Empirical Study of Analysts’ Practices in Packaged Software Implementation at Small Software Enterprises

Issam Jebreen and Ahmad Al-Qerem
Faculty of Information Technology, Zarqa University, Jordan

Abstract: This study investigates the practices of Requirements Engineering (RE) for packaged software implementation, as enacted by Small Packaged Software Vendors (SPSVs). Throughout the study, a focus on the actions carried out by SPSV analysts during RE is maintained, rather than a focus on the actions of client companies. The study confirms assertions in the literature, finding that most contemporary RE practices are unsuitable for SPSVs. The research investigated the means by which SPSVs can adopt, follow and adapt the best possible RE practices for Packaged Software Implementation (PSI), an explanation of the collection of qualitative and quantitative data during an case study in packaged software vendors. The research findings lead to introduced new methods of documentation, was not as concerned as general RE practice with looking for domain constraints or with collecting requirements and viewpoints from multiple sources, was more likely to involve live software demonstrations and screenshots to validate user needs, and was more likely to involve the compilation of a user manual. In PSI, prioritising requirements is not a basic practice; instead, analysts collect requirements in a circular process, with managers then directing analysts regarding which requirements to direct most attention toward. PSI was also found to place emphasis on assessing requirements risks and on considering the relationship between users’ needs and the inter-relationships between software functions, as analysts engaging in PSI do not wish to disrupt functions of their software when making modifications in response to client requests.

Keywords: Requirement engineering; packaged software implementation; ERP; analysts’ practices SMEs.

Received February 15, 2017; accepted May 10, 2017

1. Introduction

Small to Medium-sized Enterprises (SMEs) are considered to be fundamentally different from Large Enterprises (LEs) in several respects [14]. Some distinguishing characteristics of SMEs include ownership type, culture, structure, and market orientation [15]. Other researchers have found that when it comes to IT/IS adoption, SMEs are constrained by limited resources and limited IS knowledge, or by a lack of IT expertise [14]. Given that studies of Packaged Software Implementation (PSI) have argued that findings about implementations in large companies cannot be applied to SMEs [22]. These distinguishing characteristics of SMEs may influence the PSI issues they face [22]. Despite the importance of PSI being recognized by former studies, there has been little research exploring this issue further. In particular, discussions about SMEs rarely occur in the literature about PSI, and how the structure of SMEs shapes the software throughout its life cycle of implementation is rarely mentioned [8]. One question related to this issue is which processes Small Packaged Software Vendors (SPSVs) apply in order to identify misalignments between packaged software functionality and client requirements and how they deal with these misalignments. Implementation consists of the customization, installation, configuration and adaptation of the packaged software acquired according to the needs of the organization, and a better gap/fit analysis between the organization's needs and the functionality of packaged software can be achieved through Requirements Engineering (RE).

However, several software engineering researchers have argued that most current requirements engineering practices are unsuitable for SPSVs [9] and that SPSVs are unable to successfully apply RE methods and techniques that have been designed for larger companies [10]. For this reason, research into RE practices should focus more on the investigation of what RE is practiced in smaller companies such as SPSVs and on how the RE methods followed by SPSVs can be improved. This study therefore features a case study of what RE practices of packaged software implementation is enacted in Small to Medium-sized Software Development Companies. In this study, packaged software is defined as ready-made software products that generally require modification or customization for specific organizations. They are often exemplified by Enterprise Resource Planning (ERP).

2. Literature Review

The poor use of RE practices has often been identified as one of the major factors that can jeopardise the
success of a software project [2]. Meanwhile, researchers have also recognized that following appropriate RE practices contributes to the success of software projects [19]. For example, Aranda et al. [2] state that gathering and managing requirements properly are key factors when it comes to the success of a software project. There is a general critical consensus that RE practices plays a very important role in the success or failure of software projects [16].

However, it is not possible to improve RE practices until areas that need improvement in an organization’s current RE practice have been identified. Meanwhile, the solution for improving RE practices will be different in each company; it has been found that a one-size-fits-all approach does not work in such a scenario [16].

Merten et al. [16] argue that previous findings suggest that SPSVs may not actually need to have extremely formal and conventional forms of RE. Instead, “light organization and unconventional RE” may work better for many SPSVs. They also discuss the various RE models that have been provided in previous studies.

Furthermore, the conclusion of the 1st Workshop on Requirements Engineering in Small Companies (2010) was that “existing RE techniques are not sufficient for small companies” and that some other factors need to be realised, such as that size is “not the only measure to categorize smaller companies and describe the exact focus of research”, “that tacit knowledge and social structures” in place in SMEs [SPSVs] may play an important role in RE research, that introducing RE methods designed for larger companies may actually harm the specific features of an SME [SPSVs], and that RE methodologies need to be made more lightweight.

In many cases software engineering no longer involves building systems from scratch, but rather integrating “existing frameworks and modules” or working with a “comprehensive code base” [6].

Software engineering has a group of influential approaches that are often considered good practice. These involve such practices as “structured programming”, “stepwise refinement”, and collecting “a complete set of test cases” [6], but these approaches don’t apply for PSI. Dittrich et al. [6] argues that such implementation requires “independent consideration”.

Often, SPSVs are unable to apply RE methods and techniques without modifications [2, 16]. In addition, shortcomings in applying RE methods due to time constraints or limited resources may arise [2]. Therefore, “RE research has to intensify the investigation of RE practices in SMEs [SPSVs].”

Otherwise SMEs [SPSVs] will have to continue their search for methodical orientation and dedicated tool support. Normally, the people responsible for requirements in SMEs [SPSVs] are ambitious, but suffer from scarcity of resources. Their time for doing experiments and trying different methods is very limited. They “need quick methodical improvement of requirements elicitation, documentation, communication and traceability as well as more continuity of requirements management through the whole software lifecycle” [4]. According to past literature, there is a clear problem related to whether SPSVs can carry out effective RE.

Meanwhile, Karlsson et al. [11] state that there are “several studies that concern or include RE issues. However, none of these focus primarily on packaged software development and implementation. Furthermore, in most of these studies, the studied projects and organizations are mainly large, both in terms of the number of persons and requirements involved, and in terms of the duration of the projects”.

Quispe et al. [17] highlight that “there is a lack of knowledge about the requirements engineering practices in these types of companies [small-medium]”. Researchers in the field encounter a general lack of information when it comes to gaining knowledge about how RE is carried out in packaged software companies. It is in fact difficult for researchers to gain much knowledge about how SPSVs carry out RE given that most SPSVs seldom request external support, probably due to limited finances. However, RE research should eventually enable those companies to become aware of more state of the art or innovative RE techniques and to be able to improve their RE practice without external help [16].

One core question that remains, despite the work done in previous studies, is: what are the RE practices of packaged software implementation enacted in SPSVs? This study set out to understand packaged software implementation (customization, installation, configuration and adaptation) in terms of RE practice at SPSVs. In the context of packaged software implementation by small packaged software vendors, the research question that shall be answered through this study is: What are the analysts’ practices?

3. Research Approach

A case study research method was applied in small packaged software vendors who participated in this study. Organization considered in this study is dominated by the provision of packaged software solutions. Case study is a research strategy which focuses on understanding the research hypothesis and theory [7]. It aims to provide description, test theory generate theory. Case study can involve either single or multiple cases, and numerous levels of analysis [21].

Typically the methods for data collection are archives, interviews, questionnaires, and observations [21]. The interest of this study is to provide description from the analysis of the results. The case study is thought to be “ideally suited to the needs and resources of the small-scale researcher” who might work on one
organization, or “one element of such an organization” or “one individual, or a small number of individuals” [21].

3.1. Data Collection

In the last decade focus group interviews have become a commonly used technique to collect qualitative data by asking the participants about their perceptions, beliefs, opinions, or attitudes regarding a concept, idea, service, or product [13]. While social research typically adopts direct observation, focus groups are more appropriate for studies of attitudes and experiences. The communication between participants in the focus group allows the researcher to gain access to various areas for studies and raises unexpected issues for exploration [13]. Focus groups are used as a self-contained method as well as in addition to other research methods, like in-depth interviews [13].

Before conducting a focus group, we identified the major objectives of the meeting and developed the main questions relevant to the initial results of the data analysis and research questions. The discussion session normally lasted an hour or an hour and a half, during which we generally one topic of RE practices. To plan the session, we needed to schedule a time when all the participants could attend. The organization’s conference rooms were used, which allowed all of the participants to see each other. The main ground rules we followed during a focus group were to remain focused on the research topic while the discussion flowed and evolved, to maintain momentum, and to achieve closure of questions. The focus group meeting agendas always included: welcoming the participants, reviewing the agenda and goals of the meeting, explaining the means of recording the session, introduction, conducting a questions and answers period, and wrapping up. During the main part of the session, we might sit back and listen to the discussion.

3.2. Data Analysis

We assess the enacted RE practices according to a framework adopted from Sommerville and Sawyer [20] and Cox et al. [5]. We use four levels of assessment of RE practices (as theorized by Sommerville and Sawyer [20]). These levels of assessment are the following: standardized use, common use, discretionary use, and never used.

- **Standardized Use (SU):** This practice has a documented standard and is always followed as part of the organization's software development process i.e., it is mandatory.
- **Common Use (CU):** This practice is widely followed in the organization but is not mandatory.
- **Discretionary Use (DU):** This practice is used at the discretion of individual project managers. Some may have introduced the practice for a particular project.
- **Never Used (NU):** The practice is never or rarely applied.

The tables use guideline classifications relating to ‘good requirements practices’ to be ‘basic’, ‘intermediate’, or ‘advanced’. ‘Basic’ practices can continually be repeated, and it is possible to estimate costs, time, and resources associated with these practices. In our assessment of the RE practices involved in PSI, ‘basic’ practices match with ‘standardized’ use. Meanwhile, ‘intermediate’ practices are more complex and lead to a ‘defined’ requirements engineering process. They can be considered as aligning with ‘common’ use. Lastly, ‘advanced’ practices are designed to help support continuous improvement within any RE process. Some of these practices involve advanced technology and advanced methods which require specialist knowledge. They may also involve expectations of and guidelines for organizational change. In our assessment of RE practices, ‘advanced’ practices align with ‘discretionary’ use.

4. Findings and Discussion

The results discussed in the following section were obtained by our summarizing and synthesizing the findings we derived from the data collected during Focus Groups held with analysts, and from our descriptive analysis of analysts’ practices in those organisations. During the Focus Groups we aimed to collect information from the analysts regarding how they carried out various RE practices when implementing packaged software. During the Focus Groups we asked, in turn, about requirements documentation practices, requirements elicitation practices, requirements analysis and negotiation, practices involved with describing requirements, practices involved with system modeling, requirements validation practices, and requirements management practices. The following sections contain (in italics) some reproductions of our summarizing findings (via descriptive analysis) determined as a result of the Focus Groups and written up at the end of each day, and descriptive analysis of analysts’ practices.

Furthermore, we obtained our results relating to specific RE practices by discussing with analysts a list of practices adopted from Sommerville and Sawyer [20] and Cox et al. [5] and asking them to rate their use of each according to the four levels of assessment derived from Sommerville and Sawyer [20]. This helped us to understand the level of use of each named practice. This was understood by the analysts; however, we did run into the issue that the analysts we spoke to did not always refer to the practices they used using the same terminology as ourselves. They therefore had issues with recognizing some of the
names that we had supplied relating to RE practices. With each group, we therefore had to explain what was meant by each of the named practices on the list. Once this was done, we proceeded to discuss each RE practice on the list. A total of 8 analysts took part in these Focus Group discussions and the rating of the practices.

4.1. Requirements Documentation Practices

The requirements document itself is a document that effectively communicates requirements to customers, managers and developers. As can be seen from Table 1, the levels of all requirements documentation practices are considered ‘basic’ by Sommerville and Sawyer [20] and Cox et al. [5]. When the use of these practices in PSI is investigated, it can be found that many of these practices receive Standardized use in PSI. Thus they are practiced at a very similar level to that suggested by Sommerville and Sawyer [20] and Cox et al. [5].

The following excerpt illustrates the origin of two of these practices as captured in our analysis:

These new practices involve creating a users’ needs/misalignments specification document (referred to as RD9) that analysts sent to developers within their software company. Other new practices that the analysts told us of are estimating the time related to creating the users’ needs/misalignments document (RD10) and estimating the cost needed for creating a users’ needs document (RD11). The scopilong process carried out before beginning to develop any software involves software analysis through the discussion of high level modification requirements and new features. Including a users’ needs validation document (RD2) was another new practice that was identified. Analysts told us that using the printouts of the software function screens to add users’ needs is a strategy that they utilize during the requirements validation practice for packaged software implementation.

All of these practices are a part of PSI, being practiced with ‘standardized use’. Despite their being used so often and the fact that these practices are perceived as having a high value, these practices have not been identified during previous studies of RE and packaged software RE; this study therefore extends the current framework for requirements practices in term of PSI.

4.2. Requirements Elicitation Practices

Requirements elicitation is defined as a group of practices designed to help discover the requirements for a system. These practices are followed by analysts in order to elicit requirements from the stakeholders related to the system. However, the requirements elicited also depend on the application domain and on the organizational and operational environments of the system.

In Table 3 we see that, several RE elicitation practices are carried out at the ‘basic’ level; that is, they are almost always practiced. Practices RE1 through RE6 are practiced with ‘standardized’ use in PSI. However, just over half of the practices operate at the ‘intermediate’ or ‘advanced’ levels. Most of those practices that are ‘basic’ in RE are standard practices in PSI.

<table>
<thead>
<tr>
<th>No</th>
<th>RE Practices</th>
<th>Type</th>
<th>PSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD9</td>
<td>Users’ needs/Misalignments specification document</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RD10</td>
<td>Estimating time needed for users’ needs document</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RD11</td>
<td>Estimating cost needed for users’ needs document</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RD12</td>
<td>Include users’ needs validation document</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
</tbody>
</table>

Table 2. New Requirements documentation Practices-PSI.
RE practices RE7 through RE13, shown in Table 3, are practiced rather differently in PSI than how they are presented by Sommerville and Sawyer [20] and Cox et al. [5]. In general RE, a large range of practices could be considered as ‘intermediate’ practices, that is, they are more complex and not always practiced.

Those practices regarded as ‘intermediate’ include looking for domain constraints, recording the requirements rationale, collecting requirements from multiple viewpoints, prototyping poorly understood requirements, using scenarios to elicit requirements, and defining operational processes. In PSI, however, these practices are carried out at a range of levels. For example, prototyping poorly understood requirements and using scenarios to elicit requirements are carried out as standardized practices; this difference occurs because software is already developed and might not offer a perfect fit with user requirements.

The practice of reusing requirements (RE13) is an advanced practice, a practice used to improve a system, whereas in PSI, it has a completely standardized use.

Table 3. Requirements elicitation in PSI.

<table>
<thead>
<tr>
<th>No</th>
<th>RE Practices</th>
<th>Type</th>
<th>PSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE1</td>
<td>Assess system feasibility</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE2</td>
<td>Be sensitive to organisational and political consideration</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE3</td>
<td>Identity and consult system stakeholders</td>
<td>Basic</td>
<td>Common use</td>
</tr>
<tr>
<td>RE4</td>
<td>Record requirements sources</td>
<td>Basic</td>
<td>Common use</td>
</tr>
<tr>
<td>RE5</td>
<td>Define the system’s operating environment</td>
<td>Basic</td>
<td>Common use</td>
</tr>
<tr>
<td>RE6</td>
<td>Use business concerns to drive requirements elicitation</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE7</td>
<td>Look for domain constraints</td>
<td>Intermediate</td>
<td>Discretionary use</td>
</tr>
<tr>
<td>RE8</td>
<td>Record requirements rationale</td>
<td>Intermediate</td>
<td>Common use</td>
</tr>
<tr>
<td>RE9</td>
<td>Collect requirements from multiple viewpoints</td>
<td>Intermediate</td>
<td>Discretionary use</td>
</tr>
<tr>
<td>RE10</td>
<td>Prototype poorly understood requirements</td>
<td>Intermediate</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE11</td>
<td>Use scenarios to elicit requirements</td>
<td>Intermediate</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE12</td>
<td>Define operational processes</td>
<td>Intermediate</td>
<td>Discretionary use</td>
</tr>
<tr>
<td>RE13</td>
<td>Reuse requirements</td>
<td>Advanced</td>
<td>Standardised use</td>
</tr>
</tbody>
</table>

Table 4. New Requirements elicitation Practices - PSI.

<table>
<thead>
<tr>
<th>New Requirements Elicitation Practices</th>
<th>RE Practices</th>
<th>Type</th>
<th>PSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE14</td>
<td>Use live software demonstration to elicit users’ needs</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
<tr>
<td>RE15</td>
<td>Use a user manual</td>
<td>Basic</td>
<td>Standardised use</td>
</tr>
</tbody>
</table>

Looking for domain constraints (RE7), collecting requirements from multiple viewpoints (RE9), and defining operational processes (RE12) are practices that are only carried out with discretionary use in PSI.

We also identified some new requirements elicitation practices used during PSI. The new practices, listed in Table 4, are using a live software demonstration to elicit the users’ needs (RE14), and using a user manual (RE15). According to Sommerville and Sawyer [20] and Cox et al. [5], these practices are carried out at the ‘basic’ level in general RE. Our findings showed them to have ‘standardized’ use in PSI. It is interesting that the practice of holding live software demonstrations (RE14) is repeatedly used in PSI. The reason for frequent use of the live software demonstration in PSI is that it educates users about the software’s functionalities, helps to increase users’ participation in discussions, and helps analysts to discover and discuss user needs and any misalignments between these needs and what the software product or system can do [18].

In addition, a live software demonstration may help to convince users that there are alternative solutions to any misalignments that are identified [1]: the live software demonstration can be used to show work-around [12]. In packaged software implementation, work-around are used with the intention of minimizing customization, not in order to reduce conflicts between requirements. In the software organization we noticed, ‘work-around’ were used when the analysts tried to convince clients to use the software as it already was, rather than ask for a full-scale customization. This could be deemed a ‘work-around’ because the client does end up getting the function or result they want, just not in the particular order they desired, while the software organization avoids having to customize a function. But this kind of work-around is only possible if the client’s customization request responds to what is only a ‘perceived’ misalignment rather than a ‘real’ one. If a required function is totally missing from the software or can’t somehow be supplied by the existing software, then the function has to be built.
It was found that using a user manual (RE15) is a standard practice in PSI. The purpose of user manuals in PSI is to educate users about the software’s functionalities so that users will have initial knowledge of the software even before they use it.

### 4.3. System Modeling Practices

From Table 5 we can observe that developing complementary system models (SM1), modeling the system’s environment (SM2), and modeling the system’s architecture (SM3) are practices used at the basic level in RE, but that they are never used within PSI. It is also interesting to note that a basic system model in PSI would be similar to a prototype method with natural language description Beecham et al. [3].

However, formal system modeling such as context diagram, DFD, use cases, and so on, have not been used as requirements modeling methods in PSI. This may be because PSI analysts focus on customization requests and their impact on software functions. Rather than collect all customization requests at one time and work on them all at the same time, analysts using PSI collect new requests as they come in, possibly on a daily basis. As a result of this approach, they do not engage in system modeling.

System modeling is a process activity that relates to the building of abstract system models that aid in the understanding and analysis of requirements and of understanding their implications for the proposed system. System modeling may also follow various guidelines and can be carried out at basic, intermediate, or advanced levels.

Table 5. System modelling in PSI.

<table>
<thead>
<tr>
<th>No</th>
<th>System Modelling Practices</th>
<th>RE Practices</th>
<th>Type</th>
<th>PSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM1</td>
<td>Develop complementary system models</td>
<td>Basic</td>
<td>Never used</td>
<td></td>
</tr>
<tr>
<td>SM2</td>
<td>Model the system’s environment</td>
<td>Basic</td>
<td>Never used</td>
<td></td>
</tr>
<tr>
<td>SM3</td>
<td>Model the system architecture</td>
<td>Basic</td>
<td>Never used</td>
<td></td>
</tr>
<tr>
<td>SM4</td>
<td>Use structured methods for system modelling</td>
<td>Intermediate</td>
<td>Discretionary use</td>
<td></td>
</tr>
<tr>
<td>SM5</td>
<td>Use a data dictionary</td>
<td>Intermediate</td>
<td>Common use</td>
<td></td>
</tr>
<tr>
<td>SM6</td>
<td>Document the links between stakeholder requirements and system models</td>
<td>Intermediate</td>
<td>Standardised use</td>
<td></td>
</tr>
</tbody>
</table>

Other forms of system modeling practices are practiced at varying levels within PSI. Documenting the links between stakeholder requirements and system models (SM6) is a practice that has standardized use in PSI, but using a data dictionary (SM5) and using structured methods for system modeling (SM4) are not standardized practices in PSI. Rather, the data dictionary receives common use (SM5) and the structured methods (SM4) receive discretionary use. However, the way that analysts document the links between stakeholder requirements and system models (SM6) in PSI is by using a Users’ needs/Misalignments Specification document in which the relationships between users’ needs and other software functions are specified. The document also specifies the relationship between users’ needs and data stores.

### 4.4. Requirements Validation Practices

Requirements validation can be defined as consisting of practices that make up formal validation procedures that help analysts to check for problems related to incomplete requirements, inconsistent requirements, or incompatibility between systems or between an organization and a new system. Requirements validation practices are also established to ensure that requirements can be verifiable and to help set quality standards.

As seen in Table 6, in RE, there are four ‘basic’ practices involved with requirements validation, which are checking the standard of the requirements document (RV1), organizing formal requirements inspections (RV2), using multi-disciplinary teams when reviewing requirements (RV3), and defining validation checklists (RV4). Within PSI, however, these practices receive varying levels of practice.

While organizing formal requirements inspections (RV2) and using multi-disciplinary teams when reviewing requirements (RV3) are both basic practices in RE, they are practices that have only discretionary use in PSI. This is because with packaged software there can be different sources of requirements for a computer-based system: for example, end-users of the system, managers in the organization and customers of the organization. All have their own viewpoint on the services that the system should provide. During a requirements review process these different viewpoints should be considered in order to reduce requirements errors. It may be that RV2 and RV3 should be standardized practices for large organizations where many systems are complex and difficult to quickly understand. In this study, however, RV2 and RV3 are not standardized practices, because most of the client organizations considered are SMEs in which the number of users is small. The analysts engaged in PSI instead used software demonstrations to target users and to collect the misalignments at the same time as
validating the misalignments. Thus, answering the customization requests became part of their daily work and projects. Meanwhile, defining validation checklists (RV4) is a practice that may be widely used in PSI, but is not mandatory. Validation checklists concentrate on the requirements document as a whole and help those conducting validation to concentrate on important attributes of the requirements document. However, defining validation checklists was not considered a standard or essential practice by organizations engaging in PSI.

Three practices in RE validation are carried out at the intermediate level: using prototyping to animate requirements (RV5), using a draft user manual (RV6), and proposing requirements test cases (RV7). Within PSI, these three practices all receive standardized use. Within PSI, using prototyping to animate requirements (RV5) and proposing requirements test cases (RV7) involve software demonstration. One last practice, paraphrasing system models (RV8) is treated as an advanced practice in RE and has only discretionary use in PSI.

The way that the analysts showed their client the software in order to demonstrate that completed modifications worked successfully matches recommendations in the literature. It was recommended by Beecham et al. [3] that for each requirement one or more test cases should be proposed to identify any requirements errors. Requirements errors can have a wide impact on the success of software development projects. No software process can keep delivery times, costs and product quality under control if requirements errors are not identified and removed early on [3]. One solution recommended for managing uncertainty about requirements is prototyping. Other motivations for building a prototype are: eliciting requirements, validating requirements, and determining the feasibility of particular solutions [3]. However, in PSI, using prototyping and requirements test cases through software demonstration were also used as strategies for identifying ‘actual’ misalignments and ‘perceived’ misalignments. This is understandable because analysts support the idea of minimizing customization.

### 4.5. Requirements Management Practices

Requirements management is defined as a system of guidelines and activities used to manage requirements information throughout the project development life-cycle.

<table>
<thead>
<tr>
<th>Table 7. Requirements management in PSI.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No</strong></td>
</tr>
<tr>
<td>RM1</td>
</tr>
<tr>
<td>RM2</td>
</tr>
<tr>
<td>RM3</td>
</tr>
<tr>
<td>RM4</td>
</tr>
<tr>
<td>RM5</td>
</tr>
<tr>
<td>RM6</td>
</tr>
<tr>
<td>RM7</td>
</tr>
<tr>
<td>RM8</td>
</tr>
<tr>
<td>RM9</td>
</tr>
</tbody>
</table>

It can be seen from Table 7 that requirements management involves four basic practices: uniquely identifying each requirement (RM1), defining policies for requirements management inspections (RM2), defining traceability policies (RM3), and maintaining a traceability manual (RM4). However, three of these practices (RM1, RM3, and RM4) are not practiced at a standardized level in PSI. Instead, in PSI, identifying the requirements and defining the policies for requirements management inspections have common use, while defining the traceability policies and maintaining a traceability manual are practices that are not standard or normal, but rather, discretionary.

One of the practices that are intermediate in RE, using a database to manage requirements (RM5) is a standardized practice in PSI.

However, one practice that is intermediate in RE, identifying global system requirements (RM7), receives only discretionary use in the PSI process.

During the Focus Groups, we were told by analysts that ‘identifying global system requirements’ was not a high-priority practice because the analysts rarely had to do this. They were not usually considering global system requirements since they deal with local users.

Lastly, two practices can be considered advanced in RE: identifying volatile requirements (RM8) and recording rejected requirements (RM9). In PSI, the practice of identifying volatile requirements is completely discretionary, while the practice of recording rejected requirements is not an advanced practice, but rather a standard one that is always used.

The finding regarding the facts that analysts did always keep records of which requirements had been rejected was drawn primarily from information located in descriptive analysis forms.

In RE, analysts may also record rejected requirements (RM9). If this is done, it is practiced at the advanced level. However, in PSI, recording rejected requirements is actually treated as a core practice and has standardized use. Analysts carrying out PSI may have a greater need than analysts doing
RE to record rejected requirements. This is because analysts doing PSI may need to refer back to a list of requirements that other analysts have rejected in response to requests from other clients. This can tell the analyst which requirements were previously rejected because they would have a negative effect on the software. Alternatively, analysts may look at such a list in terms of considering possible benefits that could be gained if the rejected requirements were followed up on in future. For example, the rejected requirements could be kept in mind for inclusion in the next release requirements. There are some factors that may lead analysts to reject requirements, such as whether a requirement is ‘actual’ or ‘perceived’, the potential benefits of the requirement, the software scope, software price, and the client’s organization size.

5. Conclusions

This study presents an account of packaged software implementation requirements engineering with a focus on analysts’ practices within SPSVs. Through this study we offer extensive discussion of how SPSVs on analysts’ practices within SPSVs. Through this implementation requirements engineering with a focus

This practice involves ensuring there is a high degree of traceability and rationale for a requirement. When it comes to requirements validation, we found that validation checklists were not used as commonly in PSI. Requirements in PSI were often instead validated by using prototyping to animate requirements, using a draft user manual, or by proposing requirements test cases. Future work could be develop a requirements engineering model for PSI requirements engineering.

References


Issam Jebreen completed his Undergraduate degree in computer information system from Al-Albayt University, Jordan. Then in 2014, he received his PhD in software engineering from Auckland University of Technology (AUT), New Zealand. His research areas are packaged software implementation, Requirement Engineering, and software development methods. He had had a long and effective teaching career in different universities and countries. His role at AUT supervision/lecturing on the undergraduate level, leading a team for the delivery of software project, and updating resources for areas of the software development methodology, techniques, and methods. Beside his academic career he had been working at industry and research area, he had been working as Packaged Software Implementation consultant in Jordan as well as Research Assistant at University National Malaysia. Currently he is an assistant professor at Zarqa University.

Ahmad Alqerem obtaining a BSc in 1997 from JUST University and a Masters in computer science from Jordan University in 2002. PhD in mobile computing at Loughborough University, UK in 2008. He is interested in concurrency control for mobile computing environments, particularly transaction processing. He has published several papers in various areas of computer science and software engineering. After that he was appointed as associate professor at software engineering Depts. Zarqa University.