessment are merged into another MS Excel sheet which is organized based on the structure from Table 2 in Sect. 3.6.1, where the inter-rater agreement procedure from Sect. 3.6.1 can be conducted.

References

1. Arkley P, Riddle S (2005) Overcoming the traceability benefit problem. In: 13th IEEE international conference on requirements engineering (RE'05), Paris, France, pp 385–389
2. Basili VR, Weiss D (1984) A methodology for collecting valid software engineering data. IEEE Trans OnSoft Eng, pp 728–738
3. Bass L, Clements P, Kazman R (2003) Software architecture in practice, 2nd edn. Addison-Wesley, Reading
4. Bayer J, Muthig D, Widen T (2000) Customizable domain analysis. In: The proceedings of the first international symposium of generative and component-based software engineering (GCSE '99), Lecture Notes in Computer Science, Springer, pp. 178–194
5. Berander P (2004) Using students as subjects in requirements prioritization. In: Proceedings of the 7th international conference on empirical assessment & evaluation in software engineering, pp. 95–102
6. Breitman K, Sampaio do Prado JC (2000) Scenario evolution: a closer view on relationships. In: Fourth international conference on requirements engineering (RE'00), Illinois, US, pp 95–107
7. Burgess C, Dattani I, Hughes G, May J, Rees K (2001) Using influence diagrams to aid the management of software change. J Requir Eng 6(3):173–182
8. Carver J, Shull F, Basili V (2003) Observational studies to accelerate process experience in classroom studies: an evaluation. In: Proceedings of the 2003 international symposium on empirical software engineering (ISESE '03), Rome, Italy, pp 72–79
9. Castillo E, Gutiérrez M, Hadi A (1997) Learning bayesian networks, *Expert systems and probabilistic network models*. Monographs in computer science. Springer, New York, pp 481–528
10. Cook J, Votta LG, Wolf AL, Wolf AL (1998) Cost-Effective analysis of in-place software processes. IEEE Trans Softw Eng 24(8):650–663
11. El Emam K, Madhavji N (1995) Measuring the success of requirements engineering processes. In: Proceedings of the 2nd IEEE international symposium on requirements engineering, pp 204–211
12. Etien A, Salinesi C (2005) Managing requirements in a co-evolution context. In: 13th IEEE international requirements engineering conference (RE'05), Paris, France, pp 125–134
13. Ferrari R, Madhavji NH (2008) Software architecting without requirements knowledge and experience: what are the repercussions? J Syst Softw 81(9):1470–1490
14. Ferreira S, Collofello J, Collofello J, Shunk D, Mackulak G, Mackulak G (2009) Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. J Syst Softw 82(10):1568–1577
15. Fusaro P, El Emam K, Smith B (1997) Evaluating the interrater agreement of process capability ratings. In: Proceedings of the 4th international software metrics symposium, pp 2–11
16. Heindl M, Biffl S (2005) A case study on value-based requirements tracing. In: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on foundations of software engineering, Lisbon, Portugal, pp 60–69
17. IEEE SWEBOK (2004) Guide to the software engineering body of knowledge: 2004 Version. IEEE and IEEE Computer Society project. <http://www.swebok.org/>
18. Host M, Regnell B, Wohlin C (2000) Using students as subjects—a comparative study of students and professionals in lead-time impact assessment, Empirical Software Engineering, pp 201–214
19. Jackson M (1994) The role of architecture in requirements engineering. In: Proceedings of the 1st international conference on requirements engineering (RE '94'), pp 241
20. John I, Muthig D, Sody P, Tolzmann E (2002) Efficient and systematic software evolution through domain analysis. In: 10th IEEE joint international requirements engineering conference (RE '02), Essen, Germany, pp 237–245
21. Johnson PM, Moore CA, Dane JA, Brewer RS (2000) Empirically guided software effort guesstimation. IEEE Software 17(6)
22. Johnson RB, Christensan L (2003) Educational research: quantitative, qualitative and mixed approaches. www.southalabama.edu/coe/bset/johnson/dr_johnson/2lectures.htm. Date last accessed June 2009
23. Kamiya T, Kusumoto S, Inoue K (2002) CCFinder: a multilinguistic token-based code clone detection system for large scale source code. IEEE Trans Softw Eng 28(7):654–670
24. Kotonya G, Sommerville I (1998) Requir Eng. Wiley, England
25. Van Lamsweerde A (2003) From system goals to software architecture. In: Bernardo M, Inverardi P (eds) Formal methods for software architectures. LNCS 2804, Springer, Berlin, pp 25–43

[7] Due to readability of the template, not all columns with characteristics are shown. See Table 1 for full list.

26. Maiden N, Manning S, Robertson S, Greenwood J (2004) Integrating creativity workshops into structured requirements processes. In: Proceedings of the 5th conference on designing interactive systems: processes, practices, methods, and techniques, Cambridge, MA, US, pp 113–122
27. Mead N (1994) The role of software architecture in requirements engineering. In: Proceedings of the 1st international conference on requirements engineering, p 242
28. Miller J, Ferrari R, Madhavji NH (2008) Architectural effects on requirements decisions: an exploratory study. In: 7th working international conference on software architecture, Vancouver, pp 231–240
29. Miller J, Ferrari R, Madhavji NH (2009) Characteristics of new requirements in the presence or absence of an existing system architecture. In: 17th international conference on requirements engineering (RE '09), Atlanta, United States, pp 5–14
30. Nuseibeh B (2001) Weaving together requirements and architectures. IEEE Comput 34(3):115–117
31. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the 22nd international conference on software engineering (ICSE), pp 27–46
32. Pett MA (1997) Nonparametric statistics for health care research: statistics for small samples and unusual distributions, 2nd edn. SAGE, Beverley Hills
33. Porter AA, Selby RW (1990) Empirically guided software development using metric-based classification trees. IEEE Softw 7(2):46–54
34. Rajlich VT, Bennett KH (2000) A staged model for the software life cycle. IEEE Comput 33(7):66–71
35. Rao PV (1997) Statistical research methods in the life sciences. Brooks/Cole, Belmont
36. Rolland C, Salinesi C, Etien A (2004) Eliciting gaps in requirements change. J Requir Eng 1:1–15
37. Shaw M (2003) Writing good software engineering research papers: minitutorial. In: Proceedings of the 25th international conference on software engineering (ICSE 2003), Portland, USA, Tutorial Session, pp 726–736
38. Shekaran C (1994) Panel overview: the role of software architecture in requirements engineering. In: Proceedings of 1st international conference on requirements engineering, p 239
39. Software Requirements to Architectures Workshop (STRAW'01 & '03) (2001 and 2003) June 2001, Toronto, Canada; May 2003, Portland, USA
40. Tichy WF, Lukowicz, Prechelt L, Ernst A (1995) Experimental evaluation in computer science: a quantitative study. J Syst Softw (January), 1–18
41. Vilella K, Doerr J, Gross A (2008) Proactively managing the evolution of embedded system requirements. In: 16th international conference on requirements engineering (RE '08), Delhi, India, pp 13–22
42. Wieringa RJ, Heerkens J (2006) The methodological soundness of requirements engineering papers: a conceptual framework and two case studies. Requir Eng J 11:295–307
43. Yu E (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the 3rd IEEE international symposium on requirements engineering (RE'97) January 6–8, 1997, Washington, DC, USA, pp 226–235