Journal of Universal Computer Science, vol. 25, no. 7 (2019), 764-787
submitted: 31/8/17, accepted: 13/5/19, appeared: 28/7/19 © J.UCS

Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations

RJ Macasaet
(Pentathlon Systems Resources Inc., Manila, Philippines
rjmacasaet@pentathlonsystems.com)

Manuel Noguera
(University of Granada, Spain
mnoguera@ugr.es)

María Luisa Rodríguez
(University of Granada, Spain
mlra@ugr.es)

José Luis Garrido
(University of Granada, Spain
jgarrido@ugr.es)

Sam Supakkul
(Sabre, Inc., Southlake, Texas, USA
ssupakkul@ieee.org)

Lawrence Chung
(University of Texas at Dallas, Richardson, Texas, USA
chung@utdallas.edu)

Abstract: Initializing software for a micro-business in a remote community in a developing nation is challenging, especially when gathering requirements. This paper proposes the use of Micro-business Requirements Patterns (μbRPs) in the initial phase of software implementation. The μbRPs aim to guide the software developer when gathering requirements from a micro-business and for estimating the effort needed to implement the software. First, we present the μbRPs, which include a table, optional illustrations, and associated software components. Then, we explain how μbRPs are applied in practice. Finally, we discuss how our proposal has evolved through the years by presenting our action research and inspirations from related work.

Keywords: Micro-business, Requirements Pattern, Small-to-medium sized enterprise, Nonfunctional requirement, Software Component, Action Research
Categories: D.2.1

1 Introduction

The domain of micro-businesses is filled with challenges. The mere characterization of a micro-business has already been a challenge in itself, already with several conflicting views [Merten et al., 11]. The varying characterizations of micro-businesses (and SMEs in general) range from being based on the number of employees [European Commission, 08] [International Organization for Standardization, 11], the age of the business [Nikula et al., 00], the length of software projects [Aranda et al., 07] [Aranda, 10], the degree of collaboration on software projects [Jantunen, 10], and the degree of adaptability in software projects [Kamsties et al., 98]. We characterize micro-businesses together with their software implementations. They are resource-constrained, as opposed to multinational organizations with huge budgets [Macasaet, 17]. Software implementations in a micro-business would not exceed 10 man days of effort (8 hours per man day) nor more than 10 software components as we roughly estimated [Macasaet et al., 12].

We characterize a software component as an encapsulation of a certain set of data and functions which vary in granularity as long as they could be updated, replaced, or modified without affecting other software components in a system. On a similar note, [Medvidovic and Taylor, 00] characterize components as defined units of computation or data which could be as small as a single procedure or as large as an entire application. Examples of such components would range from an off-the-shelf customer management system, a website template, or a simple JavaScript line of code.

Initializing software implementations for micro-businesses could be very challenging. Gathering requirements from micro-business owners and estimating the effort required to implement the software could easily be overlooked. Micro-businesses must take their software requirements seriously because sloppy requirements eventually turn into problems [Kauppinen et al., 04], eventually threatening the success of the implementation [Davis et al., 06]. Proper requirements are essential, no matter the size of the business [Young, 04]. In addition, the style of communication between software developers and micro-business owners poses challenges as well. Micro-business owners are normally not exposed to technical software languages nor do they have the extra time to learn technical software jargon [Kamsties et al., 98] [Kauppinen et al., 02]. They would comfortably use their natural language and illustrations to express their requirements [Macasaet et al., 11]. Hence, communicating micro-business software requirements must be done as intuitively and as comprehensible as possible [Kruchten, 03], with as little or no technical software terms [Young, 04]. Requirements of this kind could be referred to as “lightweight but effective” [Ambler, 02].

On the other hand, the developers working on micro-business software implementations still need requirements which have technical relevance. The technical details in requirements documents and the effort estimates are valuable to the developer. It is a challenge to propose a requirements approach which can strike a balance between comprehensibility and technical relevance in this domain.

2 Challenges in Remote Communities in Developing Nations

The Republic of the Philippines is a developing nation and has 820,795 registered micro-businesses as of 2016 [Philippine MSME Statistics, 16]. These registered micro-businesses have less than 10 headcount and approximately total asset values less than 60,000 United States Dollars. The Philippines is a tropical archipelago made up of 7,641 islands [Namria, 18]. Within Philippine territory is the Province of Palawan, which is composed of approximately 1,700 islands.

Pentathlon Systems Resources Incorporated (PSRI) [PSRI, 18] has been invited by the city mayor of Puerto Princesa, the capital city of Palawan, on June 19, 2018 to rise to the challenge of connecting the people of the entire province using hardware, software, telecommunications providers, and the internet. By being connected, we mean that every current and potential tax-paying person should have a device that can connect them to the internet and with this device, communicate with others in the province. Based on a 2015 census, there are 849,469 people in the province of Palawan. Not all of the 1,700 islands in the province are inhabited but the goal is to connect approximately 1 million people. The main stakeholders of this initiative are the local government office of Palawan and both current and potential tax payers of the province. The funding will come from both the public and private sector.

One of the several solutions proposed by PSRI is to distribute Android tablets with prepaid Subscriber Identity Module (SIM) cards to the people of Palawan. PSRI plans to distribute Chinese-manufactured Android tablets which could be acquired for less than 40 United States Dollars per unit and having sufficient hardware to run apps (Android Operating System 5.0 Lollipop, Quad-core 1.8 GHz processor, 1 GB RAM, 8 GB storage, and with a SIM card slot). The largest telecommunications providers in the Philippines (GLOBE and SMART) are offering prepaid SIMs with 1 GB of mobile data for less than 1 United States Dollar. The people in the province also have the option to use their existing tablet, SIM card, and network connection if available. Part of this solution is to customize the software on their tablet based on the kind of livelihood of the user. There are several livelihoods in the province and the most common are related to the plantation and harvesting of coconut, sugar, and rice. Fishing is also popular. People who have specialized in these kinds of livelihoods are normally not familiar with customized software and technology in general so a lot of user onboarding from the developer is expected.

The implementation plan of PSRI for this solution is to send the developers to the remote communities across the inhabited islands and start distributing the Android tablets physically, one by one. One or two developers from PSRI will be responsible for one or several of the 25 municipalities in the province. A total implementation team of 30 developers has been estimated at this point in time. Each developer will talk with the locals in their respective area(s) and depending on the livelihood of each individual, the developer will install, configure, and customize the software on the tablet based on the needs of their livelihood (e.g. requirements of their micro-business). The assumption of this solution is that the entire province will become more connected as the people find it more and more useful when they use (the software on) their tablets. A critical step in this solution is to find a way for the developers to properly specify the software requirements for the people of Palawan so that when the software is implemented, their tablets become as useful as possible. In the next section, we use a coconut planter micro-business as an example.

3 Micro-business Requirements Patterns (µbRPs)

Our proposal for properly gathering software requirements is to use Micro-business Requirements Patterns, referred to as µbRPs in this paper. This section presents µbRPs: its tables, illustrations, and associated software components. We show how we create, store, and apply µbRPs as well. A µbRP is made up of (1) a table that guides the developer when asking software requirements-related questions to the micro-business owner, (2) optional illustrations that help comprehension for both developers and micro-business owners, and (3) associated software components that could be used for implementation.

3.1 The µbRP Table

We show a simplified µbRP table on Table 1 for presentation purposes. It has the name of the µbRP, its description, and keywords on the top-most part of the table. From top to bottom, we have the questions that could be asked to the micro-business owners, starting from questions that are more functional on the top and then questions that are more non-functional at the bottom. From left to right, we have the steps that are taken by the developer, starting from requirements gathering from the left and ending with acceptance of the software implementation by the micro-business owner.

In step 1, the questions, response types, and possible choices are prepared beforehand by developers. The µbRP in Table 1 is a result of observations of how ten different coconut planters in the province of Palawan conduct their day-to-day operations. The questions on the table are as down-to-earth as possible, meaning that any layman or everyday business owner must be able to understand it. These questions have no technical jargon since we assume that majority of micro-business owners, especially in remote communities, have no technical background at all.

In step 2, the responses of the micro-business owners are recorded. The response types in step 1 and the responses in step 2 correspond to the "modes" and "instantiations" of the µbRP, respectively. The letters in parentheses per question also have correspondences with encircled letters in the illustration in Figure 1. The modes, instantiations, and illustrations are explained later on in this paper.

In step 3, the estimations for the effort needed to satisfy the functional requirements are expressed in man days and are made by the estimators from the software development company. An estimator could be anyone qualified to make an estimate such as a software developer, development manager, or analyst. The effort is also confirmed by the developer who will be assigned to complete the task during actual development. Steps 1 to 3 are the steps which take place when initializing a software project and these are the steps which we will focus on in this paper.

In step 4, the developer confirms when their work per requirement has been delivered and the testers, who are ideally not the developers themselves, also test that the work done by the developer has satisfied the requirements. In the case of remote communities in developing nations, it may be necessary, although not optimal, for the implementing developer to do the testing. Finally, the developer presents and onboards the micro-business owner with the software and then the micro-business owner either accepts or rejects if a requirement has been met or not in step 5.

The bottom section of the µbRP table pertains to non-functional requirements (NFRs) [Chung et al., 2000]. The columns in this section are grouped in chronological steps as well. Step 1 is made up of a list of NFRs that could be of importance to the stakeholders of the software implementation, e.g. the micro-business owner, the software developers, and customers of the micro-business. This list of NFRs is prepared by the software developers beforehand.

µbRP – Coconut Planter							
Textual Description: µb plants and harvests coconuts, buyers pay and pick up coconuts from planters							
Keywords: Coconut, Planter, Buyer Pick-up							
Functional Requirements in Q&A form for the µb Owner **(Links to Figure 1)**							
Step 1:		2:	3:		4:		5:
done by Developers		with µb	by devs		by devs and/or testers		with µb
Question	Modes (Possible Choices)	Instantiation (Answers)	Estimated dev days	Confirmed	Delivered	Tested	Accepted
(a) When do you want your coconuts to be picked up by the buyer's truck?	Everyday / Certain Days / On Call	Tuesday, Friday	1	√	√	√	√
(b) How do I prefer to be paid by the buyer?	Cash / Bank Transfer / Digital	Cash	1	√	√	√	√
More ***questions*** *follow… (table is shortened for presentation purposes)*							
Non- Functional Requirements in Terms of Priority **(Links to Figure 2)**							
How important is ___ to the µb Owner?		Rank					
(c) Timely pick-up of coconuts		1st	2	√	√	√	√
Timely payment		2nd	2	√	√	√	√
More ***priorities*** *follow… (table is shortened for presentation purposes)*							

Table 1: A Micro-business Requirements Pattern: Coconut Planter

In step 2, the stakeholders of the software project rank the NFRs in terms of priorities (where only the top NFR (priority) is diagrammed in Figure 2 for simplified presentation purposes, as will be shown later). Approaches such as the Quantitative WinWin [Ruhe et al., 03] may be used to aid stakeholders in establishing priorities. Steps 3 to 5 in the NFR section follow the same logic as the functional requirements. As aforementioned, in this paper, we will be focusing on steps 1 to 3 since these are the steps which take place when initializing software implementations.

3.2 Optional Illustrations for µbRPs

When tight on resources such as time and budget, software implementations may be initialized solely with the guidance of a µbRP table. However, we have included optional illustrations for µbRPs in our proposal because extra comprehension of the implementation at hand could be helpful at times, both for the developer and micro-business owner. This subsection explains the kinds of supporting illustrations for μbRPs. For presentation purposes, all illustrations in this paper are simplified versions of the actual ones in practice. The Coconut Planter μbRP table in Table 1 will be used as reference for illustrations in this subsection.

μbRP illustrations always have a legend within the illustration so that it is comprehensible to any layman without a technical background. Compared to our previous proposal [Macasaet et al. 2013] [Macasaet et al., 2014], we assume that this simpler illustration approach will be more comprehensible to the micro-business owners in the remote communities in Palawan. μbRP illustrations can be created manually by hand on paper (scanned or photographed later on for archiving) resulting in sketches [Macasaet et al., 11] or with any illustration tool such as RE-Tools [RE-Tools, 14] [Supakkul and Chung, 12] [Supakkul et al., 13].

First, we explain Figure 1, where the modes of the μbRP are illustrated. Modes are the possible choices and response types as shown in Table 1. The modes are *possible* manners or ways in which a micro-business owner conducts his or her day-to-day operations. The encircled letters in Figure 1 correspond to the letters in parentheses in Table 1. [Mendling et al., 10] recommend purposeful placement of labels in order to make illustrations more relevant. Hence, we place a μ symbol where there are modes in the μbRP illustration.

Second, we explain Figure 2, where an instantiation of the μbRP is illustrated based on the responses of the micro-business owner in the μbRP table. An instance is the manner or way in which a micro-business owner *actually* conducts his or her day-to-day operations. The chosen priorities of the micro-business owner in Table 1, indicated with letters in parentheses, correspond to encircled letters in Figure 2. We place an exclamation mark ! to illustrate the NFR and its refinements and an upside down exclamation mark ¡ to illustrate when a (refined) NFR could be directly related to how a software component functions. We place a </> symbol to illustrate the associated software components that could meet the requirements from the μbRP table.

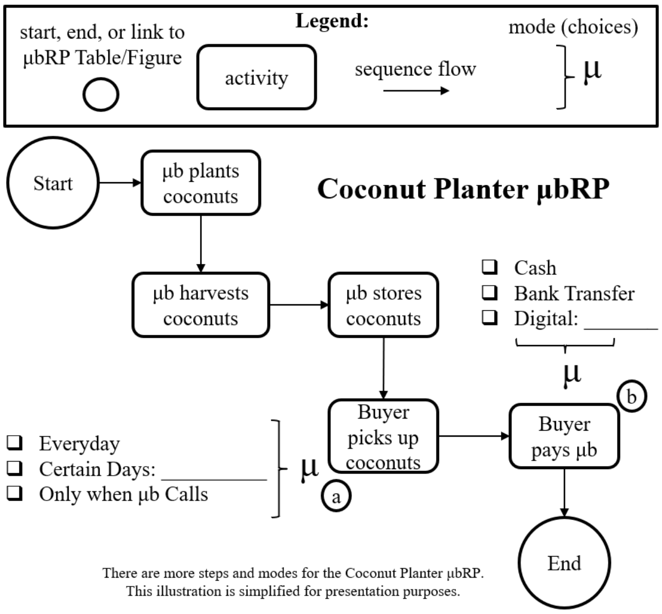

Figure 1: An illustration showing the modes of the Coconut Planter µbRP

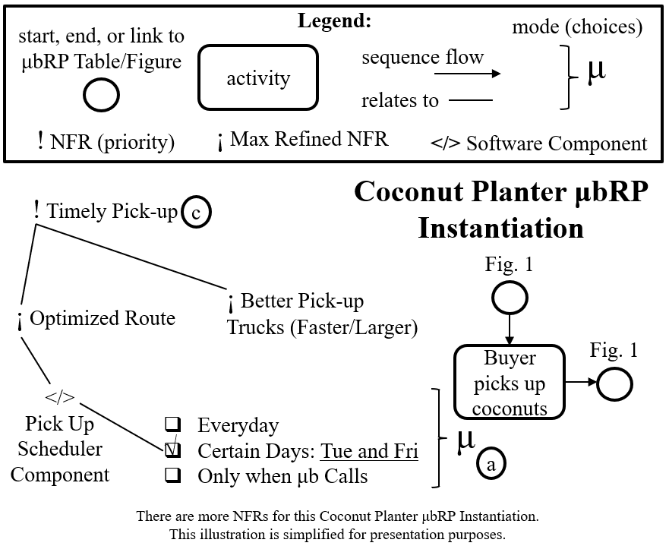

Figure 2: An illustration showing an instantiation of the Coconut Planter µbRP

Third, we explain Figure 3, which is based on the software components illustrated in Figure 2. Figure 3 illustrates the associated software components of the µbRP, including details on input-output relationships of one software component to another. This illustration is meant to guide the software developer when searching for reusable associated software components in component libraries.

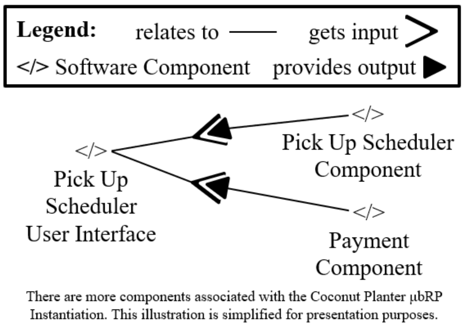

Figure 3: Associated software components of the Coconut Planter µbRP

3.3 General Purpose Illustrations for µbRPs

There are two other general-purpose illustrations which aid in understanding µbRPs. They are the µbRP meta-model and the step-by-step process for using µbRPs in practice. In Figure 4, we present the µbRP meta-model showing an µbRP having modes which have instantiations. µbRPs are derived from common, recurring requirements in actual micro-businesses. The µbRPs are realized by software components which either satisfy functional requirements (questions) or satisfice (satisfy sufficiently) [Chung et al., 2000] NFRs (priorities).

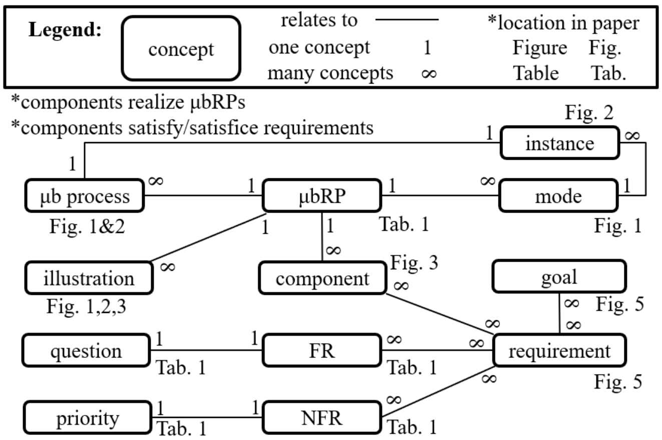

Figure 4: µbRP metamodel

The next general-purpose diagram which aids in the understanding of µbRPs is the step-by-step process for using µbRPs in practice as shown in Figure 5. Step 1 is the observation of actual micro-businesses. Business goals and requirements are recorded. There are some techniques which transform business goals into requirements such as the one on page 198 of [Kotonya and Sommerville, 03] and another one in the goal modelling section of [Respect-IT, 07]. Should the reverse be needed, transforming requirements to goals, [Cardoso et al., 11] provide such techniques as well.

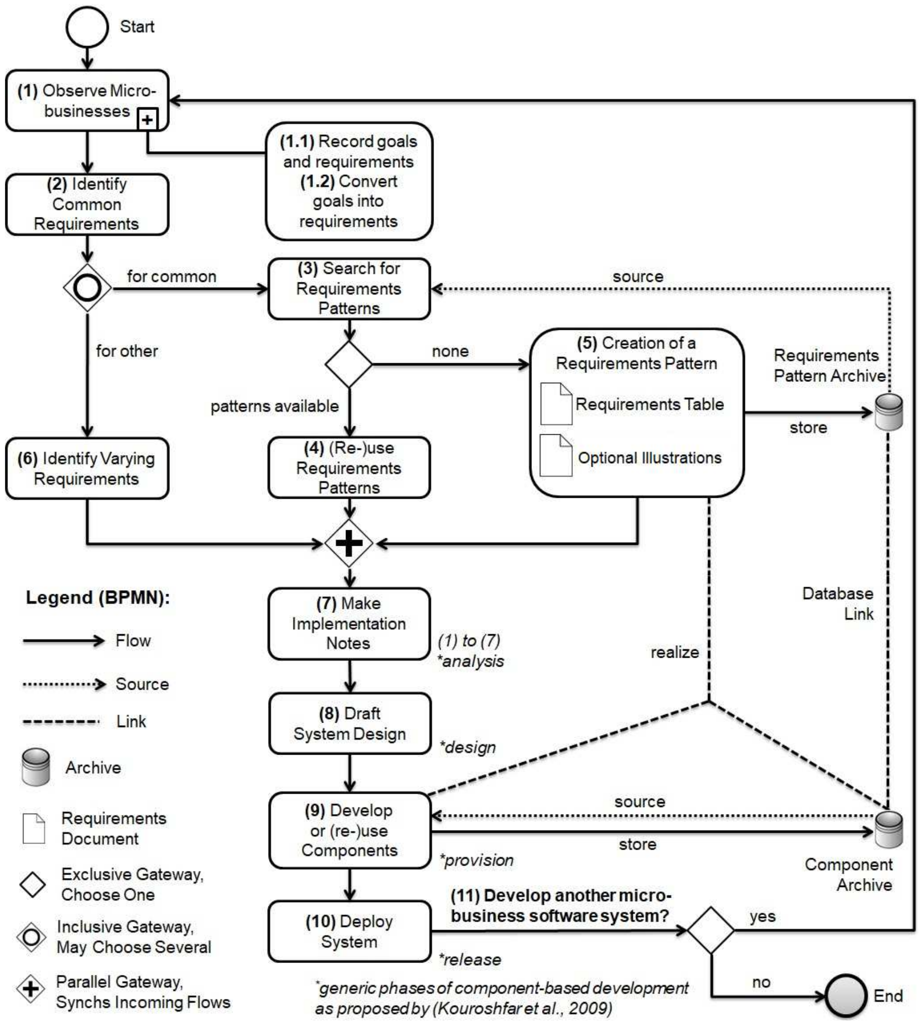

Figure 5: A step-by-step process for using μbRPs in practice

Step 2 involves identifying common requirements. When common requirements are identified, software developers search for requirements patterns in an archive in step 3. If there are available patterns then in step 4, the requirements patterns are re-used. If there are no available patterns then in step 5, a requirements pattern with the common requirements is created, consisting of the mandatory requirements table and optional illustrations. This requirements pattern is stored in an archive. In step 6, the varying requirements of a micro-business are identified.

In step 7, all the relevant information gathered from steps 1 to 6 is collated and used to make implementation notes. From a component-based software development context, step 7 could be commonly referred to as the analysis phase [Kouroshfar et al., 09]. In step 8, the architecture and design of the system is drafted which leads to step 9 where reusable software components and other reusable assets are identified for use in development. After development, the software system is implemented in step 10. Steps 8, 9, and 10 correspond to the design, provision, and release phases, respectively, in a component-based software development context [Kouroshfar et al., 09]. Should the software developer decide to implement another software system then step 11 simply loops back to step 1. Otherwise, step 11 ends the entire process.

4 Action Research

Our research team has always placed value in documenting observations of actual software implementations around the world. Action research has always been an ideal choice for us because when documenting observations, we are able to gather actual data such as the effort exerted in software implementations, the number of software components reused, and given the daily rates of developers, even the costs involved. This actual data helps us understand what is going on when μbRPs are applied in practice. In addition, the use of μbRPs requires training and influence from the researchers. In case studies and field experiments (as opposed to action research), the influence of the researcher has to be minimized or even non-existent. Given the several variants of action research [Goldkuhl, 08] [Goldkuhl, 12] [Bilandzic and Venable, 11], the action research in this paper is reported as-is, step-by-step, for clarification purposes. In order to improve the validity of action research, [Kock, 04] recommends the use of one or more of the following: units of analysis, multiple iterations, and/or grounded theory. We apply all three recommendations.

4.1 Units of Analysis

The advantage of using units of analysis is that when more instances of the unit of analysis are made, the more likely that statistical analysis can be used later on to ascertain whether there are observable trends or whether events are simply happening by chance. The first unit of analysis is the “effort exerted during software implementation,” expressed in man days. It starts on the first day of requirements gathering and ends on the day the software is accepted by the micro-business owner. The second unit of analysis is the "number of reused software components (as characterized in the introduction).”

4.2 Multiple Iterations

In order to perform iterations in action research, we needed the participation of companies which had micro-business software implementations taking place. We chose Manila (Philippines) and Granada (Spain) as cities for our action research because of the geographic proximity to our research teams and because of their abundance of micro-businesses. As of 2011, Manila had 211,974 micro, small, and medium enterprises (MSMEs) of which approximately 90% are micro, based on a headcount of less than 10 and approximately total assets less than 60,000 United States Dollars [Philippine MSME Statistics, 11]. Granada had 55,578 micro-businesses which comprise approximately 96% of total businesses, based on a headcount of less than 10 [Spain SME Statistics, 11].

Four software development companies with micro-business projects decided to learn and adapt the μbRPs – (a) Pentathlon Systems Resources Incorporated [PSRI, 18], (b) Virus Worldwide [Virus, 17], (c) Everyware Technologies [Everyware, 17], and (d) Desarrollo TIC [Desarrollo TIC, 17]. The first two are headquartered in Manila and the latter two are headquartered in Granada.

The companies were asked to identify a "previous implementation" and "implementations under study," where the latter would be the iterations in the action research. The implementations had to be of as similar nature as possible. In the "previous implementation," no μbRPs were used. It is important to note that the "previous implementation" is not a control implementation because if it were, then it would no longer be considered action research but a field experiment [Kock, 04].

Before the "implementations under study" took place, mandatory face-to-face training sessions involving two developers from each software development company were required. Afterwards, the developers could use the training material [PSRI Action Research, 14], supporting tools [RE-Tools, 14] and accompanying documentation [Supakkul and Chung, 12] [Supakkul et al., 13], and could contact the researchers anytime via email, Skype, or mobile for any μbRP-related support.

Since PSRI has contributed in developing the μbRPs under study since 2010, a fresh perspective involving two new hires developed a sales management system from scratch, their "previous implementation," without using μbRPs. Table 2 shows the units of analysis and six iterations which were performed in each company throughout a 30-month period, spanning from January 2015 to June 2017.

In addition, Table 2 also shows the exerted effort needed to set-up and maintain the μbRPs for use in each of the participating companies. This is measured from the time when there had been no μbRP knowledge up to the time when there were at least 10 μbRPs that could be used. Set-up efforts consist of (1) training the software developers and analysts regarding the creation and the use of μbRPs and (2) setting up the component libraries, μbRP tables, and optional μbRP illustrations.

If we assume that a man day costs US$ 320, then the average cost of setting up μbRPs would be US$ 1,760 (5.5 man day average setup time for the 4 sample companies in Table 2 multiplied by the assumed day rate) and US$ 320 for monthly maintenance. Table 2 also shows assumed savings based on the day rate multiplied by the reduction of exerted effort when μbRPs are used.

Observing µbRPs in Practice using Action Research																
Legend: **%** is % of reused components, **md** is man days exerted																
Company	Setup	**No µbRPs**		**µbRPs in Implementation**												reduction
Iteration		0		1		2		3		4		5		6		
	md	%	md	%	md	%	md	%	md	%	md	%	md	%	md	md
a	5	0	18	20	10	30	8	30	8	30	8	30	8	30	8	9
b	6	20	34	50	26	50	24	50	22	50	24	50	24	50	20	10
c	5	10	14	20	10	20	10	30	8	30	8	30	10	30	10	5
d	6	10	18	30	12	40	10	40	10	40	10	40	10	40	10	8

Table 2: Action research data from January 2015 to June 2017

4.3 Grounded Theory

In order to put the data in Table 2 into context, we use grounded theory as recommended by [Kock, 04]. Grounded theory traces its roots back from sociologists [Glaser and Strauss, 67]. There are three basic "types" of grounded theory: the original by [Glaser and Strauss, 67] also referred to as "classical Glaserian grounded theory," a formalized and procedural one by [Strauss and Corbin, 90], and one which clarifies ontological and epistemological ambiguities [Charmaz, 06]. We adapt the "classical Glaserian grounded theory" which has been recommended, used, and applied recently in the field of software engineering [Kock, 04] [Carver, 06] [Crabtree et al., 09] [Adolph et al., 11]. All references to grounded theory in this paper pertain to "classical Glaserian grounded theory."

Grounded theory is basically setting out to gather data and then systematically developing a substantive theory directly from the data [Glaser and Strauss, 67]. The theory is "grounded" in the data. Grounded theory differs from (other) methods which first develop theories without data and then systematically seek out data to verify the theories. Grounded theory is also different because its main purpose is not to find out irrefutable truths but to try to find out "what is going on here?"

[Schreiber and Stern, 01] suggest using grounded theory in research areas that have not been previously studied or where new perspectives are needed. In grounded theory, researchers go through the iterative steps of collecting data (where research notes are referred to "memos"), building theories, and then comparing the theories to those in existing literature. Grounded theory suggests that making comparisons to existing literature has to be delayed as much as possible so as to avoid coming up with preconceived theories which would not be grounded on the data.

On a practical level, the grounded theory in this paper is adapted for the specific needs of action research as recommended by [Kock, 04]. The data from Table 2 may be used to develop causal models [Bagozzi, 80] [Davis, 85] which are considered as the highest level of abstraction in grounded theory. The causal models link independent, moderating, intervening, and dependent variables [Arnold, 82] [Baron and Kenny, 86] [Creswell, 94] [Drew and Hardman, 85]. The variables may be classified in terms of units of analysis, which can be measured or estimated numerically or non-numerically [Drew and Hardman, 85] [Gregory and Ward, 74]

[Pervan and Klass, 92]. From the data in Table 2, we build a causal model, shown in Figure 6, which includes concepts, units of analysis, and the grounded theories we could build from the data.

Table 2 shows that the use of the µbRPs in the "implementations under study" could have reduced the number of man days expended in implementations and could have increased the number of software components reused in implementations, resulting in reduced effort for the participating software developers. Instead of drawing conclusions from the data and making claims, we build grounded theories and ask questions which would be relevant for the further evaluation and evolution of µbRPs. The result of several action research iterations in the past and grounded theories have led us to our current µbRP proposal as previously presented in this paper.

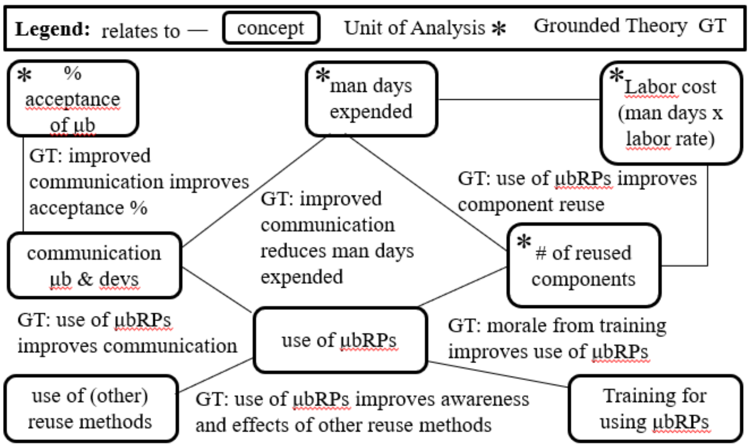

Figure 6: Causal Model

Some of the grounded theories that we can build from the data (and from memos) in Table 2 include (but are not limited to): the use of µbRPs improve component reuse, the use of µbRPs improve communication between micro-business owners and software developers, improved communication improves implementation acceptance rates, improved communication reduces the exerted effort in implementations, training for using µbRPs improves morale and the use of µbRPs, the use of µbRPs improves awareness and effects of other reuse methods.

Such grounded theories provoke questions such as: Was the improvement in software component reuse due to the use of µbRPs alone or due to a symbiosis between the use of µbRPs and (unknown, other) reuse methods? Did the morale of the software developers influence the quality of communication with micro-business owners? As a result of using the µbRPs, how many reduced man days can be attributed to improved communication? When evaluating the use of µbRPs in practice, is it possible to isolate human variables, which is indispensable in the field of software engineering? We extend the discussion of these emerging questions in the future work section of our paper.

5 Review of Related Work

Using a Venn Diagram as shown in Figure 7, we illustrate the related works and how the μbRPs have become the common point for all these works.

The first area of work are requirements patterns. (Micro-business) requirements patterns capture solutions to recurring (software) requirements challenges. They are presented in a form that can be understood by practitioners so that they can identify similar requirements (in their systems), select patterns that address those requirements and instantiate solutions that embody those patterns [Dwyer et al., 99]. Requirements patterns represent encapsulated information and help make writing requirements quicker and easier by providing information "on a plate" instead of starting from scratch. When a pattern is used to help write a (software) requirement, its job is essentially done [Withall, 07a] [Withall, 07b].

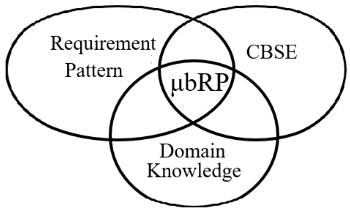

Figure 7: Venn Diagram of Related Works

The requirements in the μbRP table are written in such a way that any layman or non-technical person could understand them. In the case of remote communities in developing nations, it becomes very important to ask down-to-earth questions to a micro-business owner who barely knows software. The following requirements pattern proposals have been sources of ideas for us although they lacked the elements of associated software components and specializations in specific business domains such as remote communities in developing nations.

Software Requirements Patterns (SRPs) and Security Requirements Patterns are examples of requirements patterns that were lacking associated software components and specific domain elements needed to advance our work. [Franch et al., 10] propose 29 SRPs which aim to be of use during requirements elicitation, documentation, and validation. There has been a great percentage of reuse of NFR information in SRPs in call-for-tender requirement specifications. [Mendez-Bonilla et al., 08] propose SRPs based on a selection of published literature, mainly pertaining to functional and non-functional requirements. Here, the SRPs are used for commercial off-the-shelf software (COTS). [Hoffman et al., 12] propose 20 SRPs based on user trust. The SRPs are based on studies from the behavioural sciences, collecting antecedents that build trust. The SRPs are used mainly in recommender system development projects.

[Riaz and Williams, 12] propose security requirements patterns which aim to capture common security requirements, document the context in which a requirement manifests itself, and describe the trade-offs involved. Part of the proposal includes an outline for developing patterns and strategies for specifying reusable security requirements. In a related work, security test patterns are proposed by [Smith and Williams, 12] in order to aid in black box security testing.

The optional μbRP illustrations in our proposal are inspired by NFR visualizations. Concepts such as the operationalizing methods from Softgoal Interdependency Graphs (SIGs) [Chung et al., 2000] are rooted in NFRs. Four NFR patterns have been proposed by [Supakkul et al., 10], namely objective, problem, alternative, and selection. The purpose of the NFR patterns is to capture and reuse them in business-specific cases with the help of NFR visualizations and representations. NFR visualizations in business-specific, cloud-computing cases are also proposed in a related work by [Chung et al., 13]. The optional μbRP illustrations are also inspired by concepts such as (1) business-friendly comprehension of flows from the Business Process Modelling Notation (BPMN) and (2) component deployment diagrams from the Unified Modelling Language (UML).

We have used RE Tools which could simultaneously illustrate BPMN, SIGs, and UML as explained and demoed in our previous work [Supakkul and Chung, 12] [Supakkul et al., 13] [Macasaet et al., 14] although for our proposal for remote communities in developing nations, we believe that more lightweight illustrations are appropriate, as far as considering hand-drawn illustrations and sketches for better comprehension for its users [Macasaet et al., 11].

Domain knowledge captured in the form of business patterns is the second area of work related to our proposal. Using business patterns and finding out where the patterns can be instantiated in specific (domains and) contexts makes the work of software analysts and designers easier [Kilov and Sack, 09]. [Withall, 07a] [Withall, 07b], who have proposed 37 requirements patterns based on requirements that crop up in all kinds of systems, state that domain-specific requirement patterns (e.g. μbRPs for micro-businesses) could improve communication between product vendors (e.g. software developers) and their customers (e.g. micro-business owners).

[Aleksy and Stieger, 11] propose domain-specific patterns for mobile service businesses. Four mobile service business patterns are proposed, as derived from two industrial case studies, for the purpose of aiding in the integration of third-party partners, structuring communication between mobile workers and the back office, support for offline processing capabilities, and tailoring information support. [Seruca and Loucopoulos, 2003] propose a systematic way of capturing and reusing patterns based on their specific business domains. Their approach to pattern development is based on the analysis of domains and is process-oriented. This allows increased understanding of a business domain and the identification of opportunities to improve business processes. Specifically, their patterns have been applied in a business process improvement project in the clothing manufacturing domain.

[Barros, 07] proposes business process patterns (BPP) which result in business object (BO) frameworks, encapsulating high level business logic. The BPPs are reusable and can be applied to improve business processes or to develop a (software) application to support a business process. The resources, events, agents (REA) model has been extended by [Hruby, 06] with several structural and behavioural business patterns. The REA-based patterns are used to develop business-related (software) applications by searching for business objects and related modelling elements.

[Boukheduoma et al., 13] propose service-based cooperation patterns (SBCPs). The SBCPs are used for recurring service-based inter-organizational workflows (IOWF) that meet certain service-oriented architecture (SOA) paradigms, providing interoperability, reusability, and flexibility required when developing business-related (software) applications. [Glushko and Mcrath, 08] propose patterns based on documents, introducing the discipline of “document engineering.” Based on a set of analysis and design techniques, the document-centric patterns are created. The patterns are characterized as tangible and easy to analyse and are (re-)usable when designing business processes. [La Rosa et al., 17] propose configurable business patterns, which are based on domain knowledge of 430 municipalities in the Netherlands. The patterns can be instantiated in an actual business although the patterns still involve a lot of technical jargon and complex illustrations.

μbRPs are associated with Software Components, aiming to promote software reuse and component-based software engineering (CBSE), which is the third area of work related to our proposal. [Crnkovic et al., 02] propose the use of patterns (e.g. μbRPs) in CBSE, suggesting use in design, where reusable units are identified as pre-existing components and in development, where components are developed based on the design patterns. Associating software components with μbRPs has been inspired by some of the following CBSE proposals.

[Kouroshfar et al., 09] propose a generic process framework for component-based development. A set of high-level process patterns which are commonly encountered in seven component-based development methodologies are identified. The generic framework and the process patterns could be used in the development or customization of component-based systems. [Stepan and Lau, 12] propose controller patterns which are abstractions for defining coordination in the context of CBSE. The use of the controller patterns is demonstrated in a case study with the support of a prototype tool.

[Paludo et al., 11] propose a component specification structure based on analysis and design patterns. The purpose of the patterns is to document, retrieve, and capture composition functionalities of the components in order to achieve software reuse. The integration of the patterns and the components leverage the software reuse process through the creation of the documentation structure and a component repository capable of supporting software developers. [Kouskouras et al., 08] investigate the behaviour of component-based, object-oriented software applications when design patterns are not used, used, and when aspect-oriented programming techniques are used alongside the design patterns. The relationship between pattern use and software components is discussed in detail.

[Elizondo and Lau, 10] propose a catalogue of component connectors, describing the connectors as the “glue” which piece together components in CBSE. The purpose of the component connectors is to support the process of software development with the idea of reuse, alongside the use of design, architectural, and workflow patterns. In a related work by [Bhuta et al., 07], a framework for selecting the component connectors is proposed. Software Product Lines (SPLs) [Krueger and Clements, 17] propose the use of feature trees in SPLs using ontologies and abstractions. Although requirements could be abstracted and then later on instantiated based on actual cases, they fell short in domain knowledge and practicality when applied in remote communities in developing nations.

As part of this related work section, we would also like to discuss work that is not directly related to our µbRP proposal but may be of concern. For instance, µbRPs can be applied both in waterfall and in agile software development implementations. µbRPs can function, independent of the kind of project management or implementation methodology used. However, it is important to keep in mind that agile frameworks such as Scrum are only effective when there are 3 or more developers on the team [Scrum.org, 17]. Hence, Scrum may not be the best project management approach for using µbRPs where only one or two developers are involved in the implementation. Pair programming, an agile development method, could be applied but is not necessary. The same logic applies to the software architecture involved with µbRPs. The associated software components of the µbRP may function within a Representational State Transfer (REST) or Simple Object Access Protocol (SOAP) architecture, depending on the context of the implementation.

6 Observable Strengths and Weaknesses

This section presents observations on the use of µbRPs in practice, specifically from the experiences of those involved in the micro-business software projects of four software development companies, over a thirty-month period, from January 2015 to June 2017. Aside from the data we have provided in the action research section, we also conducted semi-annual one-on-one interviews of those involved in the implementations in order to get feedback. We have taken these transcribed interviews into account in enumerating the observable strengths and weaknesses of using µbRPs in practice.

The strengths we have observed are the following. First, the mandatory µbRP table during requirements elicitation was useful. The tables were very outright and straightforward which made it easily understandable for micro-business owners without technical backgrounds. In just one step, the micro-business owners simply had to answer questions in business terms without compromising technical details for the software developers. This saves a lot of time while maintaining quality requirements.

The µbRP table also contains a lot of domain knowledge which is useful for understanding the context of an implementation. We believe that our proposal of µbRPs for remote communities in developing nations could be useful for countries aside from the Philippines. There are also other tropical archipelagos that could benefit from our additions to the domain knowledge for remote communities in developing nations such as Indonesia, Papua New Guinea, and Cuba.

Second, the optional illustrations for µbRPs have been found useful for software component information, specifically for reuse and implementation. The use of labels indicates to software developers that there are opportunities for reusing software components in certain business process contexts and that the software developers should take advantage of these reuse opportunities if possible. The optional illustrations with associated software components also provide information regarding the relationships among the software components. Using the specific keywords and filenames found on these illustrations, the software developers are able to search the software component libraries with more guidance and more speed, knowing which associated software components to search for beforehand.

The optional illustrations indicate the activities in business processes and the associated software components which are critical and which directly or indirectly relate to NFRs. Prioritizing which business activities and which software components are critical allow software developers to focus their efforts on more important tasks, eventually contributing to the success rate for software implementations. The use of the optional illustrations provides both the software developers and micro-business owners a clearer overview of the software implementation, avoiding myopic views. Being aware of the many factors that affect the software implementation allow both software developers and micro-business owners to exert conscious efforts in areas critical to success.

Third, based on Grounded Theory, we observed that the overall length of implementations could have been shortened due to the use of μbRPs. When the internal communication within an implementation (e.g. the communication among software developers) and the external communication with the customer (e.g. the communication of software developers with the micro-business owner) are improved, the length of projects could be shortened. Improved communication and promoting software component reuse by using μbRPs could be related to shortening development time and eventually shortening total implementation time. Consequently, shorter implementation times could have translated to lesser man day effort and lower implementation costs, making software more affordable for the budget-conscious micro-businesses.

We also enumerate the weaknesses and limitations we observed applying μbRPs in practice. First, the companies which participated in evaluating the μbRPs were either involved in developing the μbRPs (PSRI) or have close working relationships with PSRI. It would be unavoidable to have biased opinions about the μbRPs. However, Grounded Theory places value on the opinion of the developers (of the μbRPs) [Glaser and Strauss, 67].

Second, given the strong ties, preferred treatment, and enthusiasm of the participating companies in the study, the implementation of μbRPs in other firms without these aforementioned factors could yield different results. It is not ascertainable if the μbRPs would continue to show positive effects. We have made the action research material available to the public [PSRI Action Research, 14] so that they could try and test the μbRPs themselves at their own convenience. We are also collecting feedback from anonymous μbRP users by constantly engaging in workshops, tutorials, and conferences.

7 Future Work and Conclusions

We start applying into practice our μbRP proposal for remote communities in developing nations on July 2018 at the province of Palawan. We continue to use our action research approach for observing and collecting data related to μbRPs. The software implementation at Palawan is expected to last for at least a year, and could extend to three to four years more, depending on the results of the first year and other circumstances. Our plan is to publish an action research report before 2020.

This paper has proposed µbRPs for remote communities in developing nations. The main purpose of the µbRPs is to guide software developers when gathering requirements and estimating effort at the initialization stage of a software implementation for remote communities in developing nations. This proposal is a result of (1) the observations gathered from several action research iterations from January 2015 to June 2017 and (2) our continuous review of related work and previous µbRP proposals. Hopefully, other software companies without strong ties to PSRI would start using µbRPs and share their results with the research community. Notwithstanding issues, our team believes that we have done the first studies on µbRPs, drawing from lessons learned in the past and also provoking relevant research questions to be answered in the future.

Acknowledgements

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness with European Regional Development Funds (FEDER) under the research project TIN2012-38600 and Pentathlon Systems Resources Incorporated (PSRI).

References

[Adolph et al., 11] Adolph, S., Hall, W. & Kruchten, P.: Using grounded theory to study the experience of software development, In Empirical Software Engineering, 16, 2011. pp. 487-513

[Aranda et al., 07] Aranda, J., Easterbrook, S. M., Wilson, G.: Requirements in the wild: How small companies do it, In Requirements Engineering RE 2007, pp. 39–48, IEEE

[Aranda, 10] Aranda, J.: Playing to the Strengths of Small Organizations, In Proceedings of the 1st Workshop on RE in Small Companies RESC 2010, pp. 141-144

[Aleksy and Stieger, 11] Aleksy, M., Stieger, B.: Mobile Service Business Patterns, In Proceedings of the IEEE 25th International Conference on Advanced Information Networking and Applications AINA 2011. pp. 62-68. IEEE

[Ambler, 02] Ambler, S.: Agile modeling. 2002, John Wiley and Sons

[Arnold, 82] Arnold, H.: Moderator variables: A clarification of conceptual, analytic, and psychometric issues, In Organizational Behavior and Human Performance, (29) 2, 1982. pp. 143-174

[Bagozzi, 80] Bagozzi, R. P.: Causal models in marketing. 1980. New York, NY: Wiley.

[Baron and Kenny, 86] Baron, R., Kenny, D.: The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations, In Journal of personality and social psychology, 51. 1986. pp. 1173—1182

[Barros, 07] Barros, O.: Business process patterns and frameworks: Reusing knowledge in process innovation, In Business Process Management Journal, 13 (1), 2007, pp. 47–69

[Bhuta et al., 07] Bhuta, J., Mattmann, C., Medvidovic, N., Boehm, B. W.: A Framework for the Assessment and Selection of Software Components and Connectors in COTS-Based Architectures, In Working IEEE/ IFIP Conference on Software Architecture WICSA 2007. IEEE Computer Society

[Bilandzic and Venable, 11] Bilandzic, M., Venable, J.: Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics, In Journal of Community Informatics, 7, 2011

[Boukheduoma et al., 13] Boukheduoma, S., Oussalah, M., Alimazighi, Z., Tamzalit, D.: Adaptation Patterns for Service-Based Inter-Organizational Workflows, In Wieringa R., Nurcan S., Rolland C., Cavarero J.L. (eds.), Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS 2013, IEEE

[Cardoso et al., 11] Cardoso, E., Almeida, J., Guizzardi, R., Guizzardi, G.: A Method for Eliciting Goals for Business Process Models Based on Non-Functional Requirements Catalogues, In International Journal of Information System Modeling and Design, 2(2), 2011. pp. 1–18

[Carver, 06] Carver, J.: The Use of Grounded Theory in Empirical Software Engineering, In V. R. Basili, H. D. Rombach, K. Schneider, B. A. Kitchenham, D. Pfahl, R. W. Selby (eds.), Empirical Software Engineering Issues. 2006. pp. 42. Springer

[Charmaz, 06] Charmaz, K.: Constructing grounded theory: a practical guide through qualitative analysis. 2006. London; Thousand Oaks, Calif.: Sage Publications

[Chung et al., 00] Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional Requirements in Software Engineering, 2000. Springer

[Chung et al., 13] Chung, L., Hill, T., Legunsen, O., Sun, Z., Dsouza, A., Supakkul, S.: A goal-oriented simulation approach for obtaining good private cloud-based system architectures, In Journal of Systems and Software, 86 (9), 2013. pp. 2242–2262

[Crabtree et al., 09] Crabtree, C. A., Seaman, C. B., Norcio, A. F.: Exploring language in software process elicitation: A grounded theory approach, In ESEM 2009. pp. 324-335

[Creswell, 94] Creswell, J. W. (ed.): A Qualitative Procedure in Research Design. Qualitative and Quantitative Approaches. 1994. London and New Dheli: Sage.

[Crnkovic et al., 02] Crnkovic, I., Larsson, M.: Challenges of component-based development, In Journal of Systems and Software, 61(3), 2002. pp. 201–212

[Davis, 85] Davis, J. A.: The Logic of Causal Order (Vol. 07-055). 1985. Beverly Hills, London, New Delhi. Sage.

[Davis et al., 06] Davis, C. J., Fuller, R. M., Tremblay, M. C., Berndt, D. J.: Communication Challenges in Requirements Elicitation and the Use of the Repertory Grid Technique, In Journal of Computer Information Systems, 46(5), 2006, pp. 78

[Desarrollo TIC, 17] Desarrollo TIC, 2017. https://desarrollotic.com

[Drew and Hardman, 85] Drew, C.J., Hardman, M.L.: Designing and Conducting Behavioral Research. 1985. Pergamon, New York, NY

[Dwyer et al., 99] Dwyer, M. B., Avrunin, G. S., Corbett, J. C.: Patterns in Property Specifications for Finite-State Verification, In Boehm B.W., Garlan D., Kramer J. (eds.), International Conference on Software Engineering ICSE 1999. pp. 411-420. ACM

[Elizondo and Lau, 10] Elizondo, P. V., Lau, K.-K.: A catalogue of component connectors to support development with reuse, In Journal of Systems and Software, 83(7), 2010. pp. 1165–1178

[European Commission, 08] European Commission.: The New SME Definition User Guide and Model Declaration, 2008

[Everyware, 17] Everyware Technology, 2017. http://www.everywaretech.es

[Franch et al., 10] Franch, X., Palomares, C., Quer, C., Renault, S., Lazzer, F. D.: A Metamodel for Software Requirement Patterns, In R. Wieringa, A. Persson (Eds.), Requirements Engineering for Software Quality REFSQ 2010. pp. 85–90. Springer

[Glaser and Strauss, 67] Glaser, B. G., Strauss, A. L.: The Discovery of Grounded Theory: Strategies for Qualitative Research, 1967. New York, NY: Aldine de Gruyter.

[Glushko and Mcrath, 08] Glushko, R., McGrath, T.: Document Engineering – Analyzing and Designing Documents for Business Informatics and Web Services. 2008. Cambridge, MA, USA: MIT Press

[Goldkuhl, 08] Goldkuhl, G.: Practical Inquiry as Action Research and Beyond, In W. Golden, T. Acton, K. Conboy, H. van der Heijden, V. K. Tuunainen (eds.), ECIS 2008. pp. 267-278

[Goldkuhl, 12] Goldkuhl G.: From action research to practice research, In Australasian Journal of Information Systems, 17, 2, 2012: pp. 57-78

[Gregory and Ward, 74] Gregory, D., Ward, H.: Statistics for Business Studies. 1974. McGraw-Hill, London, England

[Hoffman et al., 12] Hoffmann, A., Söllner, M., Hoffmann, H.: Twenty Software Requirement Patterns to Specify Recommender Systems that Users will Trust, In European Conference on Information Systems ECIS 2012

[Hruby, 06] Hruby, P.: Model-Driven Design Using Business Patterns. 2006. Secaucus, NJ, USA: Springer-Verlag New York, Incorporated

[International Organization for Standardization, 11] International Organization for Standardization (ISO).: ISO/IEC DTR 29110-1:2011 Software Engineering – Lifecycle Profiles for Very Small Entities (VSEs) – Part 1: Overview. 2011. ISO, Switzerland

[Jantunen, 10] Jantunen, S.: The Benefit of Being Small: Exploring Market-Driven Requirements Engineering Practices in Five Organizations, In Proceedings of the 1st Workshop on RE in Small Companies RESC 2010, pp. 131-140

[Kamsties et al., 98] Kamsties, E., Hormann, K., Schlich, M.: Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer, In Conference on European Industrial Requirements Engineering CEIRE 1998, London, United Kingdom

[Kauppinen et al., 02]. Kauppinen, M., Kujala, S., Aaltio, T., Lehtola, L.: Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice, In RE 2003, pp. 43–51, IEEE Computer Society

[Kauppinen et al., 04] Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S., Sulonen, R.: Implementing requirements engineering processes throughout organizations: Success factors and challenges, In Information and Software Technology, 46(14), 2004, pp. 937–953

[Kilov and Sack, 09] Kilov, H., Sack, I.: Mechanisms for communication between business and IT experts, In Computer Standards & Interfaces, 31(1), 2009, pp. 98–109

[Kock, 04] Kock, N.: The three threats of action research: a discussion of methodological antidotes in the context of an information systems study, In Decision Support Systems, 37 (2), 2004. pp. 265-286

[Kotonya and Sommerville, 03] Kotonya, G., Sommerville, I.: Requirements Engineering: Processes and Techniques. 2003. England: John Wiley and Sons Limited

[Kouroshfar et al., 09] Kouroshfar, E., Shahir, H. Y., Ramsin, R.: Process Patterns for Component-Based Software Development, In Lewis G.A., Poernomo I., Hofmeister C. (eds.), International Symposium on Component-based Software Engineering CBSE 2009. pp. 54-68, Springer.

[Kouskouras et al., 08] Kouskouras, K., Chatzigeorgiou, A., Stephanides, G.: Facilitating software extension with design patterns and Aspect-Oriented Programming, In Journal of Systems and Software 81 (October (10)), 2008. pp. 1725–1737, Elsevier

[Kruchten, 03] Kruchten, P.: The Rational Unified Process: An Introduction. 2003, Boston, MA: Addison-Wesley

[Krueger and Clements, 17] Krueger, C., Clements, P.: Enterprise Feature Ontology for Feature-based Product Line Engineering and Operations. In 27th International Symposium of International Council on Systems Engineering INCOSE, 2017

[La Rosa, 17] La Rosa, M., van der Aalst, W., Dumas, M., Milani, F: Business Process Variability Modeling: A Survey, In ACM Computing Survey, 50 (1), 2017

[Macasaet et al., 11] Macasaet, R., Chung, L., Garrido, J., Rodriguez, M., Noguera, M.: An Agile Requirements Elicitation Approach based on NFRs and Business Process Models for Micro-businesses, In Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement PROFES 2011, pp. 50-56. New York, NY, USA: ACM

[Macasaet et al., 12] Macasaet, R., Noguera, M., Rodriguez, M., Garrido, J., Supakkul, S., Chung, L.: Micro-business Behavior Patterns associated with Components in a Requirements Approach, In Proceedings of the 2nd International Workshop on Experiences and Empirical Studies in Software Modeling EESSMOD at the ACM/IEEE 15th International Conference on Model Driven Engineering Languages & Systems MODELS, 2012, New York, NY, USA

[Macasaet et al., 13] Macasaet, R.J., Noguera, M., Rodriguez, M.L., Garrido, J.L., Supakkul, S., Chung, L.: A requirements-based approach for representing micro-business patterns, In R. Wieringa, S. Nurcan, C. Rolland, J.L. Cavarero (eds.), Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS 2013, IEEE

[Macasaet et al., 14] Macasaet, R.J., Noguera, M., Rodriguez, M.L., Garrido, J.L., Supakkul, S., Chung, L.: Representing Micro-business Requirements Patterns associated with Software Components, In RCIS'13 Special Issue of Top Ranked Papers, Journal of Information System Modeling and Design IJISMD 2014. IGI-Global

[Macasaet, 17] Macasaet, R.J.: The Project Start Review Group, In M. Brambilla, T. Hildebrandt (eds.), Industrial Track Proceedings of the 15th International Conference on Business Process Management BPM 2017

[Mendez-Bonilla et al., 08] Mendez-Bonilla, O., Franch, X., Quer, C.: Requirements Patterns for COTS Systems, In Proceedings of the 7th International Conference on Composition-Based Software Systems ICCBSS 2008. pp. 232-234. IEEE

[Medvidovic and Taylor, 00] Medvidovic, N., Taylor, R. N.: A classification and comparison framework for software architecture description languages, In IEEE Transactions on Software Engineering, 26(1), 2000, pp. 70–93

[Mendling et al., 10] Mendling, J., Recker, J., Reijers, H.: On the usage of labels and icons in business process modelling, In International Journal of Information System Modeling and Design, 1(2), 2010. pp. 40–58

[Merten et al., 11] Merten, T., Lauenroth, K., Bürsner, S.: Towards a New Understanding of Small and Medium Sized Enterprises in Requirements Engineering Research, In Proceedings of the 17th International Working Conference on Requirements Engineering: Foundation for Software Quality REFSQ 2011, pp. 60-65. Springer Berlin Heidelberg

[Nikula et al., 00] Nikula, U., Sajeniemi, J., Kalvianen, H.: A state-of-the-practice survey on requirements engineering in small-and-medium-sized enterprises, In Telecom Business Research Center Lappeenranta Research Report 1, 2000, Lappeenrata University of Technology

[Namria, 18] National Mapping and Resource Information Authority of the Philippines, 2018

[Paludo et al., 11] Paludo, M., Reinehr, S. S., Malucelli, A., Bruzon, L., Pinho, P.: Applying pattern structures to document and reuse components in component based software engineering environments, In IEEE International Conference on Information Reuse and Integration IRI 2011, pp. 378-383, IEEE Systems, Man, and Cybernetics Society

[Pervan and Klass, 92] Pervan, G.P., Klass, D.J.: The use and misuse of statistical methods in information systems research, In R. Galliers (Ed.), Information Systems Research: Issues, Methods and Practical Guidelines, 1992. pp. 208–229, Blackwell, Boston, MA

[Philippine MSME Statistics, 11] Philippines MSME Statistics: Philippine Department of Trade and Industry. 2011

[Philippine MSME Statistics, 16] Philippines MSME Statistics: Philippine Department of Trade and Industry. 2016

[PSRI, 17] Pentathlon Systems Resources Incorporated, 2017. http://www.pentathlonsystems.com

[PSRI Action Research, 14] PSRI Action Research: Action Research Material for Micro-businesses, 2014. http://www.pentathlonsystems.com/ar4mb.html

[RE-Tools, 14] RE-Tools: RE-Tools, 2014. http://www.utdallas.edu/~supakkul/tools/RE-Tools/index.html

[Respect-IT, 07] Respect-IT.: KAOS Tutorial Version 1.0, 2007

[Riaz and Williams, 12] Riaz, M., Williams, L.: Security requirements patterns: understanding the science behind the art of pattern writing, In 2nd International Workshop on Requirements Patterns RePa 2012. pp. 29-34. IEEE

[Ruhe et al., 03] Ruhe, G., Eberlein, A., Pfahl, D.: Trade-off Analysis for Requirements Selection, In International Journal of Software Engineering and Knowledge Engineering, 13 (4), 2003, pp. 345-366

[Seruca and Loucopoulos, 2003] Seruca, I., Loucopoulos, P.: Towards a systematic approach to the capture of patterns within a business domain, In Journal of Systems and Software, 67(1), 2003, pp. 1–18

[Scrum.org, 17] Scrum Guide, 2017. https://www.scrum.org/resources/scrum-guide

[Schreiber and Stern, 01] Schreiber, R., Stern, P.: Using Grounded Theory in Nursing. 2001. Springer Publishing Company, New York.

[Smith and Williams, 12] Smith, B. H., Williams, L.: On the Effective Use of Security Test Patterns, In 6th IEEE International Conference on Software Security and Reliability SERE 2012. pp. 108-117. IEEE

[Spain SME Statistics, 11] Spain SME Statistics: Spanish Ministry of Industry, Energy, and Tourism. 2011

[Stepan and Lau, 12] Stepan, P., Lau, K.-K.: Controller patterns for component-based reactive control software systems, In V. Grassi, R. Mirandola, N. Medvidovic, M. Larsson (Eds.), Component-based Software Engineering CBSE 2012. pp. 71–76, ACM

[Strauss and Corbin, 90] Strauss, A., Corbin, J.: Basics of qualitative research: Grounded theory procedures and techniques. 1990. Sage Publications

[Supakkul et al., 10], Supakkul, S., Hill, T., Chung, L., Tun, T., Sampaio do Prado Leite, J.C.: An NFR Pattern Approach to Dealing with NFRs. In Proceedings of the 18th IEEE International Requirements Engineering Conference RE 2010. pp. 179-188. IEEE

[Supakkul and Chung, 12] Supakkul, S., Chung, L.: The RE-Tools: A Multi-notational Requirements Modeling Toolkit. In Proceedings of the 20th International Requirements Engineering Conference RE 2012. pp. 333-334

[Supakkul et al., 13] Supakkul, S., Chung, L., Macasaet, R., Noguera, M., Rodriguez, M., Garrido, J.: Modeling and Tracing Stakeholders' Goals across Notations using RE-Tools. In Proceedings of the 6th International i* Workshop iStar at the 25th International Conference on Advanced Information Systems Engineering CAiSE 2013

[Virus, 17] Virus Worldwide, 2017. http://www.virusworldwide.com

[Withall, 07a] Software Requirement Patterns. 2007. O'Reilly

[Withall, 07b] Software Requirement Patterns. 2007. Microsoft Press

[Young, 04]. Young, R.: The requirements engineering handbook. 2004, Artech House