

Requirements Elicitation and Analysis for Micro-businesses based on Requirements Patterns “ μ bRPs”



Universidad de Granada

Tesis doctoral elaborada para optar al título de:

**Doctor por la Universidad de Granada
dentro del Programa de Doctorado en
Tecnologías de la Información y la Comunicación**

Presentada por:

Ray James Perez Macasaet

Dirigida por:

Dr. José Luis Garrido Bullejos y Dra. María Luisa Rodríguez

Departamento de Lenguajes y Sistemas Informáticos

Universidad de Granada

Granada, 2022

Editor: Universidad de Granada. Tesis Doctorales
Autor: Ray James Perez Macaset
ISBN: 978-84-1117-421-3
URI: <http://hdl.handle.net/10481/75943>

Dedicado a mi madre, Helen Perez Macasaet

AGRADECIMIENTOS

Quiero agradecer a las siguientes personas que están involucradas en esta tesis. Sin sus apoyos y consejos, este trabajo sería inalcanzable.

- José Luis Garrido, mi director de tesis, co-autor.
- María Luisa Rodríguez, mi directora de tesis, co-autora.
- Manuel Noguera, mi co-autor, colaborador en promover las µbRPs por Action Research en Everyware Technologies (España).
- Lawrence Chung, mi co-autor, colaborador en University of Texas at Dallas (UTD) (Estados Unidos).
- Sam Supakkul, mi co-autor, colaborador en el proyecto Requirements Engineering Tools (RE-Tools).
- Antonio Rico, colaborador en promover las µbRPs por Action Research en Desarrollo TIC (España).
- Miguel Quesada, colaborador en promover las µbRPs por Action Research en Virus Worldwide (Filipinas).
- Mila Lavarias, la responsable de la gestión de financiación que viene del Pentathlon Systems Resources Incorporated (PSRI) (Filipinas).
- Los desarrolladores de software en PSRI que han promovido y están promoviendo continuamente las µbRPs en los proyectos de software de las micro-empresas en Filipinas – Kerwin Lim, Jerrick Velasco, Raymund Musa, DJ Ongmanchi, Michael Don Cribe, Rudolph Martin, James Olegario, Bernard Sanchez, Jessie Siat, Dianne Bersabe, John Raymund Calderon, Jonathan Guirigay, Sarah Jane Frias y Rodelio Arcilla.
- María José Rodríguez Fórtiz, colaboradora de nuestro equipo de investigación.
- Carlos Ureña Almagro, anterior coordinador del Programa de Doctorado.
- Héctor Pomares, actual coordinador del Programa de Doctorado en Tecnología de la Información y la Comunicación de la Universidad de Granada (UGR) para la realización de esta tesis doctoral y mi obtención del título de Doctor.

El trabajo de investigación de esta tesis ha sido financiado parcialmente por:

- Spanish Ministry of Economy and Competitiveness under the research project TIN2012-38600
- Spanish Ministry of Science and Innovation under the research projects TIN 2008-05995/TSI and TIN 2007-60199
- Spanish Ministry of Science and Innovation through the Project Ref. PID2019-109644RB-I00 / SRA (State Research Agency) / 10.13039 / 501100011033
- Andalusian Innovation Office under the research project TIN-6600
- CEI BioTIC Granada under the research project 20F2/36
- European Regional Development Funds (FEDER)
- Pentathlon Systems Resources Incorporated (PSRI)
- Erik Jonsson School of Engineering and Computer Science at the University of Texas at Dallas (UTD)

ÍNDICE GENERAL

Publicaciones relacionadas con esta tesis	15
Resumen	17
Summary	21
References	24
Chapter I	
The Micro-business Domain	25
1. Introduction	25
2. Requirements in Micro-business Software Projects	28
2.1. The differences between requirements in large versus micro-businesses ...	28
2.2. A gap in micro-business research	31
2.3. Possibilities for micro-businesses in the future	33
3. Conclusions	34
Chapter References	35
Chapter II	
Related Literature	39
1. The Process for the Review of Related Literature	39
2. The Related Literature	40
2.1. Related Research Proposals	41
2.1.1. Research proposals related to (micro-)business processes	42
2.1.2. Research proposals related to requirements	49
2.1.3. Research proposals associated with software (components)	55
2.1.4. Research proposals related to improving comprehensibility	65
2.1.5. Research proposals related to representing infrastructure	76
2.2 Related Evaluations of Research Proposals put into Practice	78
2.2.1. Evaluations of research proposals using case studies	79

2.2.2. Evaluations of research proposals using Action Research	82
2.2.3. Evaluations of comprehensibility in industry	87
3. Conclusions	90
Chapter References	94
Chapter III	
Theory and Practicality of Modeling in the Domain of Micro-businesses	103
1. The Practical use of the Business Process Modeling Notation	103
2. Practicality with the Unified Modeling Language on the Same Note	108
3. Practical Diagramming and Models for Non-Functional Requirements	111
4. A Practical Characterization and Description of Patterns in Software	114
4.1. Characterization of Patterns	114
4.2. Description of a Pattern	116
5. Conclusions	117
Chapter References	118
Chapter IV	
Micro-business Requirements Patterns “μbRPs”	121
1. Characterization of a Micro-business	122
2. Conceptual Model	125
3. The μ bRP	137
3.1. The Description of a μ bRP	137
3.1.1. μ bRP Table	141
3.1.2. Functional Requirements Table Section.....	142
3.1.3. Non-Functional Requirements Table Section.....	144
3.1.4. BPMN, UML, and SIGs.....	145
3.1.5. Complementary Notes	152
3.2. Adapting SIGs models for the micro-business domain	153
3.2.1. Operationalizing Methods	153

3.2.2. Catalog of Operationalizing methods	155
4. A Real-World Example of an μ BRP in Practice	159
4.1. Requirements Elicitation	160
4.2. Modeling	164
4.3. Instantiation	167
4.4. Software Design	171
4.5. Complementary Notes	173
5. Managing μ BRPs	174
5.1. Creating μ BRPs	174
5.2. Using μ BRPs	178
6. Tool Support for μ BRP Modeling	179
7. Conclusions	180
Chapter References	181
Chapter V	
μBRPs in Industrial Practice	185
1. The Suitability of Using μ BRPs in Micro-business Software Projects	185
2. Evaluating Comprehensibility	187
2.1. Compilation of Interview Results	189
2.2. Potential Benefits of using SIGs in μ BRP Diagrams	190
3. Evaluating Timelines and Affordability	193
3.1. Units of Analysis	193
3.2. Multiple Iterations	194
3.3. Grounded Theory	196
3.3.1. A Brief Background on Grounded Theory	196
3.3.2. Grounded Theory in the Practice of Action Research	197
4. Application of our Evaluation Approach in Other Contexts	199
4.1. A Brief Background of the Scrum Framework	200

4.2. Just in Time Demos	202
4.3. Evaluation of Just in Time Demos	203
5. Conclusions	206
Chapter References	207
Chapter VI	
Conclusion	211
1. Observable Strengths	211
2. Discussion on Weaknesses and Limitations	213
3. Conclusions and Future Work	215
Capítulo VI	
Conclusión	219
1. Fortalezas observables	219
2. Discusión sobre debilidades y limitaciones	221
3. Conclusiones y Trabajo futuro	223
Appendix A	
Catalogue of µbRPs	227
1. Inventory	227
2. Sales	231
3. Logistics	233
4. Production	236
5. Customer Relationship Management “CRM”	239
6. Human Resources	242
7. Accounting	245
8. Management (Reports)	248
9. Restaurant	251
10. Online Retail Shop	254

Appendix B	
Languages and Notations in Practice	261
1. The Business Process Modeling Notation (BPMN)	261
1.1 BPMN Concepts	261
1.2 BPMN Example	269
2. Softgoal Interdependency Graphs (SIGs)	270
2.1 SIGs Concepts	270
2.2 SIGs Example	280
3. Unified Modeling Language (UML)	281
3.1 Class Diagrams	281
3.1.1 Class Diagrams Concepts	281
3.1.2 Class Diagram Example	286
3.2 Component Diagrams	287
3.2.1 Component Diagram Concepts	287
3.2.2 Component Diagram Example	289
Appendix C	
Action Research Material for Micro-businesses	291
1. Sample Form used to Evaluate μ bRP Diagram Comprehensibility	291
2. μ bRP User Guide	293
3. Tutorials	308
4. BPMN Quick Reference Sheet (from bpmb.de)	315
5. SIGs Quick Reference Sheet	316
<i>Bibliography</i>	317

ÍNDICE DE FIGURAS

Figure II.1. The process for the Review of Related Literature	40
Figure II.2. A Venn Diagram illustrating the common ground of all the related research proposals and evaluations to this thesis	41
Figure IV.1 Conceptual model	125
Figure IV.2 Turning Goals into Requirements by Kotonya and Sommerville, 2003	132
Figure IV.3 Overview of the Description of a μ bRP showing prepared and to-be parts...	139
Figure IV.4 Transformation of FRs to business-like, question-answer format	143
Figure IV.5 From question, to mode/option, to choice/answer	144
Figure IV.6 BPMN Diagram for the Restaurant Micro-business (simplified)	146
Figure IV.7 Combining BPMN and SIGs through an operationalization target link	148
Figure IV.8 Combining UML with BPMN and SIGs	151
Figure IV.9 A UML Component Diagram showing required and provided interfaces ...	152
Figure IV.10 Adapting SIGs for Micro-business Software Systems	154
Figure IV.11 Model of the modes or “options” of the Online Retail Shop μ bRP	165
Figure IV.12 Linking μ bRP Tables and Models	166
Figure IV.13 Model of an Online Store μ bRP instantiation with “choices” and “priorities”	169
Figure IV.14 A UML component deployment diagram from the instantiated μ bRP (in Figure IV.13)	171
Figure IV.15 A BPMN model of the process of creating and using μ bRPs	175
Figure IV.16 Grouping and Modeling Common Requirements and Turning Requirements into a Question Format that can be Answered with Yes or No, and corresponding BPMN diagram	177
Figure IV.17 A screenshot of RE-Tools, showing multiple notations, BPMN, SIGs, and UML at the same time	180
Figure V.1. Evaluating the suitability of μ bRPs for micro-business software projects ...	187
Figure V.2. Sample Model Comprehensibility Form	188
Figure V.3 A causal model showing grounded theories built from the Action Research data	199

Figure V.4 Modification of the Scrum Framework in order to make way for JIT Demos ... 203

Figure V.5 Using Grounded Theory to make a Causal Model of JIT Demos 205

ÍNDICE DE TABLAS

Table I.1 Various Characterizations of Micro-businesses	27
Table I.2 Differences of Requirements between Large and Micro-businesses	31
Table II.1 Checklist of Related Literature	91
Table II.2 Table of Related Literature	92
Table III.1 Candidate BPMN Elements for this Thesis	107
Table III.2 Candidate UML Elements for this Thesis	110
Table III.3 Candidate SIGs Elements for this Thesis	112
Table III.4 Characterizations of Patterns in Software	115
Table IV.1 Definition of Small to Medium Sized Enterprises by the European Commission	122
Table IV.2 Definition of Concepts in the Conceptual model	129
Table IV.3 Relationships in the Conceptual model	135
Table IV.4 Description of the parts of a μ BRP	139
Table IV.5 Restaurant μ BRP FR Table Section (shortened)	142
Table IV.6 Restaurant μ BRP NFR Table Section (shortened)	145
Table IV.7 Catalog of Operationalizing Methods	158
Table IV.8 Online Retail Shop μ BRP Requirements Elicitation Table Before Elicitation Meeting	161
Table IV.9 Online Retail Shop μ BRP Requirements Elicitation Table After Elicitation Meeting	163
Table V.1 Summary of the Results of the 16 one-on-one interviews with micro-business owners	190
Table V.2 Observing μ BRPs in Industrial Practice Using Action Research	196
Table V.3 Units of Analysis and Multiple Iterations with JIT Demos	204

Publicaciones relacionadas con esta tesis

Macasaet, R., Chung, L., Garrido, J., Rodriguez, M., & Noguera, M. (2011). 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*, (pp. 50-56). ACM New York, NY, USA. doi: 10.1145/2181101.2181114

Macasaet, R., Noguera, M., Rodriguez, M., Garrido, J., Supakkul, S., & Chung, L. (2012). 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*, Article 7, (pp. 1-6). ACM New York, NY, USA. doi: 10.1145/2424563.2424573

Supakkul, S., Chung, L., Macasaet, R., Noguera, M., Rodriguez, M., & Garrido, J. (2013). 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*, (pp. 128-130). Last accessed on October 9, 2013 at http://ceur-ws.org/Vol-978/paper_23.pdf

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S. & Chung, L. (2013). 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*, (pp.1-12), IEEE. ISBN: 978-1-4673-2912-5. doi: 10.1109/RCIS.2013.6577703

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2014). 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 5 (4), (pp. 71-90), IGI-Global.

Macasaet, R. J. (2017). The Project Start Review Group. In M. Brambilla, T. Hildebrandt (eds.), *Proceedings of the Industry Track of the 15th International Conference on Business Process Management BPM*, (pp. 81-87).

Macasaet, R.J. (2018). Just in Time Demos in the Scrum Framework. *Proceedings of the 3rd International Conference on System Reliability and Safety ICSRS*, (pp.21-24).

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2019). *Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations*. Journal of Universal Computer Science JUCS 25 (7), (pp. 764-787).

Macasaet, R. J., Rodriguez, M. L., Garrido, J. L., & Chung, L. *Requirements Elicitation and Analysis for Micro-businesses: A Pattern-based Approach in Practice*. Submitted to the *Springer Computing Journal* on March 25, 2022.

Resumen

Los requisitos malentendidos pueden causar muchos fracasos en numerosos proyectos software (El Amam & Madhavji, 1995; Kauppinen et al., 2004; Davis et al., 2006). Algunos investigadores (El Amam & Madhavji, 1995) mencionan que la elicitación y el análisis de requisitos es el paso más importante en el desarrollo de software. En (Kauppinen et al., 2004) indican que los requisitos son tan importantes que errores en este paso resultarán en problemas durante el diseño de software e implementación. En (Davis et al., 2006) se describe cómo aproximadamente el 90% de los fracasos en el desarrollo de software vienen de errores en los requisitos.

Es imprescindible que todas las empresas hagan su elicitación y análisis requisitos bien, no sólo las grandes empresas (Young, 2004). Aunque no todos los métodos/técnicas de requisitos y/o la forma de aplicarlos para grandes empresas son útiles para las pequeñas y microempresas. Por ejemplo, el método KAOS (Respect-IT, 2007; van Lamsweerde, 2001; van Lamsweerde, 2009) que es uno de los más populares hoy en día, se usa normalmente en proyectos que necesitan más de 100 días de trabajo. Sin embargo, faltan métodos de requisitos ideados específicamente para microempresas.

Esta tesis propone una manera de llevar a cabo el proceso de elicitación y análisis de requisitos para microempresas. Los requisitos intentan ser comprensibles para los empresarios y otros participantes de las microempresas. Normalmente, los empresarios y otros participantes en las microempresas no están habituados a usar términos técnicos y prefieren usar sus idiomas nativos para expresar sus requisitos a los ingenieros. Esta tesis propone que los términos técnicos no deben ser el tema central (Reijers et al., 2011) en los procesos de ingeniería de requisitos para microempresas. En este sentido, los requisitos para las microempresas deben ser elicitados y analizados mediante procesos ligeros y efectivos (Ambler, 2002) ya que frecuentemente no son proyectos complejos los que se han de desarrollar.

También, la manera de realizar el análisis de requisitos debe ser relevante técnicamente para los analistas/desarrolladores de software. El análisis de requisitos puede acelerar el desarrollo de software por el reuso de requisitos repetidos y consecuentemente también de componentes software ya desarrollados. La comunicación entre ingenieros y empresarios de

las microempresas también mejora. La comprensión de los requisitos se mejora desde las perspectivas de desarrolladores y los empresarios de microempresas.

Para conseguir estos objetivos de comprensibilidad y relevancia técnica, el proceso de ingeniería de requisitos debe responder las siguientes **dos preguntas**:

- (1) ¿Cómo modelamos los requisitos para que los analistas/desarrolladores y empresarios de micro-empresas entiendan comprensiblemente los requisitos y para que los usen técnicamente?
- (2) ¿Cómo se aplica y se evalúa la manera de obtener y analizar los requisitos en la práctica, día a día en el mundo actual?

Gestionar requisitos comprensibles y técnicamente relevantes a la vez son conflictivos y por eso, hemos incluido en nuestra propuesta aportaciones para intentar alcanzar los dos objetivos a la vez.

En el primer capítulo de esta tesis, introducimos el dominio de las microempresas. Describimos que es una microempresa desde diferentes puntos de vista, los requisitos en proyectos de software de microempresas, retos y restricciones en los proyectos de software en microempresas y el futuro de las microempresas.

En el segundo capítulo, revisamos propuestas relacionadas con esta tesis. Analizamos propuestas relacionadas con procesos de negocio, requisitos, componentes de software, la mejora de la comprensibilidad y la representación de infraestructura de sistemas de software. También se resumen investigaciones empíricas relacionadas con la relevancia técnica acerca de la comprensibilidad de los requisitos y del desarrollo del software.

En el tercer capítulo, nos enfocamos en estudiar y describir conceptos de modelado que utilizamos para la propuesta de la tesis. Explicamos la Business Process Modeling Notation (BPMN), los Softgoal Interdependency Graphs (SIGs) y hacemos un repaso de algunos diagramas en el Unified Modeling Language (UML) – tales como diagramas de clases, componentes y actividades.

En el cuarto capítulo, presentamos nuestra propuesta. En particular, presentamos nuestra caracterización de una microempresa, el modelo conceptual, describimos lo que es nuestra propuesta en base a patrones de requisitos de microempresa (acrónimo “µbRP” para

referirnos a *Micro-business Requirements Pattern*) que incluyen tablas, modelos (BPMN, UML, SIGs) y notas complementarias para la implementación de software, y las herramientas que se usan para apoyar la representación de los patrones de requisitos de microempresas. Adicionalmente, presentamos cómo gestionar los μ BRPs en la industria.

En el quinto capítulo, presentamos los resultados de la investigación experimental realizada acerca del uso de los μ BRPs durante el día a día, en el mundo actual. Evaluamos la idoneidad de los patrones de requisitos de microempresas usando investigación de estilo acción. Evaluamos mejoras en tiempo y coste usando los patrones de requisitos. Usando entrevistas, evaluamos las mejoras en comprensibilidad cuando se usan los patrones de requisitos. También, la aplicación de nuestra investigación de estilo acción se presenta en otro contexto en la industria.

En el sexto capítulo, reflexionamos sobre los puntos fuertes y débiles del uso de los μ BRPs. También, presentamos conclusiones y propuestas de trabajo para el futuro.

En los apéndices, se presentan varios patrones de requisitos de microempresas, ejemplos de los formularios que se usaron durante la investigación de estilo acción, las guías para usar las notaciones (BPMN, UML y SIGs), tutoriales y la guía de usuario para el uso de los patrones de requisitos de microempresas.

Summary

Misunderstood requirements can be the cause of failure in various software projects (El Amam & Madhavji, 1995; Kauppinen et al., 2004; Davis et al., 2006). Some researchers (El Amam & Madhavji, 1995) mention that requirements elicitation/analysis is the most important step in software development. In (Kauppinen et al., 2004), the authors mention that the requirements are so important that errors in this step will result in problems during software design and implementation. In (Davis et al., 2006), the authors describe how approximately 90% of failures in software development come from errors in requirements.

It is imperative that all companies do their elicitation and analysis requirements well, not just large companies (Young, 2004). Although not all the methods/techniques of requirements and/or the way of applying them for large companies are useful for small and micro-businesses. For example, the KAOS method (Respect-IT, 2007; van Lamsweerde, 2001; van Lamsweerde, 2009), which is one of the most popular today, is typically used on projects that need more than 100 man-days of work. However, requirements methods designed specifically for micro-businesses are lacking.

This thesis proposes a way to carry out the process of elicitation and analysis of requirements for micro-businesses. The requirements are intended to be comprehensible to micro-business owners and stakeholders involved in micro-businesses. Typically, micro-business owners and stakeholders involved in micro-businesses are not accustomed to using technical terms and prefer to use their native languages to express their requirements to software engineers. This thesis proposes to move away from the use of technical jargon (Reijers et al., 2011) in requirements engineering processes for micro-businesses. In this sense, the requirements process for micro-businesses must elicit and analyze requirements through light and effective ways (Ambler, 2002) since the projects that have to be developed are not always complex.

The way to perform the requirements analysis must also be technically relevant to the software analysts/developers. Requirements analysis can speed up software development by reusing recurring requirements and consequently software components that have already been developed. Communication between engineers and micro-business owners also improves. The comprehensibility of the requirements is enhanced from the perspectives of both the developers and micro-business owners.

To achieve these goals of comprehensibility and technical relevance, the requirements engineering process must answer the following **two research questions**:

- (1) How should requirements be modeled so that they would be comprehensible for micro-business owners and technically relevant for software developers/analysts?
- (2) How could the proposed requirements approach be applied and evaluated in the real world?

Managing comprehensible and technically relevant requirements are somehow conflicting and for this reason, we have included contributions in our proposal to try to achieve both objectives at the same time.

In the first chapter of this thesis, we introduce the domain of micro-businesses. We describe a micro-business from different points of view, the requirements in micro-business software projects, challenges and constraints in micro-business software projects, and the future of micro-businesses.

In the second chapter, we review proposals related to this thesis. We analyze proposals related to business processes, requirements, software components, the improvement of comprehensibility, and the representation of software infrastructure. Evaluation approaches for proposals related to technical relevance and comprehensibility are also reviewed.

In the third chapter, we focus on describing the modeling concepts that we use for the thesis proposal. We explain the Business Process Modeling Notation (BPMN), Softgoal Interdependency Graphs (SIGs), and review some diagrams in the Unified Modeling Language (UML) – such as class, component, and activity diagrams.

In the fourth chapter, we present our proposal. In particular, we present our characterization of a micro-business, the conceptual model, our proposal on micro-business requirements patterns (acronym “ μ bRP”) that includes tables, models (BPMN, UML, GIS), and complementary notes for software implementation, and the tools used to support the representation of micro-business requirement patterns. Additionally, we present how to manage μ bRPs in industry.

In the fifth chapter, we present results from the evaluation on the use of μ BRPs in day-to-day practice. We assess the suitability of μ BRPs using action research. We evaluate improvements in time and cost when μ BRPs are used. Using interviews, we assess improvements in comprehensibility when using the requirement patterns. The application of our action research approach in another context in industry is also presented.

In the sixth chapter, we reflect on the strengths and weaknesses of the use of μ BRPs. We also present conclusions and future work.

In the appendices, there is an initial catalogue of μ BRPs, samples of the forms that were used during Action Research, guides on how to use the notations (BPMN, UML, and GISs), tutorials, and the User Guide for μ BRPs.

REFERENCES

Ambler, S. (2002). Agile modeling. John Wiley and Sons

Davis, C.J., Fuller, R.M., Tremblay, M.C., & Berndt, D.J. (2006). Communication Challenges in Requirements Elicitation and the Use of the Repertory Grid Technique. In *Journal of Computer Information Systems*, 46, (5), 78. url: <http://www.uta.edu/faculty/richarme/MARK%205338/Davis%20repertory%20grid.pdf>

Emam, K. E. & Madhavji, N. H. (1995). A field study of requirements engineering practices in information systems development. *Requirements Engineering*, pp. 68-80, IEEE Computer Society.

Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S. & Sulonen, R. (2004). Implementing requirements engineering processes throughout organizations: success factors and challenges. *Information & Software Technology*, 46, (pp. 937-953). doi: 10.1016/j.infsof.2004.04.002

Reijers, I., Mendling, J., & Dijkman, R.M. (2011) Human and automatic modularizations of process models to enhance their comprehension. *Information Systems*, 36, (pp. 881-897). doi: 10.1016/j.is.2011.03.003

Respect-IT. (2007). KAOS Tutorial Version 1.0. Last Retrieved on December 31, 2021, from <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. Fifth IEEE International Symposium on Requirements Engineering. (pp. 249-262). doi: 10.1109/ISRE.2001.948567

van Lamsweerde, A. (2009). *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley. ISBN: 978-0-470-01270-3

Young, R. (2004). *The requirements engineering handbook*. Artech House. ISBN: 978-1-58053-266-2

Chapter I

The Micro-business Domain

In this chapter, we provide an introduction to the domain of micro-businesses. We discuss the various characterizations made by several researchers on micro-businesses. We provide an overview of how requirements in micro-businesses differ from requirements in large corporations and discuss why it does not make sense to apply the requirements techniques for large businesses on to micro-businesses.

1. INTRODUCTION

Micro-businesses are the smallest of small-to-medium size enterprises (SMEs), varying in terms of ownership, structure, culture, motivation, and market orientation (Ghobadian & Gallear, 1997; Wong & Aspinwall, 2004, Alonso et al., 2018). Micro-businesses share characteristics with their larger counterparts, medium and large-scale enterprises, but there are also several things that set them apart from the rest.

Just like larger businesses, the ownership and structure of a micro-business could be of almost any kind, ranging from being a single proprietorship, an equal partnership, or a group of individuals with unequal share distribution. A single proprietor is one person who is basically running the entire day-to-day business. An equal partnership is a 50-50 split between two individuals who are working towards the same business goal. A group of individuals could be an incorporated company on a micro scale.

The culture in a micro-business can also vary a lot like in any business. Some of them could be very casual while other ones could be very formal. A micro-business could be a casual organization made up of recent music graduates from university who would like to make some extra money by performing music gigs at their local pub. On the other hand, a micro-business could be made up of some ex-executives from large corporations that would like to raise funds to build a sports stadium in their city or neighborhood.

The motivation and market orientation of micro-businesses also vary like any other business. Micro-business could be formed for the purpose of making profits like an online retail store that sells furniture. A micro-business can be focused on a certain market segment like supplying coconuts to restaurants in a local town. There could also be micro-businesses that are one-hundred percent oriented towards society like one that focuses on planting trees to improve the carbon footprint of the population. There are even micro-businesses that have the format of a short-term project which would be crowdfunded like launching a novel product that would not have gotten investment support from bigger investment houses.

Given the variety of micro-businesses, many researchers have characterized them in several different ways (Merten et al., 2011). For example, (Nikula et al., 2000) say that (micro-)businesses must be characterized based on their age – the younger and more fragile a business is, the more “micro” it is. The store that just opened around the corner is more of a micro-business than the supermarket near the town hall that has been operating for more than a decade.

From another point of view, the (European Commission, 2013) and the (International Organization for Standardization (ISO), 2011) say that micro-businesses must be characterized based on the number of employees (ISO refers to micro-businesses as very small entities or VSEs). Businesses with less than 10 people are considered to be micro-businesses. If the supermarket near your town hall has less than ten employees but has been operating for more than a decade, it may still be considered a micro-business by some researchers.

In addition, the (European Commission, 2013) says that micro-businesses must be characterized based on the annual revenue or annual balance sheet which should be less than two million euros. Businesses with more than two million euros in annual revenue or more than two million euros in their annual balance sheet are not considered micro-businesses. If the jewelry store, five blocks from your house, has been in business for only one week but has a balance sheet of more than two million euros then they are not considered a micro-business according to this characterization.

There is the possibility of classifying a micro-business in terms of the number of teams and its variety of customers (Aranda, 2010). The lesser the number of teams and the less variety of customers are more characteristic of a micro-business. If you have three sales teams and

three different types of customers in different geographical locations then your business may not be a micro-business anymore.

Aside from the age and the number of people in a micro-business, its software could also be a basis for classifying it as a micro-business. (Jantunen, 2010) says that the ability to collaborate in software projects should be the basis for characterization where less maturity tends to be more of a micro-business than a larger business. If you have thirty employees in your business but are unable to collaborate properly in a software project with a development team, then some researchers may consider you as a micro-business.

(Aranda et al., 2007) say that the length of software projects describes (micro-)businesses better. Businesses with shorter software projects tend to be classified as micro-businesses. If you have a software project that does not last for more than a week then you may be operating a micro-business according to some researchers.

(Kamsties et al., 1998) contend that the adaptability of (micro-)businesses to software projects must be the basis for characterization. Smaller businesses are seen to be more adaptable to software projects. This means that if your business can adapt to a software project easily then you may have a micro-business according to some researchers. We have an overview of all these characterizations in a table below where you can find the various characterizations that have been made for micro-businesses by various researchers around the world.

Table I.1 Various Characterizations of Micro-businesses

Author(s)	Criteria	Characterization
Nikula et al., 2000	age	the younger, the more micro
European Commission, 2013	number of employees	less than 10
International Organization for Standardization (ISO), 2011	number of employees	less than 10

European Commission, 2013	annual revenue or annual balance sheet	less than 2 million euros
Aranda, 2010	number of teams	lesser teams, more micro
Aranda, 2010	variety of customers	the lesser the variety of customers, the more micro
Jantunen, 2010	ability to collaborate in software projects	the less mature, the more micro
Aranda et al., 2007	length of software projects	the shorter the project, the more micro
Kamsties et al., 1998	adaptability to software projects	the more adaptable, the more micro

There is still heated debate on how micro-businesses and SMEs should be characterized these days. Despite the numerous heated debates, studies on micro-businesses are still rare these days, especially in academia (Kelliher & Reinl, 2009). This thesis is a direct response to the lack of studies being made on micro-businesses. In addition, we also propose our own characterization of a micro-business in Chapter 4 to provide more clarity in the presentation of this thesis.

2. REQUIREMENTS IN MICRO-BUSINESS SOFTWARE PROJECTS

2.1 The differences between requirements in large versus micro businesses

The way that requirements are done in micro-businesses and small companies nowadays bear little resemblance to how requirements are done in larger companies, what the textbooks say, or what is taught at universities (Aranda et al., 2007). Requirements are done for large software projects in large multinational companies but requirements for micro-businesses are not always done and, in some cases, totally neglected (Respect-IT, 2007; van Lamsweerde, 2001). There are several documented cases where small and micro-business software projects have succeeded without formal requirements documentation (Aranda et al., 2007).

However, does this mean that we should not be doing requirements for small and micro-businesses?

Due to their size and particularity, there is no thorough understanding of particular business processes specifically for micro-businesses, e.g., inventory management for micro-businesses in developing nations (Ahmad & Zabri, 2018). This lack of thorough understanding brings us to our next point. (Aranda, 2010) argues that it is a serious omission for requirements researchers to overlook small organizations and that these small organizations should not attempt to emulate the processes and practices of large organizations. In order for the work of requirements researchers to be applicable in the micro-business domain, it must be tailored to suit its specific context (Kassab, 2021).

A concrete example is the Knowledge Acquisition in Automated Specification (KAOS) requirements approach (Respect-IT, 2007; van Lamsweerde, 2001; van Lamsweerde, 2009), one of the most popular requirements approaches nowadays. It has been used in large industrial projects like hospital emergency service management systems, large-scale drug delivery management systems, and large-scale information systems for daily newspaper firms (van Lamsweerde, 2001). Albeit useful for large projects, “a KAOS-like requirements study is worth the effort as soon as the project man power is more than 100 man-days” (Respect-IT, 2007). This means that for a four-man software development team, this project would last for more than five weeks. Not all micro-businesses require this many man-days to implement software for their business.

Another reason why requirements would be different for large organizations compared to small and micro-businesses is because of bureaucracy and formalization (Blau & Schoenherr, 1971; Haveman, 1993). Large organizations are successful because they are predictable. This is due to the inertia created from the formalization of their structures and processes (Hannan & Freeman, 1989). In larger companies, requirements are done by specialists and are documented formally. Changes to requirements are normally done with the approval of a committee.

A specific example is the following. In a multinational organization working on a large project, if there is one branch of the organization in one location, exactly in the same geographical area, and there are two members of the organization sitting next to each other, then communication between these two members would be as if they were in two different

cities, continents apart. Both of them simply cannot talk to each other and agree on changing a requirement or two because they would need approval from several stakeholders.

It is in situations like this where formal requirements documentation is used (Allen, 1977). In a micro-business with two people, the case would be entirely different. Requirements information can be shared with everyone involved, informally, at almost any point in time. The ability to have everyone in the same room sorting out requirements makes formal documentation of requirements in small organizations completely unnecessary (Lethbridge et al., 2003).

Aside from the absence of bureaucracy and formalization, micro-businesses can develop rapport or working relationships with lesser effort as compared to their larger counterparts. Agile software development methodologies advocate a close communication between the developers and the customers. In this way, the development team can understand the needs and the culture of their clients in a more intuitive manner, resulting in faster resolution of technical and non-technical issues (Beck, 2005). Capitalizing on positive relationships and culture provide benefits during requirements and the rest of the software project (Alsanoosy et al., 2020).

The manner of communication differs between micro-businesses and large companies too. Micro-business owners are normally not exposed to technical languages nor do they have the extra time to learn software jargon (Kamsties et al., 1998) (Kauppinen et al., 2002). They would comfortably use their internal “oral traditions” (Aranda, 2010), natural language, and drawings to express their requirements. Using unfamiliar words and phrases causes misunderstanding and communication breakdowns with stakeholders during requirements (Alsanoosy et al., 2018). Hence, communicating micro-business software requirements must be done as intuitively and as comprehensively as possible (Kruchten, 2003), with as little or no technical software terms (Young, 2004). Requirements of this kind could be referred to as “lightweight but effective” (Ambler, 2002).

The requirements models used in smaller firms are unlikely to be formal. Based on a recent survey, small firms use semi-formal modeling three times more than formal methods and informal modeling eight times more than formal methods (Kassab, 2021). This survey points out that smaller firms tend to shy away from formal modeling and tend towards semi-formal and informal modeling.

Given the differences between large and micro businesses and their software requirements, it is also important to note that software systems are more and more intertwined with business processes as well (Immes, 1993). This means that if the business processes are different in large businesses as compared to small and micro-businesses then their software systems and requirements would have differences as well. We present a table below which shows the differences in requirements between large and micro-businesses.

Table I.2 Differences of Requirements between Large and Micro-businesses

Requirement	Large Business	Micro-business
Method	Standardized, defined, bureaucratic, and predictable	Intuitive
Author	Specialized engineer	Anyone
Communication	Functional boundaries	Direct rapport
Terminology	Technical Jargon	Lightweight but effective
Models	Formal	Semi-formal and Informal

2.2 A gap in micro-business research

As will be elaborated in the next chapter during the review of related literature, there are software requirements techniques for large companies and little or hardly any for micro-businesses which means that there is not much attention being given to software requirements for micro-businesses. Despite this, small and micro-businesses must still take their software requirements seriously because sloppy requirements eventually turn into problems during software design, implementation (Kauppinen et al., 2004), acceptance, and essentially threaten the overall success of projects (Davis et al., 2006). Proper requirements are essential, no matter the size of the business (Young, 2004).

For the software developers working on micro-business projects, there are a specific set of challenges to address. The style of communication with micro-business owners must be

“lightweight but effective” (Ambler, 2002) with as little or no technical software terms (Young, 2004). Most micro-business owners do not have the time to learn technical software jargon (Kamsties et al., 1998; Kauppinen et al., 2002). Instead, they will always try to express their requirements using their own natural language and illustrations.

If requirements are made as lightweight as possible, do they start to lose their technical relevance for developers? After all these years, developers have been studying and working with complex, technical systems with all the jargon and terminologies which help themselves communicate effectively with each other.

Developers working on micro-business software projects need requirements that are technically relevant. Analyzing information such as the technical requirements details, the design of the software, and effort estimates are of value to the developers. The challenge is to make requirements both technically relevant and comprehensible for micro-businesses.

Aside from comprehensibility and relevance, there are also resource constraints specific to micro-businesses. All modern software development companies, in which close to 99% are small-to-medium in context (Fayad et al., 2000), operate in a competitive market with time and cost constraints (Cugola & Ghezzi, 1998). For example, the number of man days expended in software projects (unit of time) multiplied by the daily rate for software developers equals direct labor costs (unit of cost) for a software project. Hence, to lower costs and make software more affordable for micro-businesses, developers must be timely and minimize the man days expended in projects.

How could a software development company with micro-business software projects minimize man days expended in projects and become more competitive in the market? Software (component) reuse saves time and contributes to minimizing man days expended in projects. One way to promote software (component) reuse is through patterns (Sommerville, 2004). We provide an extensive literature review on patterns in the next chapter.

There is not a lot of research on the way requirements elicitation is done in industry (Palomares et al., 2021). Even with the existence of patterns in literature and given the specific challenges for software requirements for micro-businesses, there are surprisingly almost no requirements approaches specifically made for micro-businesses, also known as Very Small Enterprises “VSEs” (International Organization for Standardization (ISO), 2011). If we found

some proposal in our review of related literature which could be suitable for micro-businesses, it would have shortcomings that would prevent us from progressing in our research.

For example, there is the Modeling by Example (MbE) proposal by (Kalenborn, 2010). MbE appears to be pragmatic and applicable enough in the micro-business domain although it still falls short in terms of specifications as a requirements approach. Moreover, the MbE approach does not prioritize dialogue with micro-business owners since it is more focused on winning bids for software projects, where dialogue between the developers and the micro-business owners is limited and involves conflicting goals. More details on the MbE and the shortcomings of other research proposals is provided in the systematic review of related literature in the next chapter.

2.3 Possibilities for micro-businesses in the future

Despite the lack of suitable requirements techniques for micro-businesses, software process improvement efforts, particularly in the area of requirements (Villalón et al., 2002; Bae, 2007; Pino et al., 2008), are continuously being made for SMEs (Dybå, 2003) up to this day. The future of solutions and improvements is bright for micro-businesses although it is not determined nor set in stone. (Richardson & Wangenheim, 2007) states that the challenges faced by micro-businesses and their software projects involve:

- Managing and improving the software processes
- Dealing with rapidly changing technology
- Maintenance of products
- Operation in a global software environment
- Sustenance of the organization through growth

However, the last challenge which involves the growth or stagnation of micro-businesses is still debatable. (Aranda, 2010) argues that *"...many in the software industry assume that the goal of a small firm should be growth, that size is a valid measure of success. To be sure, a large size brings certain benefits: the appearance of stability, the ability to engage in greater and more ambitious projects, the appeal of commanding the work of a large number of employees. And yet there are many rewards for small organizations, rewards that often go unnoticed and unclaimed in their push to become large by behaving as if they were already large. Some of these rewards are psychological and even ethical, such as the joy of working*

in closely-knit groups and a greater agency over one's own work. That kind of reward may be significant enough to justify a preference for small groups..."

To say that the future of micro-businesses is - to grow, become bigger, and cease to be a micro-business - is not one-hundred percent set in stone. Rather, the future of micro-businesses is driven by the goals made by its founders, owners, and stakeholders. Not all of them dream to be big. Some of them are perfectly happy where they are. Through the goals of micro-business owners and software developers, we study their requirements in detail throughout this thesis. We see what kind of future micro-business software projects may have and more questions that may unfold before us in this domain.

3. CONCLUSIONS

In this chapter, we have provided an introduction to the domain of micro-businesses. We have looked at the various characterizations made by several researchers on micro-businesses and have seen that there is heated debate going on on how micro-businesses should be characterized. For further clarity in this thesis, we provide our own characterization of micro-businesses in Chapter 4.

Taking into consideration the objectives of this thesis, we have also provided an overview of how to address requirements in micro-businesses in a more direct way than in large corporations. Micro-business requirements are usually a subset of requirements found in more complex systems developed for larger enterprises. Moreover, micro-business have far more limitations than larger enterprises when it comes to the availability of resources. Hence, requirements elicitation and analysis for micro-businesses could be made in a more specific way using effective lightweight approaches. In this direction, as we elaborate in the next chapter, we did not find enough proposals that are suitable for micro-businesses.

CHAPTER REFERENCES

- Ahmad, K. & Zabri, S. (2018). The mediating effect of knowledge of inventory management in the relationship between inventory management practices and performance: The case of micro retailing enterprises. *Journal of Business and Retail Management Research*, 12, 2. (pp. 83-93). doi: 10.24052/JBRMR/V12IS02/TMEOKOIMITRBIMPAPTCOMRE
- Allen, T. (1977). *Managing the Flow of Technology*. MIT Press
- Alonso, A., Kok, S., Sakellarios, N., & O'Brien, S. (2018). Micro-enterprises, self-efficacy and knowledge acquisition: Evidence from Greece and Spain. *Journal of Knowledge Management* 23 (3), (pp.419-438). doi: 10.1108/JKM-02-2018-0118
- Alsanoosy, T., Spichkova, M., & Harland, J. (2018). Cultural influences on the requirements engineering process: lessons learned from practice. In: 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS). IEEE, (pp 61–70)
- Alsanoosy, T., Spichkova, M. & Harland, J. (2020) Cultural influence on requirements engineering activities: a systematic literature review and analysis. *Requirements Eng* 25, (pp. 339–362). <https://doi.org/10.1007/s00766-019-00326-9>
- Ambler, S. (2002). *Agile modeling*. John Wiley and Sons
- Aranda, J., Easterbrook, S. M. & Wilson, G. (2007). Requirements in the wild: How small companies do it. *Requirements Engineering RE* (pp. 39-48), IEEE. ISBN: 0-7695-2935-6. doi: 10.1109/RE.2007.54
- Aranda, J. (2010). Playing to the Strengths of Small Organizations. In *Proceedings of the 1st Workshop on RE in Small Companies RESC*, (pp. 141-144). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Bae, D.-H. (2007). Software Process Improvement for Small Organizations. *COMPSAC* (1). (p. 17). IEEE Computer Society. ISBN: 978-0-7695-2870-0. doi:10.1109/COMPSAC.2007.193
- Beck, K. (2005). *Extreme Programming Explained: Embrace Change*; 2nd Edition. Addison-Wesley Professional
- Blau, P. & Schoenherr, R. (1971). *The Structure of Organizations*. Basic Books
- Cugola, G. & Ghezzi, C. (1998). Software processes: a retrospective and a path to the future. *Software Process: Improvement and Practice*, 4, (pp. 101-123). doi: 10.1002/(SICI)1099-1670(199809)4:3\$<\$101::AID-SPIP103\$>\$3.0.CO;2-K
- Davis, C.J., Fuller, R.M., Tremblay, M.C., & Berndt, D.J. (2006). Communication Challenges in Requirements Elicitation and the Use of the Repertory Grid Technique. In *Journal of Computer Information Systems*, 46, (5), 78. url: <http://www.uta.edu/faculty/richarme/MARK%205338/Davis%20repertory%20grid.pdf>
- Dybå, T. (2003). Factors of software process improvement success in small and large organizations: an empirical study in the Scandinavian context. *ESEC / SIGSOFT FSE* (pp. 148-157), ACM. doi: 10.1145/940071.940092

European Commission. (2013). User Guide to the SME Definition. Last accessed on April 3, 2021 at https://ec.europa.eu/regional_policy/sources/conferences/state-aid/sme/smedefinitionguide_en.pdf

Fayad, M., Laitinen, M., & Ward, R. (2000). Thinking objectively: software engineering in the small, *Communications of the ACM* 43, (pp. 115-118). doi: 10.1145/330534.330555

Ghobadian, A. & Gallea, D. (1997). TQM and organization size. *International Journal of Operations & Production Management*, 17, pp. 121-163

Hannan, M., & Freeman, J. (1989). *Organizational Ecology*. Harvard University Press

Haveman, H. (1993). Organizational size and change: Diversification in the savings and loan industry after deregulation. *Administrative Science Quarterly*, 38, pp 20-50

Immes, S. (1993). *Wahrgenommenes Risiko bei der industriellen Kaufentscheidung*, Trier

International Organization for Standardization (ISO). (2011). ISO/IEC DTR 29110-1:2011 Software Engineering – Lifecycle Profiles for Very Small Entities (VSEs) – Part 1: Overview. ISO, Switzerland, 2011. Retrieved October 9, 2013, from <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>

Jantunen, S. (2010). 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*, (pp. 131-140). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Kalenborn, A. (2010). Modelling by Example: Requirements engineering during the bidding stage of dialog-oriented software projects. In *Proceedings of the 1st Workshop on RE in Small Companies RESC*, (pp. 158-166). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Kamsties, E., Hormann, K., & Schlich, M. (1998). Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer. In *Conference on European Industrial Requirements Engineering CEIRE*, London, United Kingdom. url: <http://prof.kamsties.com/download/ceire98.pdf>

Kassab, M. (2021). How Requirements Engineering is Performed in Small Businesses? 29th International Requirements Engineering Conference Workshops (REW), 2021, IEEE, (pp. 220-223), doi: 10.1109/REW53955.2021.00041.

Kauppinen, M., Kujala, S., Aaltio, T. & Lehtola, L. (2002). Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice. *RE* (pp. 43-51). *IEEE Computer Society*. ISBN: 0-7695-1465-0. doi: 10.1109/ICRE.2002.1048504

Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S. & Sulonen, R. (2004). Implementing requirements engineering processes throughout organizations: success factors and challenges. *Information & Software Technology*, 46, (pp. 937-953). doi: 10.1016/j.infsof.2004.04.002

Kelliher, F. & Reinl, L. (2009). A resource-based view of micro-firm management practice. *Journal of Small Business and Enterprise Development*, 16 (3), (pp. 521-532). doi: 10.1108/14626000910977206

Kruchten, P. (2003). *The Rational Unified Process: An Introduction*. Boston, MA: Addison-Wesley. ISBN: 0201707101

Lethbridge, T. C., Singer, J. & Forward, A. (2003). How software engineers use documentation: the state of the practice. *IEEE Software*, 20, 35--39. doi: <http://dx.doi.org/10.1109/MS.2003.1241364>

Merten, T., Lauenroth, K., & Bürsner S. (2011). 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* (pp. 60-65). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-19858-8_7

Nikula, U., Sajeniemi, J., & Kalvianen, H. (2000). A state-of-the-practice survey on requirements engineering in small-and-medium-sized enterprises. In *Telecom Business Research Center Lappeenranta Research Report 1*, Lappeenranta University of Technology. url: <https://www.uop.edu.jo/Homeworks/13544442010.pdf>

Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., Lopez, L., & Gorschek, T. (2021). The state-of-practice in requirements elicitation: an extended interview study at 12 companies. *Requirements Eng* 26, (pp. 273–299). doi: 10.1007/s00766-020-00345-x

Pino, F. J., García, F. & Piattini, M. (2008). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal*, 16, (pp. 237-261). doi: 10.1007/s11219-007-9038-z

Respect-IT. (2007). KAOS Tutorial Version 1.0. Retrieved on January 15, 2013, from <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>

Richardson, I. & Wangenheim, C.G. (2007). Why are small software organizations different? *IEEE Software*, 24, (1), pp. 18-22

Sommerville I. (2004). *Software Engineering*. Addison-Wesley: Harlow, England.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. *Fifth IEEE International Symposium on Requirements Engineering*. (pp. 249-262). doi: 10.1109/ISRE.2001.948567

van Lamsweerde, A. (2009). *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley. ISBN: 978-0-470-01270-3

Villalón, J. A. C.-M., Agustín, G. C., Gilabert, T. S. F., de Amescua Seco, A., Sánchez, L. G. & Cota, M. P. (2002). Experiences in the Application of Software Process Improvement in SMES.. *Software Quality Journal*, 10, 261-273.

Wong, K. Y. & Aspinwall, E. (2004). Characterizing knowledge management in the small business environment. *Journal of Knowledge Management*, 8, pp. 44-61

Young, R. (2004). *The requirements engineering handbook*. Artech House. ISBN: 978-1-58053-266-2

Chapter II

Review of Related Literature

In this chapter, we provide a review of related literature and the current state-of-the-practice which exposes the research gaps and justifies the need for our proposed requirements approach today.

1. THE PROCESS FOR THE REVIEW OF RELATED LITERATURE

The review of related literature was performed using the following steps:

1. Search for manuscripts on Google Scholar, DBLP, and the University of Granada Library. The keywords used to start the review of related literature are Micro-business, Small Business, Business Process, Software Requirements, Comprehensibility, and Evaluation.
2. Select relevant manuscripts based on title. For example, from the keyword “business process,” we encounter the manuscript “Business process patterns and frameworks: Reusing knowledge in process innovation” (Barros, 2007).
3. Select relevant manuscripts based on abstract. We read the abstract of the manuscript based on title and if the information found on the abstract is relevant to our research, we proceed to the next step.
4. Read relevant manuscripts.
5. If there are relevant citations in a manuscript that are related to our research then we select the cited manuscript and repeat step 4. If there are none then proceed to step 6.
6. We summarize the relevant findings in the related literature in this chapter.

A diagram of the review of related literature process is presented in Figure II.1. The result of the review of related literature, as stated in step 6, is presented in the next section of this chapter.

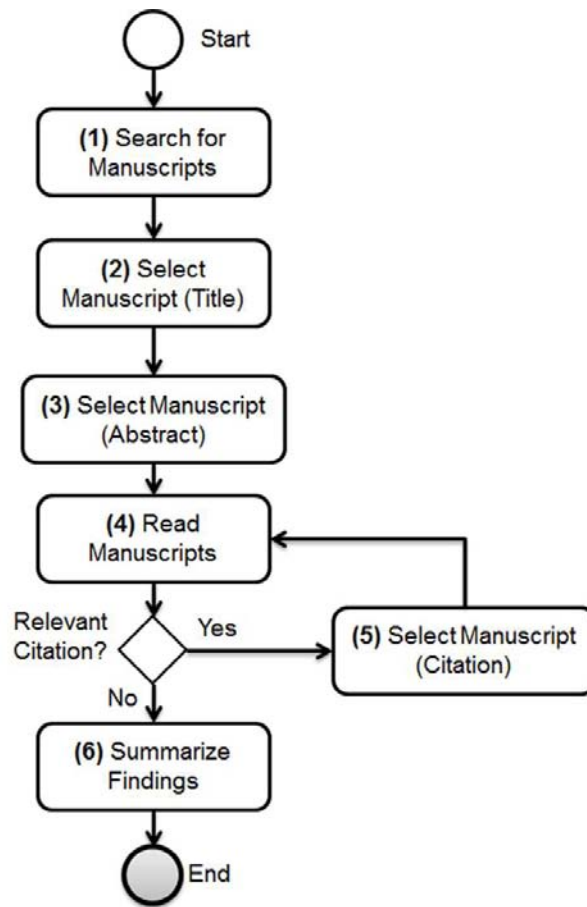


Figure II.1. The process for the Review of Related Literature

2. THE RELATED LITERATURE

We have divided the results of the review of related literature into two parts: literature related to other research proposals and literature related to the evaluations of proposals in industrial practice. The first part aims to inform the reader of what is currently happening in the research community while the second part aims to inform the reader of results and observations when the research is applied in practice. These two parts are important because the work of this thesis involves both a research proposal and putting the research proposal into practice as well. We have provided a Venn diagram in Figure II.2 to illustrate the common ground of all the related literature and our proposal.

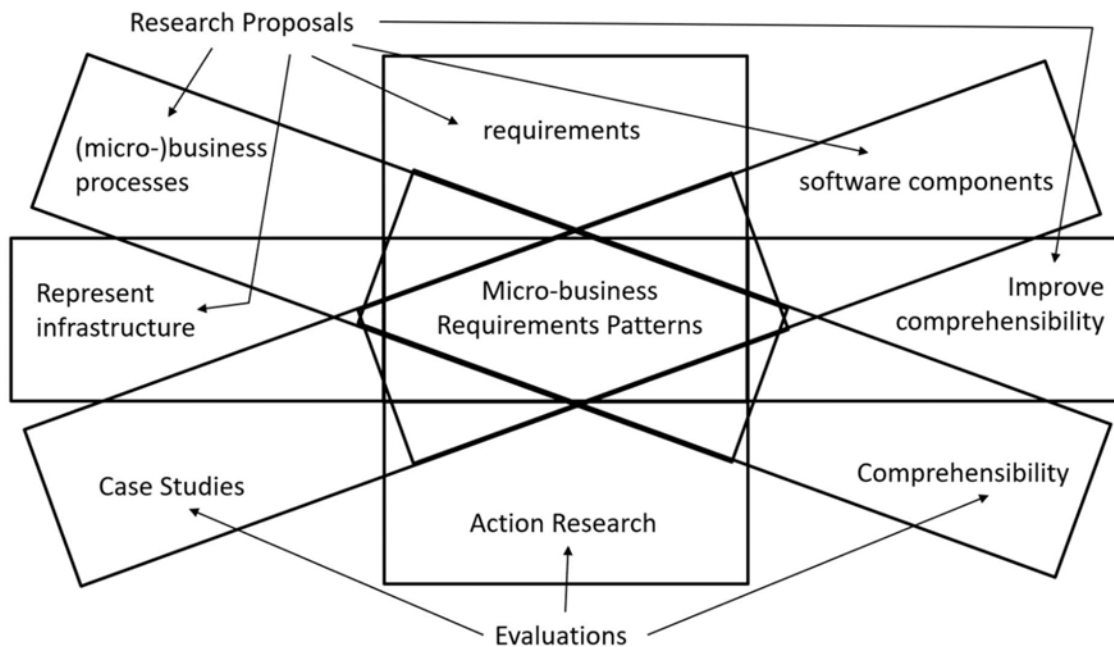


Figure II.2. A Venn Diagram illustrating the common ground of all the related research proposals and evaluations to this thesis

In the Venn Diagram in Figure II.2, the rectangles represent an area of related research. The areas of research that are research proposals are: micro-business processes, requirements, software components, representing infrastructure, and improving comprehensibility. The areas of research that are evaluation are: case studies, (evaluation of) comprehensibility, and Action Research. All of the rectangles intersect in the middle because it is related to our proposal: micro-business requirements patterns.

At the end of the chapter, a summary of all the related proposals and evaluations are presented in Table II.1 and a checklist of how each of the related proposals and evaluations relate to our research work is presented in Table II.2.

2.1. Related Research Proposals

This subsection enumerates the related research proposals that we have encountered in literature. They are related to our proposed requirements approach in terms of one or more of the following: (micro-)business processes, requirements, software (component) reuse, infrastructure representation, or are made to improve comprehensibility. Albeit **related**, each

proposal could be somewhat applicable but not entirely applicable to all the challenges we have described in the previous chapter. We discuss the relationships and the shortcomings of each proposal in the following subsections.

2.1.1. Research proposals related to (micro-)business processes

First, we enumerate research proposals related to (micro-)business processes and applications. The Modeling by Example or “MbE” approach, proposed by (Kalenborn, 2010), argues that requirements have to be done even before the project starts. The approach is designed for use during the bidding stages of a software project.

The idea of this approach is to come up with requirements in a practical manner in preparation for the actual bid (in order to win the software contract). The approach has to be practical because the resources used during this preparation stage still cannot be justified with full assurance. The chance that the software developer wins the contract would depend on various factors such as their competence and creativity (Brennan et al., 2008), among others. Moreover, if there are five bidders then the chance to win the contract would be 20%, assuming that all bidders are equally competent and all have an equal chance of winning the software contract. In any case, the practical manner of the approach makes it a semi-suitable requirements approach for small and micro-businesses.

The objective of the approach is to write a bid as quickly as possible, relying on predefined specifications. The information in the MbE-type requirements documents include project-related costs, schedule, and visual representations of the actual product in terms of screenshots or mock-ups where functional and technical specifications are assigned to each element of the mock-up. The requirements are done using an MbE tool which sources modules and templates to come up with the mock-ups for presentation to the client (micro-business owner).

(Kalenborn, 2010) justifies the information in the requirements documents based on the IKIWISI phenomenon “I-know-it-when-I-see-it” which is that software users do not understand the requirements until they see them (Boehm, 2000). (Kalenborn, 2010) argues that decision makers (which in this thesis are the micro-business owners) are often not able or not willing to understand abstract models or descriptions and in many cases, the screenshots are the only things understood by the decision makers because the technical details of an application are so complex that only the software developers are capable of understanding and assessing

them. Hence, MbE is driven by the visualization of applications, which is particularly important in web-based projects. This is also a strength of MbE as an approach.

Although the MbE has its practical merits (for micro-business software projects), we see the lack of technical detail (for the software developers), the lack of dialogue between micro-business owner and software developer, and its over emphasis on web-based projects as caveats to the approach. Documents which are designed to win bids on a software project are entirely different from documents which are designed to specify requirements correctly for a software project. We argue that if technical requirements cannot be understood then the solution should not be to remove them from the documents but instead, to explain the technical issues better. Moreover, creating mock-ups should involve regular consultation with the micro-business owner for feedback. Regular consultation during the bidding stages is not very common since there are also other bidders who want to have regular consultations with a micro-business owner who only has a limited amount of time.

In addition, if the MbE is designed for bidding purposes then the software developer and micro-business owner are unlikely to be completely aligned in terms of goals, for there are other bidders with other interests (and other requirements) in mind. There is a thin line between requirements documents designed to win contracts and requirements documents designed to specify requirements as correctly as possible. MbE's lean toward the former. Lastly, if the MbE is focused on web-based projects then software projects of different nature and with varying NFRs would make the MbE approach not the most suitable for micro-business software projects in general.

A research proposal based on business processes is one from (Barros, 2007) who proposes business process patterns (BPP) which result in business object frameworks (BOF), 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 use of patterns can be traced back to the work of (Alexander et al., 1977) in the field of architecture. Subsequently, the use of patterns has become evident in several other domains as will be detailed. For instance, using business patterns and finding out where the patterns can be instantiated in specific business contexts makes the work of software analysts and designers easier and more challenging (Kilov & Sack, 2009).

Patterns in the case of BPPs are generalized process models based on best practices which include activities, flows, and logic. The best practices are based on empirical knowledge of how the activities of a process in the best companies of a given domain are performed. In order to delve into further detail in a BPP, a domain is specified and then the business logic and flows are defined more precisely.

From the BPP models, flows, and logic, BOFs are derived which incorporate the knowledge about the solution of relevant problems in a particular domain, resulting in its ability to provide generalized solutions to the said problems. The mapping from BPPs to BOFs is based on the following:

- The structure of the BPP system supports while the business logic of the domain defines the BO classes that encapsulate the algorithms or heuristics that provide the solutions to the problem in various cases in that particular domain.
- The BO structure is modeled with UML class diagrams where the operations or methods for classes are defined based on business logic.
- The data in the business logic can be used for executing operations.
- Data can be structured into data classes that interact with BOs which can also be expressed in UML.

When using a framework to develop a software application which will support a process in a real-world case within the domain of the framework, the following procedures are performed:

- A relevant substructure of the framework which can be applied to the case is selected.
- The substructure is specialized based on the characteristics of the case, adding data and logic when necessary.
- A detailed design for an appropriate technology is made.
- Coding commences.

The main advantage of the BPPs and BOFs are that they are able to offer pre-built solutions that would allow developers to select, among several possibilities, the functionalities that will solve their problem. The BPPs and BOFs could also be specialized if they do not have a complete fit with the problem.

One of the shortcomings that we have noted with BPPs is its lack of connection with the software components that would be needed to implement the solutions. Although the process

patterns may be reused, there was little or no mention of how the software solutions will be put into place.

The resources, events, agents (REA) model has been extended by (Hruby, 2006) with several structural and behavioral business patterns. The REA-based patterns are used to develop business-related (software) applications by searching for business objects and related modeling elements.

The idea of using REA is that one would be able to determine unknown pieces or links of software design by using relationships between modeling elements. The argument is compared to mathematical equations which would not change such as the $e=mc^2$ equation by Albert Einstein.

In the REA extension, the business relationships between resources, events, and agents are assumed to be just as fixed, which in turn, gives rise to the REA-based patterns. The REA-based patterns are those applied in day-to-day business management and development of business (software) applications.

When we were reviewing REA to find ways to meet our challenges, it fell short when it came to challenges specific to resourcing for micro-businesses. There was no mention of optimizing resources such as software reuse and the reduction of software development timelines.

Service-based cooperation patterns (SBCPs) are proposed by (Boukheduoma et al., 2013). 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.

The basic idea of the SBCPs is to deal with IOWFs flexibly. There are three main dimensions of an SBCP. The first is the structuring of IOWFs into services. Each workflow fragment is encapsulated into a single composite service (or set of services) depending on the kind of IOWF architecture to meet. The workflows can be encapsulated because they have both technical and conceptual similarities with services.

Second is the control of their execution. This is expressed through global and local orchestration functions. The functional, behavioral, and interactional aspects of SBCPs are provided in detail.

The third pertains to the structure of interaction with external services which are provided by other partners taking part in the cooperation. These proposed SBCPs are applied on process models during design time.

Upon reviewing SBCPs, they fell short when it came to its use for micro-businesses which have limited resources. There is no mention of software reuse related to the business processes specific to micro-businesses nor there is any mention of attempts to reduce the timelines in software development projects.

Document-based patterns are proposed by (Glushko & Mcrath, 2002; Glushko & Mcrath, 2008). They introduce the discipline of “document engineering” and argue that document exchange is the mother of all patterns. They cite an example of an Aramaic tax receipt which was created over 2000 years ago and which is still preserved and considered documented up to this day. They further argue that with careful design, the same documents can be reused in different business processes.

The activity of document analysis ushers the discipline of document engineering. From document engineering arise the document patterns. Based on a set of analysis and design techniques, document-centric patterns can be created. The patterns are characterized as tangible and easy to analyze and are (re-)usable when designing business processes.

These document patterns promote reuse, which results in reduced maintenance, better consistency, and standardization. The patterns are also proposed to be in a library or a repository so that they can be found easily. They propose that the document engineered models be the front-end to the libraries because document-engineered artifacts provide the metadata and query structure which will guide searches for the needed patterns.

The document patterns fell short when it came to how software developers would communicate with micro-business owners. There was little or no attention paid to the difficulty that micro-business owners will have when they start encountering the technical jargon in the documents.

Domain-specific patterns for mobile service businesses are proposed by (Aleksy & Stieger, 2011). 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.

The mobile service business patterns were proposed in response to four pressing factors in industrial service business sectors from high-wage countries: an aging workforce, competition, product complexity, and partnerships. The ability to provide tailored information support and usability is both an opportunity and a challenge faced in industrial field service applications. Several companies have gained efficiency improvements in field service processes via mobile technologies.

The four patterns are as follows. The first pattern is the container pattern and it is designed to foster the integration of third parties by hiding the internal organization of the partner. The second pattern is the mobile phone as a primary device pattern and it defines a primary communication channel to structure the general way of communication between the back office and mobile workers. The third pattern is the information package to go pattern and it is designed to foster the offline working capability of mobile workers. Finally, the fourth pattern is the pluggable information sources pattern and it is designed to provide tailored information support to mobile knowledge workers.

The combined use of the mobile service business patterns can also be used to address certain problems as well. For instance, for tailored information support for offline processing, both the information package to go and the pluggable information sources pattern may be used together. For enabling offline processing capabilities for third-party service staff, both the container pattern and the information package to go pattern may be used together.

The mobile service business patterns proposed by (Aleksy & Stieger, 2011) are designed to support industrial field service processes but they think that the patterns may also be used outside this area such as in M-Business applications. Also, roles such as mobile service engineer could be replaced with other agents such as insurance agents or real estate agents in cases where similar problems arise.

When we were reviewing this proposal, there were no specific methods for instantiating the patterns in other specific cases. Based on the examples, if similar industrial cases arise, such

patterns may be applied. Although this was not discussed in detail and would still be subject to testing and evaluation in practice. The mobile service business patterns were also too specific to be used in several cases especially with the variety of cases in the micro-business domain.

A systematic way of capturing and reusing patterns based on their specific business domains is proposed by (Seruca & Loucopoulos, 2003). 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.

(Seruca & Loucopoulos, 2003) stress the fact that patterns are not invented and they must be found in existing models that characterize real-life business systems. Hence, the development of patterns is essentially about identification, collection, and codification of knowledge and not creation of patterns out of nothing. Also, the process of discovering patterns is an empirical activity and that patterns are built by observing practice in a domain and by trial-and-error.

Given the arguments just mentioned, (Seruca & Loucopoulos, 2003) propose a systematic approach for collecting and analyzing business domain knowledge in order to support the capture and development of business patterns. This approach is called the PattCar method. This method is composed of two main procedures, namely pattern collection and pattern reuse. The main purpose of the method is to design the patterns for reuse. According to (Coplien, 1995a; Coplien 1995b), patterns must help us understand existing organizations and also help us build new ones.

The steps of the PattCarr method and the resulting documents from the steps are as follows:

- Define the domain and analyze the context, resulting in domain definitions and business context models.
- Define domain core business process and vocabulary, resulting in domain vocabulary and domain taxonomy of business processes.
- Describe Sub-domains in terms of existing generic business processes, resulting in sub-domain process specialization.

- Develop Sub-domain enterprise models for a number of businesses, resulting in process capability models, use case models, and object models.
- Define the patterns for the Sub-domain, resulting in commonality and variability analysis and patterns defined in a template.
- Organize and interrelate the patterns, resulting in patterns classified in a hierarchy of subjects, facets, and links to other patterns.

The PattCar method has been used in collaboration with a consulting firm and seven of their clients, involving seven Portuguese SMEs in the clothing manufacturing domain. The firms involved aimed to describe, evaluate, and redesign their business processes and concepts.

The PattCar method is not designed as a substitute for creative or insightful pattern creation. It is mainly designed to promote:

- A disciplined observation of practice in a business domain.
- The creation of domain models which match the knowledge and the experience acquired.
- The discovery of business patterns from the study and analysis of the domain models.

The PattCar method falls short when it comes to resource management for software projects for micro-businesses. Although there is mention of reuse in practice, there is no mention of the reuse of the software or how the patterns could help in optimizing resources such as reducing timelines in software projects for micro-businesses.

2.1.2. Research proposals related to requirements

Second, we enumerate the proposals related to requirements. 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., 1999).

(Franch et al., 2010) propose 29 software requirements patterns (SRPs) which aim to be used during requirements elicitation, documentation, and validation. The reuse of the SRPs aim to help requirements engineers in eliciting, validating, and documenting software

requirements and as a result, come up with software requirements specifications which have better quality in terms of content and syntax. There has been a great percentage of reuse of NFR information in the SRPs in call-for-tender requirement specifications for selecting commercial off-the-shelf (COTS) software.

The 29 SRP patterns proposed were all focused on NFRs since they were the least sensitive to changes in the said problem domain. The 29 SRP patterns were based on a study of 7 real-world software requirements specifications in real-world call-for-tender projects. The structure of an SRP is made of:

- Goals: drives the elicitor in the selection of the pattern to be applied in a project
- Description: short phrases which describe the pattern, like an abstract
- Forms:
 - Fixed Parts: expresses the pattern itself, the general purpose of the pattern, and its identification
 - Template
 - Extended Parts Constraint: if not complied, proceeds to extended parts
 - Extended Parts: extends the fixed parts and describe the technical elements of a pattern
 - Template
 - Parameter(s): take on specific values when the pattern is applied. Metrics are defined for each parameter

The application of an SRP is dependent on choosing and applying the most suitable form dependent on the parameters. There are also relationships among the SRPs which are:

- Pattern relationships: the most general relationship and implies related patterns based on the forms and the parts of the forms
- Form relationship: the relationship at the level of forms which implies all the parts of the related forms
- Part relationships: the relationship which applies to two parts

In a related work by (Mendez-Bonilla et al., 2008) which also proposes the previously mentioned SRPs, bases the SRPs on their experiences working in different projects in different domains such as mail server systems, e-learning software, and web content management systems. They observed similarities in requirements already being used among the projects.

The SRPs mainly pertained to FRs, NFRs, and non-technical requirements and aim to be of use in COTS-based systems.

When we reviewed the SRPs from (Franch et al., 2010) and (Mendez-Bonilla et al., 2008), we noticed that there was a lot of technical jargon which would not be suitable for the majority of micro-business owners without technical backgrounds.

(Hoffman et al., 2012a; Hoffman et al., 2012b) propose SRPs based on user trust. The SRPs are based on studies from the behavioral sciences, collecting antecedents that build trust. The SRPs are used mainly in recommender system development projects.

The idea of the trust-based requirements patterns is based on the ability of users to adopt trust and reduce social complexities which are caused by the lack of knowledge or information about how information systems work. The trust-based patterns aim to translate the results from research on the trust of users in new technologies into the said requirements patterns.

Antecedents that build trust are collected and then they are developed into a set of requirements patterns. The requirements patterns are made to be functional enough to support the antecedents. The trust-based SRPs are specified with:

- The name of the trust-based SRP
- The requirements engineering activity it pertains to, such as elicitation or specification
- Type of pattern
- Stakeholders involved in the pattern
- Goals
- Problems
- Forces
- A predefined requirement template which has a solution and which can be used for further specification
- Examples of where the trust-based SRP can be applied

When we reviewed the trust-based SRPs, they had a very particular use case and were comprehensible to micro-business owners however, there was no mention of improving resource management for micro-business owners such as reducing timelines for software projects.

Security requirements patterns are proposed by (Riaz & Williams, 2012). A security requirement is somewhat a security policy and a security mechanism. A security policy states what is and what is not allowed while a security mechanism is a method, tool, or procedure which will be used to enforce the security policy (Bishop, 2003). Security requirements are identified by analyzing assets, threats, and vulnerabilities while considering the multiple points of views of users or attackers.

When systems have the same security objectives, security requirements may be reused. Security requirements patterns capture common security requirements, document the context in which a requirement manifests itself, and describe the trade-offs involved. Part of the proposal of (Riaz & Williams, 2012) includes an outline for developing SRPs and strategies for specifying reusable security requirements. In a related work, security test patterns are proposed by (Ben & Williams, 2012) in order to aid in black box security testing.

(Álvarez et al., 2002) propose hierarchically structured parameterized and non-parameterized templates for reusable security requirements. These templates comply with IEEE standards for specifying quality requirements. Such quality attributes are NFR in nature such as identification, priority, criticality, viability, risk, source, and traceability.

The elicitation process for the security requirements is largely based on the spiral model for requirements engineering (SIREN) and states explicitly that the requirements are to be reused. In this proposal, a repository for reusable requirements is maintained, with an annotation to a domain and profile, for annotation for the possibility of reuse. For instance, if the domain is in finance and the profile is information systems security then the requirement may be reused if there is a match. What is lacking in the proposal is a step-by-step procedure on instantiating the templates and the consequences when the requirements templates are reused. (Firesmith, 2004) also proposes parameterized templates to model reusable security requirements but lacks the step-by-step procedures on instantiating such templates.

(Schumacher et al., 2006) documents security requirements patterns as well, including patterns for secure design and architecture. The catalog uses natural language and includes security requirements patterns pertaining to Access control, Audit, Intrusion Detection, Non-repudiation, and Accounting.

(Wen et al., 2011) propose security requirements patterns which have a focus on Ownership, Authorization, Attack and Protection, Analysis of Assets, Threats, and Attacks.

The work of (Withall, 2007a; Withall, 2007b) on security requirements patterns covers 1- Access control, in particular, the registration, authentication, and authorization, 2 - Audit, and 3 - Privacy, in particular, archiving and compliance with standards.

Connected closely to security is privacy. Privacy has been a concern of many when it comes to software systems. There must be a systematic approach in order to meet privacy requirements of a software system and (Peixoto & Silva, 2018) do just that. They propose a framework for privacy modeling capabilities that must be addressed by requirements modeling languages to better support privacy specification.

The privacy modeling capabilities are used to compare three goal-oriented modeling languages, namely i*, NFR-Framework, and Secure-Tropos. The framework is based on a conceptual foundation and model of privacy which was built from an analysis of a standard, regulation, guidelines, and other bibliographical sources related to privacy. An example related to health care is used to show how the framework can be used to compare the chosen modeling languages.

The study involved fourteen privacy modeling capabilities which were defined in the framework. They observed that the analyzed modeling languages do not fully support them. The study concluded that the proposed framework contributes towards the consolidation of a privacy conceptual foundation and that it can also be used to evaluate modeling languages for privacy in requirements engineering. The comparison performed by using this framework also indicates that Secure-Tropos is the most complete language to model privacy among the analyzed goal-oriented modeling languages.

Although all the security and privacy requirements patterns were useful in specific cases, they were limited when it came to their practical application to micro-businesses in general. Their practicality when it came to resource management of micro-businesses was not evident, especially when reducing timelines and software development time in micro-business projects.

Security and Privacy are just some types of NFRs and it would be unmindful not to mention NFR patterns in general when it comes to the field of requirements engineering. Four NFR patterns have been proposed by (Supakkul et al., 2010). The purpose of the NFR patterns is to capture and reuse them in business-specific cases with the help of NFR visualizations and representations. The four NFR patterns proposed are:

- Objective pattern: used to identify important NFRs for a context or capture a particular definition of an NFR due to various interpretations stakeholders may have when it comes to NFRs.
- Problem pattern: used for capturing the knowledge of soft problems. Soft problems are problems without clear-cut resolution criteria (Supakkul & Chung, 2009).
- Alternatives pattern: pertains to two things: the alternative means for achieving a softgoal and alternative solutions for mitigating a soft problem. The alternatives may be captured along with one or more side effects.
- Selection pattern: used when faced with alternatives for a softgoal or soft problem and the user of the pattern must choose an alternative that maximizes the positive while minimizing the negative side effects.

The NFR patterns are organized as follows:

- The knowledge of an NFR which is captured in an NFR pattern may be specialized for more specific situations using refinements, resulting in sub-patterns. The pattern specialization relationships are of partial order, reflexive, anti-symmetric, and transitive, which means as a result, a specialized pattern is further specialized by other patterns.
- Pattern composition consists of pre-assembling multiple patterns into a new pattern in order to form a larger chunk of knowledge. Again, the “part-of” relationship between a pattern and its composite pattern is partial order, reflexive, anti-symmetric, and transitive which means as a result, a composite pattern could be assembled to become a part of another larger composite pattern.
- When instantiating the patterns, a piece of knowledge would be applicable in similar situations. Pattern instantiation also allows a pattern to be used as a template when creating another new pattern.

The NFR patterns proposed by (Supakkul et al., 2010) have been applied in an empirical study. The objective was to find out whether the approach could capture and reuse relevant NFR knowledge in a project or an organization which is similar in nature. The results were positive, providing preliminary evidence that the patterns could help in capturing, organizing, and reusing a big chunk of NFR knowledge in model and tool-based requirements engineering.

NFR visualizations are also proposed by (Chung et al., 2013). The NFR visualizations are geared towards combining goal-oriented requirements engineering techniques with simulations for cloud computing cases, complementing qualitative goal models in the NFR Framework (Chung et al., 2000) with quantitative approaches, and for the development of an interactive, iterative, and interleaving simulation technique.

The NFR visualizations are applied in a case study involving a contactless smartcard system. The smartcard system was created to automate and harmonize the ticketing in all the public transportation channels in Victoria, the most densely populated state in Australia. The public transportation system consists of trains, buses, and trams.

The proposed approach involving the NFR visualizations consists of the following steps:

- Identification of the stakeholders and goals
- Analysis of stakeholder goals, including the identification of conflicts
- Quantitative augmentation of the qualitative goal model using domain specific information

The NFR visualizations involve the use of Softgoal Interdependency Graphs (SIGs) as proposed in the NFR Framework (Chung et al., 2000). The SIGs are practical when it comes to relating and diagramming the relationships between NFRs and operationalizations, e.g. transforming goals into tangible, operational solutions for micro-businesses. However, the NFR patterns and SIGs fell short when it came to optimizing resources for micro-businesses such as reusing software components and reducing timelines for micro-business software projects. SIGs are explained further and also adapted for the micro-business domain as will be discussed in the next chapters.

2.1.3. Research proposals associated with software components

Third, we enumerate the proposals related to patterns in software (component) reuse. According to (Schmidt et al., 1996), software patterns are a means of providing successful solutions to common software problems. The software patterns community identifies several benefits when patterns are used by practitioners, namely:

- Facilitation and the ease of reuse
- Identification and capture of abstract concepts

- Aid in defining interfaces and interactions
- Means of sharing documentation
- Construction of software with defined properties
- Provisions for a common vocabulary (as described in detail by (Budgen et al., 2008))

Software patterns are known to have been used in the following stages of software development:

- Requirements, as has been described in the previous subsection
- Analysis (Fowler, 1997)
- Design (Gamma et al., 1995)
- Architecture (Fowler, 2002)
- Testing (Smith & Williams, 2012)
- Security (Yoshioka et al., 2008)
- Configuration Management (Berczuk, 2003)

(Crnkovic et al., 2002) propose the use of patterns in component-based software engineering (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.

The CBSE discipline is a combination of several different disciplines in software engineering and computer science, including object-oriented programming, reuse, software architecture, modeling languages, and formal specifications. The discipline of CBSE is still growing and there are still several concepts that are still not formalized, terms which are not clearly defined, and relationships which are still not explained very well. Even the term “component” is still under discussion and has not yet been formally specified. Proposals in the discipline of CBSE are discussed as follows.

(Kouroshfar et al., 2009) propose a generic process framework for component-based development. This framework is based on what was commonly encountered in seven component-based development methodologies. These seven component-based development methodologies are the following:

- UML components is a UML-based methodology which aims to help developers use COM+ and JavaBeans technologies for defining and specifying software components (Cheesman & Daniels, 2003).
- Select Perspective (Rumbaugh et al., 1991) is the combination of object modeling language with the Objectory use-case-driven process which was later integrated into the Rational Unified Process (RUP).
- The Feature-Oriented Reuse Method (FORM) (Kang et al., 2002; Sochos et al., 2004) provides a component-based development methodology by adding architectural design and construction of object-oriented components to the Feature Oriented Domain Analysis (FODA) (Kang et al., 1990) which presented the idea of using features in requirements engineering in 1990.
- Kobra is a methodology aimed at developing high quality, component-based software systems in a systematic manner. The methodology is based on several software engineering technologies such as product-line engineering, frameworks, architecture-centric development, quality modeling, and process modeling (Atkinson, et al., 2002; Atkinson et al., 2000).
- Adaptive Software Development (ASD) was introduced in 1997 and is an agile method that promotes component-based development (Ramsin & Paige, 2008).
- Catalysis is a component-based approach which is based on object-oriented analysis and design and provides a framework for component-based software development (Ramsin & Paige, 2008).
- The Rational Unified Process (RUP) is a use-case-driven, architecture-centric, object-oriented methodology which incorporates specific guidelines for component-based development (Ramsin & Paige, 2008). RUP has recently evolved into a method engineering framework called Rational Method Composer (RMC).

The proposed generic framework as a result of noting the similarities of these seven component-based methodologies is as follows:

- The Analysis Phase is when the requirements of the system are elicited. During this phase, the infrastructure of the project is defined and a preliminary project plan and schedule is outlined. During this phase, the applicability of component-based software development is assessed.
- The Design Phase is when the components of the system are identified and specified based on their interactions with one another.

- The Provision Phase is when components are classified as being retrievable from a repository of reusable components or needed to be written from scratch. Components are also tested during this phase.
- The Release Phase is when components are assembled in order to form the final system. A system test is conducted and then the final system is deployed live.

The generic process framework of (Kouroshfar et al., 2009) provided strong connections and relationships between requirements and software development although such framework must be further specialized when used for micro-businesses. There is still much technical jargon which may not be suitable for micro-business owners.

(Stepan & Lau, 2012) propose controller patterns which are abstractions for defining coordination in the context of CBSE. The patterns are used in component-based development for control software for reactive systems. Reactive systems are systems which continuously react to their environment (Harel & Pnueli, 1985). The behavior of a reactive system is described as an infinite cycle involving the following steps:

- Reading inputs from the environment
- Computing the reaction of the system
- Outputting the reaction back to the environment

Some examples of reactive control systems are cruise control systems in cars and control systems which prevent meltdowns in nuclear power plants. The controller patterns which aid in the component-based development of such systems are composed of an interface and constructors.

The interface determines the interactions in the system architecture and is composed of the following:

- Data ports which define the entry and exit points for the data routed by the connectors, which are further defined by constraints in terms of type and directionality.
- Control ports which are the center of control coming from a superior coordinator.
- Control parameters which are connected to subordinates.

The constructors in the controller patterns are used to build up the connectors and are composed of the following mechanisms:

- Aggregation which is the way of composition in which control and data flows are defined, without any interaction between flows.
- Composition via data ports is the way of composition in which data flows are defined by a data connector.
- Composition via control ports is the way of composition for the control connectors.
- Hierarchies are used for complex controller patterns which consist of other simpler patterns that are useful on their own.

The controller patterns are demonstrated in a case study using a prototype tool and have proven to be useful in providing (reusable) abstractions suitable for the construction of controllers in reactive software systems.

The controller patterns of (Stepan & Lau, 2012) demonstrate a strong relationship between requirements and software components although they have a lot of technical jargon and would not be suitable for requirements elicitation for micro-business owners without technical backgrounds.

A component specification structure which is based on analysis and design patterns is proposed by (Paludo et al., 2011). The purpose of the specification 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.

One of the problems that this proposal addresses is the ability to find reusable components (Alnusair & Zhao, 2010). Even when there is a component repository available to the developers, it does not automatically mean that the components are easily discoverable. If it took significant effort to discover the components then trying to reuse the components could be even more difficult than writing the components from scratch.

The other problem that this proposal addresses is that when trying to achieve reuse, software architecture must be considered. Software architecture is the basis for an entire family of systems which is built using common assets. A mistake that is made by most software development organizations is treating the architecture of a family of systems across an enterprise with the same levels of abstraction (Allen, 2001). According to (Clements et al.,

2002), software architecture is an organizational asset which is created at a considerable expense and should be reused.

Given the aforementioned challenges, (Paludo et al., 2011) propose the following component specification which would aid in reuse. The specifications involve:

- The name of the component
- Alternative names for the component
- Properties of the component, considering its type, subtype, and level
- Type of the component, being creational, behavioral, structural, or system
- Subtype of the component, relating to specification, implementation, execution, or deployment
- Level of the component, being unit, component, or architectural
- Purpose and context, explaining the scope and the environment under which the component exists
- Problem which is a brief description of the problem to be treated by the component which includes presentation of the design issues which are faced by the developer. In this specification, an example would be helpful.
- Applicability of the component which includes the pros and cons, including the drawbacks, when the component will be used
- Description of the component which includes a detailed discussion of the component, what the component does and how it behaves.
- Structure Solution of the component which is a class diagram including the basic solution structure and a sequence diagram which represents a dynamic model
- Solution Strategy which presents the ways that a component can be implemented
- Interfaces of the component which is the way the component makes its service available where common multiple interfaces may be provided in response to various points of access
- Forces of the component which include a list of the rational and motivational aspects that affect the problems and the solutions. Points that may be included in this specification are reasons why one would choose to use the component and justifications on why the component would be used.
- Quality characteristics and sub-characteristics addressed by the component and if possible, accepted software product quality metrics could be provided
- Sample code for the component could be provided if possible
- Variants of the component could be provided for alternate implementations

- Related components which are basically other components which are associated internally or externally from the perspective of the repository

Based on specifications like this, (Alnusair & Zhao, 2010) propose search methods which would involve one or more of the following: semantics, keywords, and signatures. With an effective search method, reuse of components would be promoted.

The component specification structure proposed by (Paludo et al., 2011) makes it easier to reuse software components by making them more searchable in a repository or a database. However, such a specification still uses a lot of technical jargon and would not be suitable for eliciting requirements for micro-business owners who have limited technical backgrounds.

It is important in the design of software that future changes and extensions may be easily incorporated without severely affecting other quality attributes so that the software may be more maintainable. (Kouskouras et al., 2008) investigate how the adoption of design patterns and aspect-oriented programming techniques could be useful in fulfilling this purpose. The investigation was done in the telecommunications industry because of the size and the evolution of systems in this field. As a result, several alternatives were proposed and they are the following:

- Naming patterns could be used which would force any new command and parameter class to adopt a specific naming pattern which would be included in a specific package.
- Registry patterns are proposed in order to overcome design limitations. The registry patterns could be used with or without aspect-oriented programming techniques. Such design limitations address awkward provisioning of different customizations like couplings between new command classes and other classes in the module representing their functional area.

Implementing one component and another component altogether does not necessarily mean that both components would work together seamlessly. Hence, (Elizondo & Lau, 2010) propose a catalog 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 order to promote the reuse of components and the component connectors, it is important that the component connectors be treated with importance and adequate descriptions and specifications are made for them (the connectors). Hence, (Elizondo & Lau, 2010) propose the following catalogue of component connectors to support development with reuse.

The first group of component connectors is the adaptation connectors. They are the following:

- Guard connectors are used to “guard” the execution of the computation in the adapted component in accordance to the evaluation of a Boolean expression
- Condition controlled loop connectors are used for repeating the execution of computations in the adapted component according to the evaluation of a Boolean expression
- Counter-controlled loop connectors are used to repeat the execution of the computation in the adapted component a specific number of times
- Delay connectors are used for delaying the execution of computation in adapted components for a specific period of time

The second group of component connectors is the composition connectors. They are the following:

- Sequencer connectors provide a composition scheme where the computation in the composed components is executed sequentially one after the other
- Pipe connectors provide a composition scheme where computation in the composed components is executed sequentially one after another and the output of an execution is the input of the next one and so on and so forth
- Selector connectors provide a composition scheme where the computation in only one of the composed components is executed based on the evaluation of a Boolean expression

The third group of component connectors is the composite composition connectors. They are the following:

- Observer connectors provide a composition mechanism where once the computation in the “publisher” component has been pre-formed, the computation in a set of “subscribers” components is executed sequentially

- Chain of responsibility connectors provide a composition mechanism where more than one component in a set can handle a request for computation
- Exclusive choice sequencer connectors provide a composition mechanism where once the computation in a “predecessor” component has been pre-formed, the computation of only one component in a set of “successor” components is executed
- Exclusive choice pipe connectors are a version of the exclusive choice sequencer connector with internal data communication among the “predecessor” and the “successor” components
- Simple merge sequencer connectors provide a composition mechanism where once the computation in only one component in a set of “predecessor” components has been pre-formed, the computation in a “successor” component is executed
- Simple merge pipe connectors are a version of the simple merge sequencer with internal data communication between the “predecessor” and the “successor” component

All these thirteen proposed component connectors have been defined by taking into consideration the syntax semantics of new component models. Also, the feasibility of implementing this proposed catalogue in industrial practice has been demonstrated.

The component connectors proposed by (Elizondo & Lau, 2010) could be useful for the software architecture and design of software systems. However, they are more oriented to larger software systems and not to micro-business software systems.

In line with component connectors, (Bhuta et al., 2007) proposes a framework for selecting component connectors. They discuss a framework for selecting commercial-off-the-shelf (COTS) software components and connectors with the goal of ensuring that they are interoperable. Standard definitions for COTS components and connectors are used in this framework. A COTS system, adapted from the SEI COTS-Based System Initiatives definition (Albert et al., 2002) could be:

- Sold, leased, or licensed to the general public
- Offered by a vendor trying to profit from it
- Supported and evolved by the vendor, who retains the intellectual property rights
- Available in multiple identical copies
- Used without source code modification
- Open-source where code may be modified by users (Bhuta et al., 2007)

If a component is defined as a unit of computation or data store (Medvidovic & Taylor, 2000) and may be as small as a single procedure or as large as an entire application, then the component connectors are defined as architectural building blocks which are used to model interactions among components and rules that govern those interactions (Medvidovic & Taylor, 2000).

Piecing together available Open-source components and COTS components is very different from traditional development. While the latter has a requirements-design-develop-test-deploy process, the former has an assessment-selection-composition-integration-test-deploy process (Albert & Brownsword, 2002; Ballurio et al., 2003; Boehm et al., 2003; Comella-Dorda et al., 2002; Yang et al., 2005). The assessment and selection steps of the former are important and are composed of:

- assessing both functional and non-functional requirements for the COTS system
- assessing the interoperability to ensure that the selected COTS components would interact with each other properly

The first assessment has a fair share of solutions (Albert & Brownsword, 2002; Ballurio et al., 2003; Boehm et al., 2003; Comella-Dorda et al., 2002; Yang et al., 2005) while the second assessment continues to be puzzling for researchers. For instance, in a study conducted by (Garlan et al., 1995), a base set of four reusable software components were used to construct a software system. Prototyping the COTS interactions as it would occur in the conceived system became time and effort intensive. In the interest of limited resources, developers are compelled to neglect the interoperability issue altogether, simply hoping that there will be no problems during integration or are compelled to neglect interoperability until the number of COTS combinations are reduced to a manageable number. Both of these cases increase project risk dramatically.

Under the assumption that interoperability is completely neglected, developers could end up writing tons of glue code (connector code) which in turn expends resources. In software architecture, the connectors are the embodiment of the interactions and associations between software components and must be dealt with importance (Shaw et al., 1996). Hence, (Bhuta et al., 2007) propose an attribute-driven framework that attempts to address the selection of COTS components and connectors so that they would be interoperable. The proposed framework is composed of:

- COTS interoperability evaluator
- COTS representation attributes
- Integrations Rules

The result of applying the framework is an interoperability assessment report. This proposal is also automated and enables the evaluation of a large number of architectures and COTS combinations, able to analyze large trade-off spaces for COTS component and connector selection. Since COTS characteristics are evolving constantly, this kind of framework must be constantly updated.

The interoperability of software components and the attribute-driven framework proposed by (Bhuta et al., 2007) are focused on minimizing resource expenditure on software projects through proper reuse of software components and their connectors. However, the systems that this proposal addresses are for large software systems and not for micro-businesses which are way less complex.

2.1.4. Research proposals related to improving comprehensibility

Fourth, seeing proposals with a lot of technical jargon in the previous subsection, we enumerate the proposals related to patterns which improve comprehensibility, including proposals which involve patterns in the comprehension of software modeling languages and notations.

(Lakhal et al., 2013) propose patterns for Unified Modeling Language (UML) profiles. Four aspects of the patterns are identified, relating to the change, impact, reusable solutions, and in relation to other evolving patterns. The patterns are meant to make adaptation to evolving UML profiles less costly. The pattern-based approach has the following objectives:

- Distinguish atomic evolutions from complex evolutions. An atomic evolution is when one independent change is applied on one profile element while a complex evolution is a combination of atomic changes with dependency links. The proposal consists of analyzing changes in order to identify the atomic operations groups which describe the recurrent complex evolutions.
- Identify the evolution patterns and their formalization. In order to achieve this objective, an empirical study is made on the evolution of two automobile domain profiles.

- Classify the patterns so that there will be faster adaptation of the patterns. As (Buschmann & Meunier, 1995) would say, it is important to provide a guide when selecting a pattern for a particular design situation. The classification scheme must have categories for criteria or design issues that play a significant role in software development. With an evolution pattern classification according to impact criteria in place, there could be a reduction in overall costs to achieve adaptation (of the patterns).

The proposed pattern catalog must offer:

- The ability to identify recurrent evolution issues
- The ability to formalize specific relations towards and between patterns in an understandable format
- The ability to classify the evolution pattern in order to facilitate their access and their reuse in different instantiations

In order to meet such objectives, **four aspects** of evolution patterns are proposed and specified as follows:

- The **interface** has all the elements which allow for selecting a pattern. The elements are:
 - The name is used to name the evolution treated in the pattern and the associated solution
 - The problem is used to describe the context of the evolution and the problem which is resolved by the pattern
 - The keywords are used to cite all the profile elements invoked in the pattern and the kind of evolution. The kind of evolution could be addition, removal, modification, among others
 - The classification parameters are used to reference the key parameter of each element implied in the pattern and their key values used to classify the pattern.
- The **solution M2** has all the elements which are needed to describe the evolution at a profile level, meaning the meta-operation instantiations which are needed to describe the evolution. The elements are:
 - The change representation which specifies the evolution by the group of the change operations as defined in the delta model

- The informal describes the elements in which the pattern is applied and the context in which it is executed, using natural language
 - The formal describes the sequence of operations needed to realize an evolution, the function call, and the settings filling, using formal language
 - The visual describes the evolution treated in the pattern using a profile diagram
- The **solution M1 with category** defines the solution and the instantiation for adaptation of the instance models to the new profile version. This aspect is used for identifying different solutions for the same pattern. The elements are:
 - The informal is a description of the evolution where in each category, parameters that have an impact and values which belong to the category are expressed.
 - The formal describes the informal using formal language
 - The visual describes the result of the pattern solution on the models using a representation of the model before and after the evolution
- The **relations** define the relation between patterns in order to organize the patterns catalog. Four kinds of relations are based on the following:
 - Use
 - Required
 - Alternative
 - Refine

This proposal for evolution patterns in UML has been tested on a prototype named Papyrus Profile Evolution (P²E) in order to implement an entire catalog in an industrial case study. Although it was difficult to cover all possible profile evolutions, the formalism enabled extensions and explanations in relation to how the patterns were used in combinations and instantiations.

The UML profiles proposed by (Lakhali et al., 2013) have attempted to make technical notations such as UML to be more comprehensible to more users. However, such profiles are still not oriented for the domain of micro-businesses.

A rigorous and practical technique for specifying pattern solutions in UML, which also serves as a supporting guide for the development of related tools, is proposed by (France et al., 2004).

Formal pattern specification using mathematical languages are capable of providing the necessary concepts to precisely describe pattern solutions (Eden, 1999; Lano et al., 1996). The problem with formalities is that sophisticated mathematical skills are needed in order to use these kinds of patterns. Hence, patterns with specification languages that are based on familiar software modeling concepts such as UML would more likely be used by software developers (France et al., 2004). The reasons for using UML are the following:

- UML is considered to be the de facto standard for object-oriented modeling which means that design patterns using UML models would be relevant.
- Model-driven architecture (MDA) is being promoted by the Object Management Group (OMG). In MDA, models are used as the primary artifacts for development. MDA has raised the level of abstraction for which complex software systems are developed and tools to support the models and patterns have become necessary. The tools would require that patterns be precisely specified in modeling notations such as UML.

In order to specialize the UML meta-model for pattern specifications, the following must be done:

- The abstract syntax must be specialized by subtyping UML meta-model classes and by making well-formed-ness rules more restrictive. This would result in an abstract syntax for models describing the pattern solutions.
- Parameterized Object Constraint Language (OCL) (Warmer & Kleppe, 1999) must be defined. These would be the constraint templates which represent constraints that must be expressed in models characterized by the specialized meta-model. The semantic properties of the patterns are captured using the parameterized constraints.

The pattern specification as proposed by (France et al., 2004) would consist of a structural pattern specification (SPS) and a set of interaction pattern specifications (IPSs). The SPSs, which are the core of the pattern specification, would specify the class diagram view of the pattern solutions. The SPS notation would consist of the following:

- A classifier role, which consists of three parts, namely:
 - The label of the form, also known as the name of the meta-model class
 - The declaration of the form, indicating the name

- Realization multiplicities which are used to restrict the number of classifiers playing the role in a conforming class diagram. Multiplicities may be omitted if the number of conforming classifiers does not have any constraints. In UML, this would be denoted with an *.
- A structural feature role is used to specify the properties represented by structural features of conforming classifiers, such as an attribute or a query.
- A behavioral feature role is used to specify behavioral properties which are associated with conforming classifiers, such as an operation.

The IPSs specify the interactions among the pattern solutions and its definitions would be based on the terms and roles defined in an SPS. The SPS roles are used to specify the participants in an interaction pattern. IPSs are specified with:

- The invocation of an operation of a subject. This is an operation that conforms to the feature roles, resulting in calls to the operation in each observer linked to the subject.
- Each operation, calling an operation in the subject.

Based on the SPSs and IPSs, (France et al., 2004) have been able to fully develop pattern specification for the following design patterns (France et al., 2002), namely:

- Abstract factory
- Bridge
- Decorator
- Singleton
- Observer
- Composite
- Visitor

These pattern specifications have been presented and used by graduate students in a software engineering course in order to develop the specifications of the design patterns. All the students were familiar with UML and patterns based on their previous courses.

The pattern specification proposed by (France et al., 2004) could be used as a base for tools that support the creation and the evolution of pattern and for a rigorous application of design patterns to UML models. Also, this UML-based notation, which is tool-independent, could facilitate the sharing of design patterns among UML modeling tools.

In relation to UML-related patterns, (Kim et al., 2004) propose a role-based meta-modeling language (RBML) which is primarily used for expressing domain-specific patterns. The RBML is a sub-language of UML and can be used by developers when creating UML diagrams. The RBML is demonstrated using a check-in-check-out (CICO) application for use at rentals such as car rentals, book rentals, and video rentals.

The work of (France et al., 2004) and (Kim et al., 2004) show that UML can be made more comprehensible when modeling patterns. However, there would always be those who would argue that UML is complex and not easy to adapt and learn. Specifically, (Siau & Cao, 2002) say that UML is 2-11 times more complex than other modeling methods. Since UML is a technical language, micro-business owners without technical backgrounds would have a difficult time communicating with software developers if only UML-based diagrams are used.

(Oliveira & Belo, 2012) have chosen BPMN to express their patterns because of its clarity and simplicity in process representation. BPMN is expressive, very capable for implementation, and could control tasks within models. Business process modeling notation (BPMN) patterns which can be used for extract-transform-load (ETL) systems are proposed by (Oliveira & Belo, 2012).

The ETL-BPMN patterns are designed to map standard data warehousing ETL processes and test them before deploying final systems. The work of (Oliveira & Belo, 2012) extends and builds on the ETL-BPMN work done by (Akkaoui & Zimány, 2009; Akkaoui et al., 2011). ETL systems are known to be complex yet could be specified conceptually, in a very concrete way, and subsequently, the models could be validated when running the model defined in BPMN.

In order to properly model ETL processes, the different flows of control and data between various tasks must be reflected (in the model). BPMN is capable of including the specific features for the representation and description of data flows.

The proposal of (Oliveira & Belo, 2012) allows the creation of models that can be defined as containers of operations in which their specifications depend on an input of data or tasks. The output is a set of pre-established activities. (Oliveira & Belo, 2012) use the Bizagi tool (Bizagi, 2013) for demonstrating their work. Although the patterns of (Oliveira & Belo, 2012) use a more business-oriented notation, such concepts like ETL continue to be technical and

oriented to developers and would not be the most suitable way of communicating with micro-business owners.

In relation to other BPMN-based patterns, Workflow Activity Patterns (WAPs) in BPMN are proposed by (Thom et al., 2011). The objective of their proposal is two-fold:

- Easy adoption of WAPs when using BPMN tools. BPMN is already becoming a well-known standard notation for business process modeling. BPMN is also suitable for modeling WAPs.
- Use of WAPs in business process design which would help in the automation and facilitation when designing process models, reducing process modeling time and cost, improving process model quality, and enabling the reuse of captured process knowledge.

Based on their previous work (Thom et al., 2009a; Thom et al., 2009b), 200 real-world process models were analyzed in order to confirm the existence of the following seven WAPs:

- Approval Pattern is a pattern which involves the approval of a single role or multiple roles either concurrently or iteratively
- Question-Answer pattern is a pattern which involves sending to one or multiple roles and actors respectively
- Uni-directional Performative is a pattern which involves activity execution requests being sent to one or multiple actors, waiting for a response from one end is not necessary
- Bi-directional Performative is a pattern which involves activity execution requests being sent to one or multiple actors, waiting for a response from one end is necessary
- Notification is a pattern which involves notifications sent to one or multiple actors
- Informative Request is a pattern which involves information requests sent to one or more multiple actors
- Decision is a pattern which involves a final decision based on the results of an activity or a set of activities

WAPs (in BPMN) are ideal for designing processes from different application domains and organizations and could be used as a conceptual framework for building workflow patterns in the micro-business domain. However, they must still be more lightweight for the micro-business owner to understand.

In some cases, a new definition language may have to be designed to fulfill a particular purpose. For example, a design pattern definition language (DPDL) for representing software design patterns has been proposed by (Khwaja & Alshayeb, 2013). The objectives of the DPDL are to be:

- Easy to understand and to use. If the DPDL is understood by the designers easily then it should be easy to use
- Unambiguous which means that the DPDL should be as clear as possible because any ambiguity would result in bugs in software production and eventually reducing the quality of the software product
- Extensible because since technology changes and progresses, the DPDL must be able to accommodate extensions
- Based on existing technology so that it would be able to get wider and faster acceptance
- Supportive of graphical output so that quick overviews of the design patterns may be done. Even if the language is text-based, it should be able to support graphics

The DPDL design patterns are made up of three parts:

- Attributes define the different properties that are related to the design pattern. They are:
 - The Pattern Name (which is mandatory) is used as a handle to describe a design problem, its solutions, and the consequences in a few words.
 - The Owner Name which is the name of the person who first introduced the design pattern.
 - The Author Name is the name of the person who is designing the design pattern.
 - The Design Pattern Version allows the design pattern to be identified specifically since it is possible to have different versions of the pattern.
 - Intent is a short statement which answers the following questions:
 - What is this design pattern supposed to do?
 - What is the rationale of this design pattern?
 - What are the design issues of problems that this pattern addresses?

- Motivation is the scenario that illustrates the design problem and how the class and object structure in the pattern solve the problem.
 - Applicability answers the following questions:
 - In what situations may the design patterns be applied?
 - What are some examples of poor designs that the pattern can address?
 - How is it possible to recognize these situations?
 - Known Uses provides examples of the design pattern when found in practical applications and in real-world systems
 - Related Patterns are a list of patterns which are closely related to the pattern
 - Consequences pertain to the results and the trade-offs that have to be made when applying the design pattern. It is important that design alternatives, costs and benefits, and other implementation issues be addressed when applying the pattern
 - Language is if the design pattern is created for a specific application. In these cases, the language of the application could be mentioned as an attribute. A graphical output tool could also be used with this attribute to display the correct diagram for the design pattern
- Structural Attributes represent the static view of the pattern which shows the elements of the pattern in terms of classes and the relationship between these elements. It is important to note that the schema is designed so that both the template of the design pattern and the particular instance of the design pattern are taken into consideration. The main elements of the structural attributes are as follows:
 - Classes Elements are a list of all the participant class elements. The DPDL will be able to handle all the possible instances of the particular design pattern.
 - Sub group elements help in making a template of the design pattern, enabling the handling of variations of the design pattern in a clear and concise manner.
 - Class elements are used to describe classes in the design pattern and all the details about the classes of a design pattern are defined in this class element.
 - Operations Elements are containers with all the functions and operations in the design pattern. There are two sub elements:
 - Sub Group Op Element handles the templates for the design patterns. When defining a particular instance of a pattern, functions may be

- described in a single Sub Group Op, enabling the creation of a simple, extensible, and easily understandable hierarchy for grouping the operations. This element is optional because the instance of the design pattern can be created using DPDL without using these attributes.
- Function elements contain all the details of the actual functions.
 - Object Elements are the containers for all the objects in the design pattern. It has two main elements:
 - Sub Group Ob Element is used in the design pattern template. Its attributes are optional as the instance of the design pattern can be created in DPDL without using these attributes.
 - Object Element is a defined single object where the attributes of the particular object are described in the object element.
 - Relationship Elements are the relationships between the classes. This is important especially for the structure of the design pattern. The relationships basically tell how the different classes interact with each other. This element includes:
 - Sub Group R Element which is included in the design pattern template. This is optional because the instance of the design pattern could be created in DPDL without the use of the attribute.
 - Relationship Element is the individual unique relationship between two classes and is described in the relation element of the scheme. This describes the relationship accurately, completely, simply, and as easily as possible.
- Behavioral Attributes are attributes which represent the dynamic view of the pattern, showing how the elements of the pattern communicate. The sub elements of the behavioral attribute are as follows:
 - Set Object Element is used for assigning a variable or an object to another object, like typecasting one object into another object or object type, which is common when using different design patterns.
 - Call Element is the most used behavioral element and is used when capturing a function which is invoked in a design pattern.
 - Create Element is used for depicting the creation of some objects in the design pattern, defining the creation properties.
 - Loop Element pertains to all the loops that are part of the design pattern

- Condition Element is used to manage the sequencing of the design patterns, where conditions would be used to modify the sequencing
- In each of the behavioral attributes are special common attributes and they are:
 - In Group Id is used when an action is dependent on another structural part where the attribute identifies the independent group
 - For Each is used to identify which structural part of the behavioral attribute is repeated
 - In Each is used to handle the situation when the user wants to describe a particular behavioral action which is present in all the classes of the subgroup.

In order to go into further detail and specifications, the complete DPDL schema is made available at (DPDL, 2013). A prototype has been developed in order to validate the DPDL where two Open-source tools were extended to represent the structural and behavioral views of patterns. DPDL is a text-based description language and although it has graphical support, the descriptions involve a lot of technical information which would not make it suitable for micro-business owners without technical backgrounds.

(El Boussaidi & Mili, 2012) propose patterns based on the problems they solve, prioritizing comprehensibility and applicability. Although the description of the proposed patterns is informal at best, it is the explicit representation of the problem solved by a pattern which is important. The proposed patterns consist of triples $\langle MP, MS, T \rangle$ where MP is a model of the problem solved by the pattern, MS is a model of the solution proposed by the pattern, and T is a model transformation of an instance of the problem into an instance of the solution. Proper use within a development context requires that the proposed patterns (i) must be understood, (ii) applicable or relevant to the problem at hand, and (iii) faithfully applied.

In a similar fashion, (Hsueh et al., 2008) propose a systematic and objective approach to verify pattern design, where the design pattern indicates the problem to be solved and the solution. Their approach aims to provide the following benefits:

- An evaluation approach which could help pattern developers check if a design pattern is properly designed
- A quantitative method which could measure the effectiveness of the quality improvement of a design pattern

In order to realize such benefits, the design patterns are characterized as a tuple $\{I_F, I_N, Q, S_F, S_N, T\}$ where:

- I_F is a functional requirement intent which describes what the pattern does. This is a textual description
- I_N is a non-functional requirement intent which describes how well the pattern can contribute to the quality attributes like reusability, maintenance, or extensibility. This is a textual description
- Q is quality focus which represents the quality focus from I_F to I_N
- S_F is the functional requirement structure which represents the structure that can realize the functional requirement intent or I_F
- S_N refers to the non-functional requirement structure and it represents the structural model that can enhance the non-functional requirement intent I_N
- T is the transformation which represents the transformation function from S_F to S_N

In order to verify the consistency between the intent and the structure of the design patterns, an object-oriented quality model is used. The idea is that if the intent of the pattern maps to an object-oriented property, then its structure should support that property. The object-oriented property of a structure could be evaluated using object-oriented metrics such as the coupling factor (COF) (Brito & Abreu, 1995).

Both (El Boussaidi & Mili, 2012) and (Hsueh et al., 2008) have problem-solving-oriented patterns. However, such patterns have not been made to solve problems that are focused on the micro-business domain. The “triples” or “tuples” may not be the right factors to determine whether a pattern is applicable in a micro-business problem.

2.1.5 Research proposals related to representing infrastructure

Fifth, this subsection enumerates a variety of proposals which involve the representation of infrastructure requirements in relation to software in several contexts, from large-scale enterprise systems to smaller systems. After enumerating these research proposals, an explanation is provided on why this group of proposals would not be totally applicable to the domain of micro-businesses.

(Boer et al., 2012) propose RadioMarché, a voice- and web-based market information system aimed at stimulating agricultural trade in Sahel countries. They represented

infrastructure elements such as the internet, power supplies, local radio, and the local population (intrinsically representing literacy, the ability of the local population to read and write). Two subsequent works related to the RadioMarché proposal were also reviewed. (Gyan et al., 2013) represent the internet as infrastructure in rural Africa and (Bon et al., 2013) represent radio stations as infrastructure in Mali.

(Wouters et al., 2009) propose a patient monitoring system which would support home-based health care in South African rural communities using Unstructured Supplementary Service Data (USSD) technology. The infrastructure represented in their work includes the internet, human resources, and USSD facilities.

(Bhimani et al., 2013) demonstrate different methods for creating and curating collaborative content in and over remote locations. Their infrastructure representations involve live cameras, dedicated networks, projectors, cloud servers, and human resources, particularly the users.

The work of (Izumi et al., 2007) demonstrated how a vehicle probe system (which they also refer to as the Floating Car Data “FCD” System) functions in its environment. The infrastructure representations involve a global positioning system (GPS), communication platforms, and machinery in detail such as car lights, car wipers, and speedometers.

(Supakkul et al., 2010) modeled infrastructure related to NFRs in a credit card theft case. The infrastructure representations involved human resources, telecommunications, the internet, and hardware in detail such as cashiers, laptops, desktops, and corporate servers.

(Chung et al., 2011) propose Goal-Oriented Software Architecting (GOSA) which they applied in a specific case, the London Ambulance Service computer-aided dispatch system. The infrastructure representations involved tracking devices for rescue vehicles (ambulances and helicopters), telecommunications networks, and human resources (particularly focusing on their roles in the system).

In a subsequent work, (Chung et al., 2013) use goals, particularly SIGs, to model large-scale infrastructure in a contactless smartcard system which automates the ticketing on all channels of public transport in Victoria, the most populated state in Australia. The infrastructure representations involved mobile phones, kiosks, personal computers, vehicles (mainly public transportation such as trains), data centers, and human resources.

When representing infrastructure and software (systems and components) altogether (their architecture), The Open Group Architecture Framework (TOGAF) (The Open Group, 2009) must be mentioned as related literature since it is currently being considered as the standard way of developing and deploying modern IT systems in enterprises (Dietz & Hoogervorst, 2011). TOGAF models infrastructure on an enterprise level, mainly representing the interaction of software systems with its environment.

Although there are clear benefits of using TOGAF in large enterprises, its use in SMEs is still questionable (more so for micro-businesses) (Alm & Wißotzki, 2013). The use of TOGAF is accompanied by corresponding manpower, maintenance, and training costs, which may not be apt for the majority of micro-businesses.

These seven ways of representing infrastructure and software systems are very context-specific. The advantage of using these pre-built infrastructure requirements is that they are applicable if the case is almost exactly the same as the case from which the infrastructure requirement is built from. However, the infrastructure requirements are a disadvantage when taking the entire domain of micro-businesses into context. Unguided or “one-size-fits-all” requirements representation approaches (like TOGAF) would not be fitting (Quispe et al., 2010; Aranda et al., 2007; Bürsner & Merten, 2010). In addition, given the limited technical exposure and limited resources of micro-business stakeholders (Laukkanen et al., 2007; Buonanno et al., 2005), a proposal involving a step-by-step, “lightweight but effective” (Ambler, 2002) infrastructure requirements representation technique which micro-business owners and developers could adapt and use in their software projects is needed.

From our review of related work, we have **not** found proposals which have (requirements) patterns that have all of the following: involving (micro-)business processes, based on requirements, represents infrastructure, and balancing the priorities of software (component) reuse for developers and comprehensibility for micro-business owners, factors which make (requirements) patterns suitable specifically for the domain of software systems for micro-business.

2.2. Related Evaluations of Research Proposals put into Practice

(Requirements) patterns for micro-businesses can be technically relevant for developers and comprehensible for micro-business owners in practice. We identified related works that

evaluate research proposals using case studies and Action Research. In addition, there is also a subsection that covers evaluations specific to comprehensibility in practice.

2.2.1. Evaluations of research proposals using case studies

The following are evaluations of research proposals using case studies. They are relevant because they provide an up-close, in-depth, examination of the research proposal being put into practice.

As mentioned in the previous subsection related to component-based software engineering, a result of combining component-based software development and reusing software architecture has led to the notion of software product lines (Bosch, 2000).

(van Gorp et al., 2010) compared four software projects using case studies, comparing both integration-oriented SPLs and open source projects. Although all of the case studies were successful, they wanted to understand which practices were suitable in which context. Based on their findings, they found out that large-scale open software development can be performed successfully using practices that differ substantially from SPL practices.

The trends in the case studies suggest that several software organizations find themselves a part of an increasingly large ecosystem that develops the software they productize and consequently, a more compositional style of development would be more appropriate. The evaluation that they performed is considered an important step towards empirical evaluation methods for software processes.

(Zhao & Zou, 2011) evaluated the use of clustering algorithms to derive software modular structures from business processes using a case study. Business processes describe the operations of a business in an organization and are capable of capturing business requirements. These business processes are composed of a set of interrelated tasks which are joined together by data flow and control flow constructs. The data flows describe the inputs into tasks and outputs generated from the tasks. Data items are abstract representations of information flowing through the tasks. Control flow constructs specify the order of the execution of tasks such as being sequential, alternative, or iterative.

A software modular structure represents the structure of a business application and represents the distribution of functionality among software components. Software modular

structures are widely used to bridge the gap between business requirements and business applications. Software modular structure refers to the logical view of software architecture and it represents the structure of a business application using software components, the interactions between and among the software components otherwise known as the connectors, and the constraints on the components and the connectors.

Within the software modular structure are components which capture particular functionalities. The connectors of the components define the control and the data transitions among the components. The constraints specify the properties of the components and the connectors and how they are combined.

The problem with business processes and software modular structures is that design approaches rely on the craftsmanship of the software architects which means in large scale business applications which need to satisfy thousands of business requirements, a manual design approach would be inefficient and would lead to inconsistency between business requirements and business applications.

Hence, (Zhao & Zou, 2011) propose an approach which consists of clustering algorithms that automatically generate software modular structures from business processes. The clustering algorithms analyze dependencies among data and tasks captured in a business process and group the strongly dependent tasks and data into a software component. There are two major steps in their approach:

- Derive the software components from business processes to fulfill the functional requirements
- Apply the software architectural styles and design patterns to address the quality requirements

(Zhao & Zou, 2011) conducted a case study to evaluate the effectiveness of their proposed approach. There are five steps in their case study:

- Generate software modular structures from business processes of the subject business systems
- Recover the as-implemented software modular structures from various sources such as documentations or source code of the subject business systems

- Compare the generated software modular structures with the as-implemented software modular structures to assess the authoritativeness of the generated software modular structures
- Analyze the extent to which the generated software modular structures are affected by the changes in the business processes in order to examine the stability of the proposed approach
- Evaluate the modularity of the generated software modular structures

This five-step approach was used on two large-scale business systems, the IBM WebSphere Commerce (WSC) server (IBM WSC, 2012) and Opentaps (Opentaps, 2013). The result of the experiment was successful, showing that based on the proposal, meaningful software modular structures with high modularity can be derived from business processes using the clustering algorithms.

(Crnkovic & Larsson, 2002) evaluated the challenges arising from evolving component-based systems in a case study involving the ABB Advant control system (ABB, 2013), an industrial control system. According to the study, the success of this system in the market is primarily due to its appropriate functionality and quality. The success in the development, maintenance, and continued improvement of the system is a result of careful architecture design, where the main goal is component reuse.

There are several advantages when the goal of design is component reuse. In order to reap the benefits, there must be a systematic approach in design planning, extensive development, support for more complex maintenance processes, and more consideration given to components. Also, if a more reusable component is to be developed, then there would be a more complex development process and more required support from the organization.

Several factors are examined in the case study, including evolving requirements and its management, architecture, and other business-related factors such as marketing issues. One of the main problems highlighted in the case study are unpredictable extra costs. ABB had to pay extra costs for a change to a Windows NT platform, which was not given sufficient consideration during the project.

Another problem that was highlighted in the case is the movement from old to new technologies, requiring the re-creation of components or the inclusion of standard components which are available in the market. According to the experience in the case study, the process

of replacing proprietary components with standard components available from third parties is unavoidable and it is important to have a plan for migrating from old components to newer ones. In a related work, (Crnkovic & Larsson, 2000) specifically present the successful implementation of the ABB industrial process component-based system in a case study as well.

(Ampatzoglou & Chatzigeorgiou, 2007) evaluated the use of object-oriented (OO) design patterns in game development. The patterns were applied in a game development case study. Two Open-source games were studied: namely Cannon Smash Version 0.6.6 and Ice Hockey Manager Version 0.2.

The results of the study of the two games showed that using the patterns reduces complexity, decreases decoupling (of software components), and increases cohesion of the overall (game) software. However, the study also showed that the size of the project increased when the design patterns were used. In any case, with the evolving nature of games, (Ampatzoglou & Chatzigeorgiou, 2007) still believe that the appropriate employment of design patterns should continue to be encouraged in the programming and development of games.

Evaluating these research proposals using case studies provided us insight on how to observe the implementation of software projects in the micro-business domain in practice. However, merely observing how micro-businesses operate in practice may not be enough to complete the objectives of our thesis since **we may have to influence the software developers or micro-business owners in our work**. Hence, we look at Action Research in the next subsection.

2.2.2. Evaluations of research proposals using Action Research

In case studies or field experiments, the influence of the researcher is at a minimum or even non-existent. Action Research is when a research proposal is applied into practice and the researchers are actively participating. Action Research has a lot of variants (Goldkuhl, 2008) (Goldkuhl, 2012) (Bilandzic and Venable, 2011) which is why we detail our Action Research step-by-step in the later chapters. Based on an initial survey, the use of Action Research has been increasing in the field of software engineering (dos Santos & Travassos, 2009), despite representing only a small fraction of the studies being conducted in software engineering. In this section, we enumerate related evaluations using Action Research.

(Grant & Ngwenyama, 2002) evaluated the usefulness of a manufacturing information system development (ISD) methodology at a manufacturing technology company using Action Research. The Action Research provided the theoretical framework for the intervention of the ISD into the organization and the Action Research also guided the investigation and critical analysis of the problem situation. The Action Research consisted of five stages, namely: diagnosis, action plan, action taken, evaluation, and learning.

The Action Research methodology and the ISD methodology were used to solve five technical and organizational problems which were identified in the engineering release function of the company. The five problems were: lack of shared understanding, lack of communication and coordination among departments, insufficient throughput and long cycle time, inconsistencies in documentation, and duplication of effort.

Based on the study, it appears that the application of an ISD methodology would depend on three factors: the original design of the methodology, the background knowledge and motivation of those applying the methodology, and the organizational climate and culture

Given this, the same ISD methodology could lead to several different outcomes based on whether developers apply an ISD with different orientations and under varying organizational circumstances. By using Action Research, it was established that the ISD methodology could be successful in the case organization. This means that success in other settings is not guaranteed.

The study found that the culture of the organization, its power structure, climate, and management philosophies were factors that contributed to the success of the ISD methodology. The company had a history of using planning approaches and several of the executives served in the military, having strict discipline. The results of the Action Research showed reductions in product cycle time from 12 to 3 days, work-in-process reductions by approximately 75%, and rework reductions by approximately 30% when the ISD methodology was applied.

It is important to note that even if the results of the Action Research were unsuccessful (that the ISD methodology did not produce as astounding results), results from Action Research could still be useful for two reasons: important lessons are learned from failed projects if the results are reported to the research community through scholarly journals (Lyytinen & Robey, 1999) *and* the success of Action Research is not only measured by its

practical success but also by its ability to add to the stock of knowledge of the research community.

(Roost et al., 2013) evaluated business architecture development by students working in socially networked groups using Action Research. The focus of the study was on the social interactions of one group of medical technology students and another group of IT students.

In this case, Action Research is described as a collaboration between a “client system” and a “change agent” (which is the opposite of a traditional “observer”). In this Action Research study, the client system is a concrete enterprise, a medical laboratory with a laboratory information management system. The change agents were composed of medical technology students forming one group and IT students forming the other group. In this Action Research study, it is noted that every agent in the context of the enterprise was seen as a change agent.

The students of medical technology played the dual roles of core business process owners and business analysts who are supposed to be knowledgeable of the problem at hand. The IT students played the roles of business designers who are knowledgeable on IT-enabled possible solutions for the problem. The relationships between and within these two groups was managed using Google Sites social software.

The lessons learned from the Action Research study are as follows:

- The approach that they wanted to test (social self-development (SSD) of evolutionary information systems) showed to be applicable in collaborative learning contexts that are similar to the student project although richer supporting infrastructure is required.
- The use of the Google Sites social software based on blogs has shown to be useful in the tested contexts.
- In similarity to pair programming, it was also possible to perform strategic analysis and design effectively in pairs composed of a business person (like a student of medical technology) and an IT person (like an IT student).
- If continuous community-based modeling activities involving the SSD is removed, the approach would not work.

(Millman & El-Gohary, 2011) evaluated how the marketing practices of a micro-business can take advantage of digital media using Action Research. The Action Research was made to answer two main questions: what are the factors that can influence the innovative activities

of micro-businesses, especially marketing practices? *and* how can a small firm use new digital media to improve their marketing practice?

Using Action Research, a qualitative approach was applied. The review of related literature allowed an in-depth view in diagnosing the problems and the issues which are encountered by the micro-business practitioners. In order to illustrate the process of actions taken, a longitudinal study is conducted.

The results of the Action Research showed that in practice, much of the marketing activities in micro-businesses are driven by incremental innovation, emphasizing that integrating new technologies (such as digital media) in marketing requires that marketers take an active managerial role beyond their traditional role.

(Lee, 2002) evaluated the relationship between the national IT infrastructure (of Korea) and the success of a digital library using Action Research. The Action Research consisted of five stages: diagnosing, action planning, action taking, evaluation, specifying learning.

Surveys, evaluating user satisfaction in the newly established national digital library, were made part of the Action Research. As a result of the Action Research and the additional surveys, the following observations were made:

- There were times when the system was slow due to the large number of users logged on to the system.
- Usage of a bulletin board system was low because of the lack of trust of users for the system. Koreans are distrustful of authority because of a long history of oppressive military dictatorship.
- A lot of users were experiencing difficulty obtaining the full texts of requested articles (as the survey results have shown).

From the Action Research, (Lee, 2002) recommended the following critical success factors for the digital library to succeed:

- People must be able to find some useful content in the digital library which may or may not be available in a physical library.
- The digital library must have an interface which is easy to use so that people would use it.

- The digital library must be fast enough so that people would use it.

(dos Santos & Travassos, 2011) evaluated subjective decision-making made by software developers using Action Research. The context of the Action Research was in the refactoring of source code to improve source code quality. The practice of Action Research does not always go as planned and of course, its application comes with a variety of challenges. The researchers noted several difficulties in using Action Research as a methodology during their study. These difficulties are as follows:

- Guidelines are lacking on how to use Action Research in software engineering as evidenced by the absence of technical papers on the theme ((dos Santos & Travassos, 2011) conducted a systematic literature review as part of the work).
- There is difficulty in perceiving the difference between an Action Research study and a case study when selecting an evaluation method. The main difference would be that if there is more emphasis on observation then a case study would be apt. If there is more emphasis on intending to change organizational practices then Action Research would be more apt. Determining whether a case study or Action Research should be done is difficult to assess, especially when the decision has to be made in the middle of planning and diagnosing a problem.
- Collecting data in Action Research is difficult because it is difficult to identify what data should be collected. There would obviously be unplanned events that happen during Action Research studies. Although Action Research accommodates these unplanned events, registering and justifying the events in the data has to be done by the researcher.
- Conditions of collaborations in Action Research would not always be met. It would be ideal if professionals would always genuinely collaborate with the goals of the researcher and vice versa but this is not always the case. Hence, diplomatic abilities of the researcher (and professional) are needed in order to benefit fully from Action Research. Communication plays an important role in Action Research because it is the conducting medium that enables the collaboration among the participants of the Action Research study.
- There are also ethical issues to address when conducting Action Research. There are two main ethical issues:
 - It is difficult for a researcher to fulfill scientific goals such as publishing in conferences and journals while at the same trying to address the real problem at hand, the reason why Action Research is being conducted.

- Some organizations would participate in Action Research but would not be compelled to publish their data. The freedom of consent to publish results will not always be achieved. There are organizations that wish to protect their privacy.

These five evaluations of research proposals using Action Research have provided us a lot of insight on how we would conduct the evaluations of our research proposal in practice. More details of our Action Research are detailed step-by-step in the later chapters.

2.2.3. Evaluations of comprehensibility in industry

For the remainder of the review of related work, we enumerate evaluations of comprehensibility in industry. Evaluations of comprehensibility are important to our research proposal because the ease of communicating with micro-business owners is vital to the success of our research proposal when applied in practice.

A recent study was conducted by (Weitlaner et al., 2013) regarding the comprehensibility of process models. In the first part of the study, a pen and paper experiment which involved 43 participants was performed, where four process archetypes were used. The results of the experiment showed that formal business process management (BPM) is still not yet fully accepted and considered as useful in industry. This is because flowcharts are mostly used for designing processes.

The second part of the study involved a survey of 77 employees regarding the comprehensibility of BPM languages. The survey examined to what extent process models were understood by individuals.

The findings of the study were that comic representation storyboard design was intuitive and more easily understood. BPMN and UML were also comprehensible but not as comprehensible as the storyboards. Hence, from the study, the recommendation made by (Weitlaner et al., 2013) is to use storyboards in field BPM.

(Reinhartz-Berger et al., 2011) evaluated the comprehension of variability issues in UML in the field of software product line engineering (SPLE). SPLE deals with two main activities: domain engineering where a family of software products called product lines are analyzed,

designed and implemented, and application engineering where particular software products and applications are customized and developed.

An important topic in SPLE is that a software artifact could be (re-)used in several different contexts for the purpose of increasing productivity. There are several proposals which have this goal in mind and are referred to as variability modeling methods. A significant proportion of these proposals are expressed in UML.

The UML-based methods would normally introduce profiles in order to specify a set of required and optional elements, while identifying dependencies between the elements, and also modeling the variation points and possible variants. The comprehensibility and utilization of UML in these methods is evaluated.

An evaluation framework is created which allows the comparison of different aspects of variability specification. Variability specification is important because it helps create valid applications in specific domains. A specific UML-based method (for variability specification) was chosen for evaluation which was the Application-based Domain Modeling (ADOM). ADOM is based on five stereotypes where the range of elements in a product artifact can be classified as the same element in the core asset. Each element in the core asset may be defined as a variation point.

The evaluation framework examined how advanced information systems students understood and utilized the (ADOM) model. The results of the evaluation showed that the different means for specifying variability were only utilized and understood to a limited extent. Also, the variation points were also the least comprehensible among the variability specification means. One possible solution to improving the adaptability of variability languages is the use of the Common Variability Language (CVL) (Haugen et al., 2008).

A quantitative evaluation of different design alternatives expressed in UML would aid in understanding system performance and would also aid in the design of complex systems. (Pokozy-Korenblat et al., 2004) propose a tool called BioSpi which provides quantitative analysis of the performance of UML specifications without the complexities of formal descriptions. The BioSpi tool allows users to maintain a full record of the evolution of each process in the system. Such a record specifies all the communications, including the processes involved, the time and the channel where they occurred, the communication partner, and the process which results from each communication.

The feasibility of the approach is demonstrated in an application of the tool in a web-based micro-business case study. The steps of the evaluation process are as follows:

- UML activity, sequence, and deployment diagrams are enriched with supplementary quantitative parameters.
- The simulation is performed using the BioSpi tool.
- Based on the simulation, performance curves are derived which characterize the influence of the given quantitative measures on the behavior of the entire system.

Based on the quantitative data, the performance analysis using the BioSpi tool resulted in the following findings:

- The global system showed sensitivity to the number of available sellers (in the micro-business).
- Authentication time depended on the number of parallel requests and on the complexity of the cryptographic algorithms used for the secure transmission of data.
- If the encryption took more than half a second to run then the most influential factor is the time needed to serve concurrent requests.

Aside from comprehensibility studies on business process models and UML, evaluating comprehensibility in different software notations, specifically Use Case and Tropos, has been done by (Hadar et al., 2013). The objective of their work is to compare the comprehensibility of requirements models which are expressed in different but comparable modeling approaches from the perspective of a requirements analyst.

The comprehensibility of a requirements model is measured in the context of three types of tasks which are: the mapping between textual descriptions and model elements, the reading and understanding of the model, irrespective of the original textual description, and the modification of the model.

The experimental evaluation is conducted within a family of controlled experiments for the purpose of comparing Use Case, a scenario-based method, and Tropos, which exploits goal modeling. Three runs of the experiment are performed involving 79 information systems students. The data for each experiment was analyzed separately followed by a meta-analysis.

The results of the experiment show that Tropos models are more comprehensible with respect to the three types of requirements analysis tasks but take more time to make as compared to the Use Case models. The ability to measure the comprehensibility of models based on a series of controlled experiments shows that measuring comprehensibility of models is feasible, despite the fact that both Tropos and Use Case are different modeling approaches.

The work of (Hadar et al., 2013) is only one of many rare studies. It is difficult to find empirical studies related to the comprehensibility of requirements models, especially if the languages and notations belong to different modeling approaches.

Based on our review of related literature, we have found that the use of Softgoal Interdependency Graphs SIGs as proposed by (Chung et al., 2000) could be a vital piece in our proposal. However, we have not found any comprehensibility studies where SIGs are applied in practice. We provide a comprehensibility study of SIGs in practice in the later chapters and also consider this as one of our unique contributions to literature.

3. CONCLUSIONS

In this chapter, we discussed several research and studies related to our proposal in terms of their field of study: business processes, patterns, requirements, software components, reuse, comprehensibility, representing infrastructure, evaluations with case studies, Action Research, and evaluations on comprehensibility. We have also discussed both the weak and strong points of each proposal in relation to the proposal we will present in this thesis. A checklist and table of the comparisons we have made are shown in Table II.1 and Table II.2.

We build on the strong points from other proposals and address the weaknesses of other proposals as we discuss the development of the thesis in the subsequent chapters. In the next chapter, we will discuss the fundamental concepts that are related to our proposal.

Table II.1 Checklist of Related Literature

Proposal and Author(s)	business process	requirements	patterns	software components	reuse	comprehensibility	representing infrastructure	evaluations with case studies	evaluations with action research	evaluation of comprehensibility
Modeling by Example (MbE) by Kalenborn, 2010	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Business Process Patterns (BPP) by Barros, 2007	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Resources, Events, Agents Model (REA) by Hruby, 2006	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Service-based cooperation patterns (SBCPs) by Boukheduoma et al., 2013	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Document-based patterns by Glushko & Mcrath, 2002, 2008	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Domain-specific patterns for mobile service businesses by Aleksy & Stieger, 2011	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PattCar by Seruca & Loucopoulos, 2003	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Software Requirements Patterns (SRPs) by Franch et al., 2010 and by Mendez-Bonilla et al., 2008	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Software Requirements Patterns based on User-Trust by Hoffman et al., 2012	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Security Requirements Patterns by Riaz & Williams, 2012	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Privacy Modeling Capabilities by Peixoto & Silva, 2018	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Non-Functional Requirements (NFR) Patterns by Supakkul et al., 2010	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Patterns in Component-Based Software Engineering by Crncovic et al., 2002	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Generic process framework for component-based development by Kouroushfar et al., 2009	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Controller Patterns by Stepan & Lau, 2012	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Specification Structure by Paludo et al., 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Connectors by Elizondo & Lau, 2010	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Attribute-driven framework for selecting COTS components and connectors by Bhuta et al., 2007	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Patterns for Unified Modeling Language (UML) profiles by (Lakhal et al., 2013)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Structural pattern specification (SPS) and interaction pattern specifications (IPSS) by France et al., 2004	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Business process modeling notation (BPMN) patterns for extract-transform-load (ETL) systems by Oliveira & Belo, 2012	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Workflow Activity Patterns (WAPs) in BPMN by Thom et al., 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Design pattern definition language (DPDL) by Khwaja & Alshayeb, 2013	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Patterns with triples <MP, MS, T> by El Boussaidi & Mili, 2012	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Patterns with tuples (IF, IN, Q, SF, SN, T) by Hsueh et al., 2008	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7 ways of representing infrastructure proposed by [1] Boer et al., 2012, [2] Wouters et al., 2009, [3] Bhimani et al., 2013, [4] Izumi et al., 2007, [5] Supakkul et al., 2010, [6] Chung et al., 2011, 2013, [7] The Open Group, 2009	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comparison of Software Product Lines (SPLs) by van Gorp et al., 2010	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Evaluated the use of clustering algorithms to derive software modular structures from business processes by Zhao & Zou, 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Evaluating the challenges arising from evolving component-based systems by Crnkovic & Larsson, 2002	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Evaluation of the use of object-oriented (OO) design patterns in game development by Ampatzoglou & Chatzigeorgiou, 2007	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Evaluation of the usefulness of a manufacturing information system (ISD) development methodology by Grant & Ngwenyama, 2002	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Evaluation of business architecture development by students working in socially networked groups by Roost et al., 2013	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Evaluation on how the marketing practices of a micro-business can take advantage of digital media by Millman & El-Gohary, 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Evaluation of the relationship between the national IT infrastructure (of Korea) and the success of a digital library by Lee, 2002	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Evaluation of subjective decision-making made by software developers by dos Santos & Travassos, 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Comprehensibility of process models by Weitlaner et al., 2013	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Comprehensibility of variability issues in UML in the field of software product line engineering (SPLE) by Reinhartz-Berger et al., 2011	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
BioSpi provides quantitative analysis of the performance of UML specifications without formal descriptions by Pokozy-Korenblat et al., 2004	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Evaluation of comprehensibility in Use Case and Tropes by Hadar et al., 2013	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Table II.2 Table of Related Literature

Proposal and Author(s)	Field of Study	Brief Description	Strong Points	Weak Points
Modeling by Example (MbE) by Kalenborn, 2010	business processes	Requirements are done before the project even starts, MbE is designed for bidding in projects	Practical and straightforward, fast and quick to implement, uses a lot of visualization,	Lack of technical detail, very little dialogue with the final customer; lack of alignment with goals
Business Process Patterns (BPP) by Barros, 2007	business processes, patterns	Encapsulates high business logic to improve business processes or applied to develop software	Patterns are based on empirical data,	lack of connection with the software components that would be needed to implement the software solutions
Resources, Events, Agents Model (REA) by Hruby, 2006	business processes	used to develop business-related (software) applications by searching for business objects and related modeling elements	processes are based on and used in day-to-day business processes, also used in development of software applications	lack of resource management for applications in micro-businesses, there is no mention of reuse and reduction of software development timelines
Service-based cooperation patterns (SBCPs) by Boukheduoma et al., 2013	business processes, patterns	used for recurring service-based inter-organizational workflows (IOWF) that meet certain service-oriented architecture (SOA) paradigms	ability to deal with IOWFs flexibly	no mention of resource management for micro-businesses such as software reuse or reduction of software development timelines
Document-based patterns by Glushko & Mcrath, 2002, 2008	business processes, patterns	document exchange is the mother of all patterns, e.g. "document engineering"	promote reuse, which results in reduced maintenance, better consistency, and standardization	no mention of improving communication with micro-business owners, there is still a lot of technical jargon in the documents which may not be comprehensible for the micro-business owners
Domain-specific patterns for mobile service businesses by Aleksy & Stieger, 2011	business processes, patterns	four mobile service patterns are proposed to aid in solving domain-specific problems	based on two real-world cases and can be directly applied in the specific domain	there were no mentions of instantiating patterns, given the variety of micro-business cases, applications would be limited
PattCar by Seruca & Loucopoulos, 2003	business processes, patterns	a systematic approach for collecting and analyzing business domain knowledge in order to support the capture and development of business patterns	based on real-world cases, takes into consideration the idea of reuse	does not specify the reuse of the software
Software Requirements Patterns (SRPs) by Franch et al., 2010 and by Mendez-Bonilla et al., 2008	requirements, patterns	aims to reuse software requirements, mostly non-functional requirements and those which involve commercial-off-the-shelf (COTS) software	based on 7 real-world software requirement specifications and are applicable in the real-world	a lot of technical jargon which may not be suitable for the comprehension of micro-business owners
Software Requirements Patterns based on User-Trust by Hoffman et al., 2012	requirements, patterns	Antecedents that build trust are collected and then they are developed into a set of requirements patterns	they are applicable to specific use cases	there is no mention of resource management for micro-business software projects such as improving timelines
Security Requirements Patterns by Riaz & Williams, 2012	requirements, patterns	Security requirements patterns capture common security requirements, document the context in which a requirement manifests itself, and describe the trade-offs involved.	they are applicable to specific use cases	there is no mention of resource management for micro-business software projects such as improving timelines
Privacy Modeling Capabilities by Peixoto & Silva, 2018	requirements, patterns	framework for for privacy modeling capabilities that must be addressed by requirements modeling languages to better support privacy specification	they are applicable to specific use cases	there is no mention of resource management for micro-business software projects such as improving timelines
Non-Functional Requirements (NFR) Patterns by Supakkul et al., 2010	requirements, patterns	capture and reuse NFR patterns in business-specific cases with the help of NFR visualizations and representations	practical when it comes to relating and diagramming the relationships between NFRs and operationalizations	does not take into account optimizing resources for micro-businesses such as reusing software components and reducing timelines for micro-business software projects
Patterns in Component-Based Software Engineering by Crncovic et al., 2002	software components, patterns	reusable units are identified as pre-existing components during software design and then in development, components are developed based on the patterns	the idea of reusing software components in CBSE is a great idea, taking into account improvements in resource management for several projects, including those of micro-businesses	there is still lack of formalization and specification of patterns in CBSE since there are several disciplines involved, moreover, there have been no studies focused on reusing patterns in CBSE particularly for micro-business software projects
Generic process framework for component-based development by Kouroshfar et al., 2009	software components, patterns	based on 7 CBSE approaches, this proposal identifies four phases: analysis, design, provision, and release	provides a strong framework between software development and requirements	the proposal must be further refined for use in the micro-business domain
Controller Patterns by Stepan & Lau, 2012	software components, patterns	patterns for component-based development for control software for reactive systems (systems which react to their environment)	provides a strong framework between software components and requirements	a lot of technical jargon in the proposal may not be suitable for the comprehension of those in the micro-business domain
Component Specification Structure by Paludo et al., 2011	software components, patterns	the specification documents, retrieves, and captures composition functionalities of the components in order to achieve software reuse	the specification makes it easier to reuse software components by making them more searchable in a repository or a database	specification is still very heavy on technical jargon and may not be suitable for eliciting requirements for micro-business owners who have limited technical exposure
Component Connectors by Elizondo & Lau, 2010	software components, reuse	a catalog of component connectors, describing the connectors as the "glue" which piece together components in CBSE	useful for the software architecture and design of software systems	component connectors are more oriented to larger software systems and not to micro-business software systems
Attribute-driven framework for selecting COTS components and connectors by Bhuta et al., 2007	software components, reuse	attributes are used in a framework in order to select the COTS components and connectors	focused on minimizing resource expenditure on software projects through proper reuse of software components and their connectors	proposal is meant for large software systems and not for micro-businesses which are way less complex
Patterns for Unified Modeling Language (UML) profiles by Lakhali et al., 2013	comprehensibility, patterns	UML profiles are identified based on their interface, solutions, category, and relationships	UML notations are made more comprehensible to the users	UML profiles are not oriented towards micro-businesses
Structural pattern specification (SPS) and interaction pattern specifications (IPSS) by France et al., 2004	comprehensibility, patterns	UML patterns are specified based on their behavior, structure, and class	The patterns in UML format appeared to be comprehensible to graduate students when applied	UML per se may not be the most suitable language for the comprehension of micro-business users without technical backgrounds
Business process modeling notation (BPMN) patterns for extract-transform-load (ETL) systems by Oliveira & Belo, 2012	comprehensibility, patterns	ETL patterns are modeled using BPMN which allow the creation of pre-established specifications	BPMN is better for business users	ETL patterns could be useful but not always in the context of micro-business software projects

Workflow Activity Patterns (WAPs) in BPMN by Thom et al., 2011	comprehensibility, patterns	WAPs are based on 200 real-world workflows	conceptual framework could be used for building workflow patterns in the micro-business domain	the patterns have to be more lightweight for the domain of micro-business owners
Design pattern definition language (DPDL) by Khwaja & Alshayeb, 2013	comprehensibility, patterns	DPDLs are used to represent software design patterns and have three main parts: attributes, structure, and behavior	DPDL has graphical and diagram support	text-heavy descriptions would be difficult to use for micro-business cases
Patterns with triples <MP, MS, T> by El Boussaidi & Mili, 2012	comprehensibility, patterns	MP is the problem solved, MS is the solution, and T is a transformation of an instance of the problem into an instance of the solution	problem-solving-oriented patterns are directly applicable in the real world	triples may not be the right variables to determine whether a pattern is directly applicable in a micro-business-related problem
Patterns with tuples (IF, IN, Q, SF, SN, T) by Hsueh et al., 2008	comprehensibility, patterns	Problem-Solution pattern with IF as functional requirement, IN as non functional requirement, Q is quality focus, SF structure of IF, SN is structure of IN, and T is the transformation	problem-solving-oriented patterns are directly applicable in the real world	tuples may not be the right variables to determine whether a pattern is directly applicable in a micro-business-related problem
7 ways of representing infrastructure proposed by [1] Boer et al., 2012, [2] Wouters et al., 2009, [3] Bhimani et al., 2013, [4] Izumi et al., 2007, [5] Supakkul et al., 2010, [6] Chung et al., 2011, 2013, [7] The Open Group, 2009	representing infrastructure	representing infrastructure in various ways, depending on several factors such as context, can help users comprehend software systems	pre-built infrastructure requirements are applicable if a case is almost exactly the same as the case from which the infrastructure requirement is built from	a one-size-fits-all infrastructure representation may not be suitable for the variety of micro-business software projects
Comparison of Software Product Lines (SPLs) by van Gurp et al., 2010	evaluating research proposals with case studies	from four case studies, the researchers found out that large-scale open software development can be performed successfully using practices that differ substantially from SPL practices	case studies provide insight on how software projects are implemented in certain domains	case studies are not very useful when involvement of researchers is needed for proposal implementation
Evaluated the use of clustering algorithms to derive software modular structures from business processes by Zhao & Zou, 2011	evaluating research proposals with case studies	a clustering algorithm approach that automatically generates software modular structures from business processes is evaluated using a case study	case studies provide insight on how software projects are implemented in certain domains	case studies are not very useful when involvement of researchers is needed for proposal implementation
Evaluating the challenges arising from evolving component-based systems by Crnkovic & Larsson, 2002	evaluating research proposals with case studies	several factors such as management, architecture, marketing, updating technologies were evaluated using a case study	case studies provide insight on how software projects are implemented in certain domains	case studies are not very useful when involvement of researchers is needed for proposal implementation
Evaluation of the use of object-oriented (OO) design patterns in game development by Ampatzoglou & Chatzigeorgiou, 2007	evaluating research proposals with case studies	Results show that patterns reduce complexity, decrease decoupling, increases cohesion, but also increases project size	case studies provide insight on how software projects are implemented in certain domains	case studies are not very useful when involvement of researchers is needed for proposal implementation
Evaluation of the usefulness of a manufacturing information system (ISD) development methodology by Grant & Ngwenyama, 2002	evaluation using action research (AR)	AR provided the theoretical framework for the intervention of the ISD into the organization and also guided the investigation and critical analysis of the problem situation	AR provides us a lot of insight on how we would conduct the evaluations of our research proposal in practice	There are elements of subjectivity in action research if one wants to obtain objective results
Evaluation of business architecture development by students working in socially networked groups by Roost et al., 2013	evaluation using action research (AR)	the focus of the study was on the social interactions of one group of medical technology students and another group of IT students	AR provides us a lot of insight on how we would conduct the evaluations of our research proposal in practice	There are elements of subjectivity in action research if one wants to obtain objective results
Evaluation on how the marketing practices of a micro-business can take advantage of digital media by Millman & El-Gohary, 2011	evaluation using action research (AR)	investigation of marketing practices that influence innovation in micro-businesses and on how small firms can use digital media to improve their marketing	AR provides us a lot of insight on how we would conduct the evaluations of our research proposal in practice	There are elements of subjectivity in action research if one wants to obtain objective results
Evaluation of the relationship between the national IT infrastructure (of Korea) and the success of a digital library by Lee, 2002	evaluation using action research (AR)	AR included surveys and some of the results say that a lot of concurrent users resulted to capacity and load problems to the systems, not a lot of requests to the system were being completed, and that Koreans dislike authority	AR provides us a lot of insight on how we would conduct the evaluations of our research proposal in practice	There are elements of subjectivity in action research if one wants to obtain objective results
Evaluation of subjective decision-making made by software developers by dos Santos & Travassos, 2011	evaluation using action research (AR)	in the context of code refactoring, several challenges related to the subjectivity of AR are observed and documented	AR provides us a lot of insight on how we would conduct the evaluations of our research proposal in practice	There are elements of subjectivity in action research if one wants to obtain objective results
Comprehensibility of process models by Weitlaner et al., 2013	evaluation of comprehensibility in industry/practice	results of the study were that formal business process model (BPM) was still not fully accepted and that storyboards were still preferred	studies of comprehensibility provides us a lot of insight on how we would conduct the evaluations of comprehensibility in practice	there are not a lot of comprehensibility studies on requirements which means there is no real standard for evaluating it
Comprehensibility of variability issues in UML in the field of software product line engineering (SPL) by Reinhartz-Berger et al., 2011	evaluation of comprehensibility in industry/practice	evaluation framework examined how advanced IT students comprehended the Application-based Domain Modeling (ADOM) model and results were that not a lot of them comprehended it	studies of comprehensibility provides us a lot of insight on how we would conduct the evaluations of comprehensibility in practice	there are not a lot of comprehensibility studies on requirements which means there is no real standard for evaluating it
BioSpi provides quantitative analysis of the performance of UML specifications without formal descriptions by Pokozy-Korenblat et al., 2004	evaluation of comprehensibility in industry/practice	BioSpi keeps records of the users in the system and some of the initial findings were that the number of available requests became the most important factor if it took more than half a second to process	studies of comprehensibility provides us a lot of insight on how we would conduct the evaluations of comprehensibility in practice	there are not a lot of comprehensibility studies on requirements which means there is no real standard for evaluating it
Evaluation of comprehensibility in Use Case and Tropos by Hadar et al., 2013	evaluation of comprehensibility in industry/practice	79 IT students participated and the result was that Tropos was more comprehensible than Use Case	studies of comprehensibility provides us a lot of insight on how we would conduct the evaluations of comprehensibility in practice	there are not a lot of comprehensibility studies on requirements which means there is no real standard for evaluating it

CHAPTER REFERENCES

- ABB. (2013). ABB Automation Products. Last accessed on October 6, 2013 at <http://www.abb.com/>
- Akkaoui, Z. E. & Zimányi, E. (2009). Defining ETL workflows using BPMN and BPEL. In I.-Y. Song & E. Zimányi (eds.), DOLAP (pp. 41-48): ACM. ISBN: 978-1-60558-801-8
- Akkaoui, Z. E., Zimányi, E., Mazón, J.-N. & Trujillo, J. (2011). A model-driven framework for ETL process development. In I.-Y. Song, A. Cuzzocrea & K. C. Davis (eds.), DOLAP (pp. 45-52): ACM. ISBN: 978-1-4503-0963-9
- Albert, C. & Brownsword, L. (2002). Evolutionary Process for Integrating COTS-Based Systems (EPIC). CMU/SEI Technical Report CMU/SEI-2002-TR-005, 2002
url: <ftp://ftp.sei.cmu.edu/public/documents/02.reports/pdf/02tr005.pdf>
- Aleksy, M. & Stieger, B. (2011). Mobile Service Business Patterns. In *Proceedings of the IEEE 25th International Conference on Advanced Information Networking and Applications AINA* (pp. 62-68). IEEE. doi: 10.1109/AINA.2011.74
- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language*. Oxford University Press, New York.
- Alm, R. & Wißotzki, M. (2013). TOGAF Adaption for Small and Medium Enterprises. In W. Abramowicz (ed.), *BIS (Workshops)* (p./pp. 112-123): Springer. ISBN: 978-3-642-41686-6. doi:10.1007/978-3-642-41687-3_12
- Álvarez, J. A. T., Nicolás, J., Moros, B. & Garcia, F. (2002). Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach. *Requir. Eng.*, 6, 205-219.
- Allen, C. (2001). *Realizing e-business with components*. Addison-Wesley, Harlow, Boston
- Alnusair, A., & Zhao, T. (2010). Component Search and Reuse: an ontology-based approach. In proceedings of the IEEE International Conference on Information Reuse and Integration (IRI 2010), pp. 258-261
- Ambler, S. (2002). *Agile modeling*. John Wiley and Sons
- Ampatzoglou, A. & Chatzigeorgiou, A. (2007). Evaluation of object-oriented design patterns in game development. *Information and Software Technology*, 49 (May (5)), (pp. 445–454), Elsevier. doi:10.1016/j.infsof.2006.07.003.
- Aranda, J., Easterbrook, S. M. & Wilson, G. (2007). Requirements in the wild: How small companies do it. *Requirements Engineering RE* (pp. 39-48), IEEE. ISBN: 0-7695-2935-6. doi: 10.1109/RE.2007.54
- Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., & Zettel, J. (2002). *Component based product line engineering with UML*. Addison-Wesley.
- Atkinson, C., Bayer, J., Laitenberger, O., Zettel, J. (2000). *Component-based Software Engineering: The Kobra Approach*. In 22nd International Conference on Software

Engineering (ICSE 2000), 3rd International Workshop on Component-based Software Engineering, Limerick, Ireland

Ballurio, K., Scalzo, B. & Rose, L. (2002). Risk Reduction in COTS Software Selection with BASIS. In J. C. Dean & A. Gravel (eds.), ICCBSS (p./pp. 31-43), Springer. ISBN: 3-540-43100-4

Barros, O. (2007) Business process patterns and frameworks: Reusing knowledge in process innovation. *Business Process Management Journal*, 13 (1), (pp. 47-69). doi: 10.1108/14637150710721122

Berczuk, S.P., (2003). Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Addison-Wesley Professional, 2003

Bilandzic, M. & Venable, J. (2011). Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics. *J. Community Informatics*, 7. Last accessed on October 9, 2013 at <http://ci-journal.net/index.php/ciej/article/view/786/804>

Bizagi. (2013). Bizagi Business Process Management. Last accessed on October 6, 2013 on <https://www.bizagi.com/>

Bhimani, J., Nakakura, T., Almahr, A., Sato, M., Sugiura, K. & Ohta, N. (2013). Vox populi: enabling community-based narratives through collaboration and content creation. In P. Paolini, P. Cremonesi & G. Lekakos (eds.), EuroITV (p./pp. 31-40), ACM. ISBN: 978-1-4503-1951-5. doi: 10.1145/2465958.2465976

Bhuta, J., Mattmann, C., Medvidovic, N. & Boehm, B. W. (2007). A Framework for the Assessment and Selection of Software Components and Connectors in COTS-Based Architectures. Working IEEE/IFIP Conference on Software Architecture WICSA. (p. 6). IEEE Computer Society. ISBN: 978-0-7695-2744-4. doi: 10.1109/WICSA.2007.2

Bishop, M. (2003). Computer Security: Art and Science. Addison Wesley

Boehm, B. W. (2000). Requirements that Handle IKIWISI, COTS, and Rapid Change. *IEEE Computer*, 33, 99-102.

Boehm, B. W., Port, D., Yang, Y., Bhuta, J. & Abts, C. (2003). Composable Process Elements for Developing COTS-Based Applications. ISESE (p./pp. 8-17), IEEE Computer Society. ISBN: 0-7695-2002-2

Bon, A., de Boer, V., Gyan, N.B., Aart, C., De Leenheer, P., Tuyp, W., Boyera, S., Froumentin, M., Grewal, A., Allen, M., Tangara, A., & Akkermans, H. (2013). Use Case and Requirements Analysis in a Remote Rural Context in Mali. In 19th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ, 2013, Essen, Germany, April 8-11, 2013, pp 331-346. doi: 10.1007/978-3-642-37422-7_24

Bosch, J. (2000). Design and use of software architectures. Addison-Wesley, England

Boukheduoma, S., Oussalah, M., Alimazighi, Z., & Tamzalit, D. (2013). Adaptation Patterns for Service-Based Inter-Organizational Workflows. In *Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS*, (pp. 1-10). IEEE. doi: 10.1109/RCIS.2013.6577722

- Brennan, R., Canning, L., & McDowell, R. (2008). *Business-to-Business-Marketing*, Sage Publications Limited, London
- Brito, F. & Abreu, E. (1995). The MOOD metric set. In proceedings of ECOOP '95 Workshop on Metrics
- Budgen, D., Turner, M., Brereton, P., & Kitchenham, B. (2008). Using mapping studies in software engineering. in 20th Annual Psychology of Programming Interest Group Conference, PPIG. Lancaster University, United Kingdom
- Buonanno, G., Faverio, P., Pigni, F., Ravarini, A., Sciuto, D. & Tagliavini, M. (2005). Factors affecting ERP system adoption: A comparative analysis between SMEs and large companies. *J. Enterprise Inf. Management*, 18, 384-426. doi: 10.1108/17410390510609572
- Bürsner, S., & Merten, T. (2010). RESC 2010: 1st Workshop on Requirements Engineering in Small Companies. In workshop proceedings of Requirements Engineering for Software Quality REFSQ 2010, ICB-Research Report no. 40, October 2010, p. 128-130. url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Buschmann, F., & Meunier, R. (1995). A System of Patterns. *Pattern Languages of Program Design*, 1, May, 1995, pp. 325-343
- Cheesman, J., Daniels, J. & Szyperski, C. (ed.) (2001). *UML Components - A Simple Process for Specifying Component-Based Software*. Addison-Wesley.
- Chung, L., Hill, T., Legunsen, O., Sun, Z., Dsouza, A. & Supakkul, S. (2013). A goal-oriented simulation approach for obtaining good private cloud-based system architectures. *Journal of Systems and Software*, 86, (pp. 2242-2262). doi: 10.1016/j.jss.2012.10.028
- Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering*. Boston, Dordrecht, London. Kluwer Academic Publishers.
- Chung, L., Supakkul, S., Subramanian, N., Garrido, J. L., Noguera, M., Hurtado, M. V., Rodríguez, M. L. & Akhlaki, K. B. (2011). Goal-Oriented Software Architecting. In P. Avgeriou, J. Grundy, J. G. Hall, P. Lago & I. Mistrik (ed.), *Relating Software Requirements and Architectures* (pp. 91-109). Springer. ISBN: 978-3-642-21000-6. doi 10.1007/978-3-642-21001-3_7
- Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating software architectures: methods and case studies*. Addison-Wesley, Boston
- Comella-Dorda, S., Dean, J. C., Morris, E. & Oberndorf, P. (2002). A Process for COTS Software Product Evaluation. In J. C. Dean & A. Gravel (eds.), *COTS-Based Software Systems, First International Conference, ICCBSS <p> 2002*, Orlando, FL, USA, February 4-6, 2002, Proceedings (p./pp. 86--96), Berlin, u.a.: Springer Verlag.
- Coplien, J.O. (1995a). A development process generative pattern language, AT&T Bell Laboratories. url: <http://www.bell-labs.com/people/cope/Patterns/Process/index.html> last accessed on October 6, 2013
- Coplien, J.O. (1995b). A generative development-process pattern language. In: Coplien, J.O., Schmidt, D.O. (Eds.), *Pattern Languages of Program Design*. Addison Wesley, Reading, MA, pp 183-237

- Crnkovic, I., Hnich, B., Johnson, T., Kiziltan, Z., (2002). Specification, implementation, and deployment of components. *Communications, Association of Computing Machinery* 45 (October (10)), (pp. 35–40). doi: 10.1145/570907.570928
- Crnkovic, I. & Larsson, M. (2000). A Case Study: Demands on Component-based Development. ICSE'2000 -- *International Conference on Software Engineering* (pp. 23--31), Limerick, Ireland. doi: 10.1109/ICSE.2000.870393
- Crnkovic, I. & Larsson, M. (2002). Challenges of component-based development. *Journal of Systems and Software*, 61, (pp. 201-212). doi: 10.1016/S0164-1212(01)00148-0
- de Boer, V., Leenheer, P. D., Bon, A., Gyan, N. B., van Aart, C., Guéret, C., Tuyp, W., Boyera, S., Allen, M. & Akkermans, H. (2012). RadioMarché: Distributed Voice- and Web-Interfaced Market Information Systems under Rural Conditions. In J. Ralyté, X. Franch, S. Brinkkemper & S. Wrycza (eds.), CAiSE (p./pp. 518-532), Springer. ISBN: 978-3-642-31094-2. doi: 10.1007/978-3-642-31095-9_34
- Dietz, J. L. G., Hoogervorst, J. A. P. (2011). A critical investigation of TOGAF - based on the enterprise engineering theory and practice. In Albani, A., Dietz, J.L.G., Verelst, J. (eds.), EEWC 2011, pp. 76-90, Springer. ISBN: 978-3-642-21057-0. doi: 10.1007/978-3-642-21058-7_6
- DPDL. (2013). Design Pattern Definition Language DPDL. Last accessed on October 6, 2013 on <http://alshayeb.com/DPDL/>
- dos Santos, P. S. M. & Travassos, G. H. (2009). Action Research use in software engineering: An initial survey. ESEM (pp. 414-417). ISBN: 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5316013
- dos Santos, P. S. M. & Travassos, G. H. (2011). Action Research Can Swing the Balance in Experimental Software Engineering. *Advances in Computers*, 83, (pp. 205-276). doi: 10.1016/B978-0-12-385510-7.00005-9
- Dwyer, M. B., Avrunin, G. S. & Corbett, J. C. (1999). Patterns in Property Specifications for Finite-State Verification. In B. W. Boehm, D. Garlan & J. Kramer (eds.), ICSE (pp. 411-420). ACM. ISBN: 1-58113-074-0. DOI: 10.1145/302405.302672.
- Eden, A. (1999). Precise Specification of Design Patterns and Tool Support in Their Application. PhD Thesis, University of Tel Aviv, Israel, 1999.
- El-Boussaidi, G. & Milli, H. (2012). Understanding design patterns - what is the problem? *Software: Practice and Experience*, 42, (pp. 1495-1529). doi: 10.1002/spe.1145
- Elizondo, P. V. & Lau, K.-K. (2010). A catalogue of component connectors to support development with reuse. *Journal of Systems and Software*, 83, (pp. 1165-1178). doi: 10.1016/j.jss.2010.01.008
- Firesmith, D. (2004). Specifying Reusable Security Requirements. *Journal of Object Technology*, 3, 61-75.
- Fowler, M. (1997). Analysis patterns: Reusable Object Models. Addison Wesley Longman, Inc.

- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional
- France, R., Kim, D.-K., Ghosh, S. & Song, E. (2004). A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering*, 30, (pp. 193-206). doi: 10.1109/TSE.2004.1271174
- France, R., Kim, D., Song, E., Ghosh, S. (2002). Role-Based Modeling Language (RBML) Specification v1.0. Technical Report 02-106, Computer Science Department, Colorado State University, Fort Collins, Colorado, June, 2002
- Franch, X., Palomares, C., Quer, C., Renault, S. & Lazzer, F. D. (2010). A Metamodel for Software Requirement Patterns. In R. Wieringa & A. Persson (eds.), *Requirements Engineering for Software Quality REFSQ*, pp. 85-90, Springer. ISBN: 978-3-642-14191-1. doi: 10.1007/978-3-642-14192-8_10.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley.
- Garlan, D., Allen, R. & Ockerbloom, J. (1995). Architectural Mismatch or Why It's Hard to Build Systems Out of Existing Parts. In D. E. Perry, R. Jeffrey & D. Notkin (eds.), *ICSE* (p./pp. 179-185): ACM. ISBN: 0-89791-708-1
- Glushko, R.J. & McGrath, T. (2002). Document engineering for e-business. *ACM Symposium on Document Engineering*, pp. 42-48, ACM
- Glushko, R.J. & McGrath, T. (2008). *Document Engineering – Analyzing and Designing Documents for Business Informatics and Web Services*. Cambridge, MA, USA. MIT Press.
- Goldkuhl, G. (2008). Practical Inquiry as Action Research and Beyond. In W. Golden, T. Acton, K. Conboy, H. van der Heijden & V. K. Tuunainen (eds.), *ECIS* (pp. 267-278). Last accessed on October 9, 2013 at <http://aisel.aisnet.org/ecis2008/118>
- Goldkuhl G. (2012). From Action Research to practice research. *Australasian Journal of Information Systems*, 17, 2, (pp. 57-78). url: <http://dl.acs.org.au/index.php/ajis/article/view/688>.
- Grant, D. & Ngwenyama, O. K. (2003). A report on the use of Action Research to evaluate a manufacturing information systems development methodology in a company. *Inf. Syst. J.*, 13, (pp. 21-36). doi: 10.1046/j.1365-2575.2003.00137.x
- Gyan, N. B., de Boer, V., Bon, A., van Aart, C., Akkermans, H., Boyera, S., Froumentin, M., Grewal, A. & Allen, M. (2013). Voice-based web access in rural Africa. In H. C. Davis, H. Halpin, A. Pentland, M. Bernstein & L. A. Adamic (eds.), *WebSci* (p./pp. 122-131), ACM. ISBN: 978-1-4503-1889-1. doi: 10.1145/2464464.2464496
- Hadar, I., Reinhartz-Berger, I., Kuflik, T., Perini, A., Ricca, F. & Susi, A. (2013). Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments. *Information & Software Technology*, 55, (pp. 1823-1843). doi: 10.1016/j.infsof.2013.05.003.
- Harel, D. & Pnueli, A. (1985). On the Development of Reactive Systems. *Logics and models of concurrent systems*, pp. 477-498

- Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G.K., & Svendsen, A. (2008). Adding Standardized Variability to Domain Specific Languages. In Proceedings of the 12th International Software Product Line Conference SPLC, pp. 139–148. doi: 10.1109/SPLC.2008.25v
- Hoffmann, A., Söllner, M. & Hoffmann, H. (2012a). Twenty Software Requirement Patterns to Specify Recommender Systems that Users will Trust. ECIS. (p. 185) Last accessed on October 9, 2013 at <http://aisel.aisnet.org/ecis2012/185>
- Hoffmann, A., Söllner, M., Hoffmann, H. & Leimeister, J. M. (2012b). Towards trust-based software requirement patterns. RePa, pp. 7-11, IEEE. ISBN: 978-1-4673-4374-9
- Hruby, P. (2006). *Model-Driven Design Using Business Patterns*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.
- Hsueh, N.-L., Chu, P.-H. & Chu, W. C. (2008). A quantitative approach for evaluating the quality of design patterns. *Journal of Systems and Software*, 81, (pp. 1430-1439). doi: 10.1016/j.jss.2007.11.724
- IBM WSC. (2012). IBM WebSphere Commerce. Last accessed on October 6, 2013 at <http://pic.dhe.ibm.com/infocenter/wchelp/v6r0m0/index.jsp>
- Izumi, M., Sato, M. & Sunahara, H. (2007). Requirements for Protection Methods of Personal Information in Vehicle Probing System. SAINT Workshops (p./pp. 70): IEEE Computer Society. ISBN: 0-7695-2757-4. doi: 10.1109/SAINT-W.2007.92
- Kalenborn, A. (2010). Modelling by Example: Requirements engineering during the bidding stage of dialog-oriented software projects. In *Proceedings of the 1st Workshop on RE in Small Companies RESC*, (pp. 158-166). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Kang, K., Cohen, S., Hess, J., Novak, W. & Peterson, S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-21). Software Engineering Institute, Carnegie Mellon University
- Kang, K. C., Lee, J. & Donohoe, P. (2002). Feature-Oriented Project Line Engineering. *IEEE Software*, 19 (4), (pp. 58-65), doi: 10.1109/MS.2002.1020288
- Khwaja S. & Alshayeb M. (2013). Towards design pattern definition language. *Software: Practice and Experience*, 43, (pp. 747-757). doi: 10.1002/spe.1122.
- Kilov, H. & Sack, I. (2009). Mechanisms for communication between business and IT experts. *Computer Standards & Interfaces*, 31(1), (pp. 98-109). doi: 10.1016/j.csi.2007.11.001
- Kim, D., France, R. & Ghosh, S. (2004). A UML based language for specifying domain-specific patterns. *Journal of Visual Languages and Computing*, 15(3-4): (pp.265-289) doi: 10.1016/j.jvlc.2004.01.004
- Kouroshfar, E., Shahir, H. Y. & Ramsin, R. (2009). Process Patterns for Component-Based Software Development. In G. A. Lewis, I. Poernomo & C. Hofmeister (eds.), CBSE (pp. 54-68). Springer. ISBN: 978-3-642-02413-9. doi: 10.1007/978-3-642-02414-6_4

- Kouskouras, K., Chatzigeorgiou, A., & Stephanides, G. (2008). Facilitating software extension with design patterns and Aspect-Oriented Programming. *Journal of Systems and Software* 81 (October (10)), (pp. 1725–1737), Elsevier. doi: 10.1016/j.jss.2007.12.807
- Lakhal, F., Dubois, H., & Rieu, D. (2013). Pattern-based Methodology for UML profiles evolution management. In *Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS*. IEEE. (pp. 1-12) doi: 10.1109/RCIS.2013.6577681
- Lano., K., Bicarregui, J., & Goldsack, S. (1996). Formalising Design Patterns. In *Processing of the 1st BCS-FACS Northern Formal Methods Workshop, Electronic Workshops in Computer Science*, 1996.
- Laukkanen, S., Sarpola, S., & Hallikainen, P. (2007). Enterprise size matters: objectives and constraints of ERP adoption. *J. Enterprise Inf. Management*, 20, 319-334. doi: 10.1108/17410390710740763
- Lee, O. (2002). An Action Research report on the Korean national digital library. *Information & Management*, 39, (pp. 255-260). doi: 10.1016/S0378-7206(01)00094-5
- Lyytinen, K. & Robey, D. (1999). Learning failure in information system development. *Information Systems Journal*, 9, 85–101.
- Mendez-Bonilla, O., Franch, X., & Quer, C. (2008) Requirements Patterns for COTS Systems. In *Proceedings of the 7th International Conference on Composition-Based Software Systems ICCBSS* (pp. 232-234). IEEE. doi: 10.1109/ICCBSS.2008.34
- Medvidovic, N. & Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26, 70–93.
- Millman, C. & El-Gohary, H. (2011). New Digital Media Marketing and Micro Business: A UK Perspective. *IJOM*, 1, (pp. 41-62). doi: 10.4018/978-1-4666-1598-4.ch076
- Oliveira, B. & Belo, O. (2012). BPMN Patterns for ETL Conceptual Modelling and Validation. In L. Chen, A. Felfernig, J. Liu & Z. W. Ras (eds.), *ISMIS* (pp. 445-454), : Springer. ISBN: 978-3-642-34623-1. doi: 10.1007/978-3-642-34624-8_50
- Opentaps. (2013). Opentaps. Last accessed on October 6, 2013 at <http://www.opentaps.org/>
- Paludo, M., Reinehr, S. S., Malucelli, A., Bruzon, L. & Pinho, P. (2011). Applying pattern structures to document and reuse components in component-based software engineering environments. *IRI* (pp. 378-383), *IEEE Systems, Man, and Cybernetics Society*. ISBN: 978-1-4577-0964-7. doi: 10.1109/IRI.2011.6009577
- Peixoto, M. & Silva, C. (2018). Specifying privacy requirements with goal-oriented modeling languages. In *Proceedings of the 32nd Brazilian Symposium on Software Engineering (SBES '18)*. ACM, New York, NY, USA, 112-121. doi: 10.1145/3266237.3266270
- Pokozy-Korenblat, K., Priami, C. & Quaglia, P. (2004). Performance Analysis of a UML Micro-business Case Study. In C. Priami & P. Quaglia (eds.), *Global Computing* (pp. 107-126), Springer. ISBN: 3-540-24101-9. doi: 10.1007/978-3-540-31794-4_7

Quispe, A., Marques, M., Silvestre, L., Ochoa, S. F. & Robbes, R. (2010). Requirements Engineering Practices in Very Small Software Enterprises: A Diagnostic Study. In S. F. Ochoa, F. Meza, D. Mery & C. Cubillos (eds.), SCCC, pp. 81-87, IEEE Computer Society.

Ramsin, R. & Paige, R. F. (2008). Process-centered review of object-oriented software development methodologies. *ACM Comput. Surv.*, 40 (1), Article 3, (pp. 1- 89)

Reinhartz-Berger, I., Sturm, A. & Tsoury, A. (2011). Evaluating Comprehension and Utilization of Variability Aspects in UML-Based Models. In S. Nurcan (ed.), *CAiSE Forum (Selected Papers)* (pp. 156-171), Springer. ISBN: 978-3-642-29748-9. doi: 10.1007/978-3-642-29749-6_11

Riaz, M. & Williams, L. (2012). Security requirements patterns: understanding the science behind the art of pattern writing. *RePa* (pp. 29-34): IEEE. ISBN: 978-1-4673-4374-9. doi: 10.1109/RePa.2012.6359977

Roost, M., Taveter, K., Rava, K., Tepandi, J., Pihö, G., Kuusik, R., & Öunapuu, E. (2013). Towards Self-development of Evolutionary Information Systems: An Action Research of Business Architecture Development by Students in Socially Networked Groups. *Proceedings of the International Workshop on Approaches for Enterprise Engineering Research AppEER 2013 at the 25th International Conference on Advanced Information Systems Engineering (CAiSE)*. Vol 148. doi: 10.1007/978-3-642-38490-5_1

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. (1991). *Object-Oriented Modeling and Design*.

Schmidt, D.C., Fayad, M., & Johnson, R.E. (1996). *Software Patterns*. Communications of the ACM, October, 1996

Schumacher, M., Fernandez-Buglioni, E., Hyberston, D., Buschmann, F., & Sommerlad, P. (2006). *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Limited

Seruca, I. & Loucopoulos, P. (2003). Towards a systematic approach to the capture of patterns within a business domain. *Journal of Systems and Software*, 67 (1). (pp. 1-18) doi: 10.1016/S0164-1212(02)00083-3

Shaw, M. & Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall

Siau, K. & Cao, Q. (2002). How Complex Is the Unified Modeling Language? In *Advanced Topics in Database Research*, Vol. 1 (pp. 294-306)

Sochos, P., Philippow, I. & Riebisch, M. (2004). Feature-Oriented Development of Software Product Lines: Mapping Feature Models to the Architecture. In M. Weske & P. Liggesmeyer (eds.), *Net.ObjectDays* (p./pp. 138-152), Springer. ISBN: 3-540-23201-X

Smith, B. H. & Williams, L. (2012). On the Effective Use of Security Test Patterns. *SERE* (p./pp. 108-117), IEEE. ISBN: 978-0-7695-4742-8

Stepan, P. & Lau, K.-K. (2012). Controller patterns for component-based reactive control software systems. In V. Grassi, R. Mirandola, N. Medvidovic & M. Larsson (eds.), *CBSE* (pp. 71-76). ACM. ISBN: 978-1-4503-1345-2. doi: 10.1145/2304736.2304749

- Supakkul, S. & Chung, L. (2009). Extending Problem Frames to deal with stakeholder problems: An Agent- and Goal-Oriented Approach. In S. Y. Shin & S. Ossowski (eds.), SAC pp. 389-394, ACM. ISBN: 978-1-60558-166-8
- Supakkul, S., Hill, T., Chung, L., Tun, T., & Sampaio do Prado Leite, J.C. (2010). An NFR Pattern Approach to Dealing with NFRs. In *Proceedings of the 18th IEEE International Requirements Engineering Conference RE* (pp. 179-188). IEEE. doi: 10.1109/RE.2010.31
- The Open Group. (2009). TOGAF Version 9. Van Haren Publishing, Zaltbommel
- Thom, L. H., Lazarte, I. M., & lochpe, C. (2009a). Activity patterns in process-aware information systems: basic concepts and empirical evidence. *IJBPM* 4 (2), 93-110, 2009
- Thom, L. H., Lazarte, I. M., & lochpe, C. (2009b). On the Support of Workflow Activity Patterns in Process Modeling Tools: Purpose and Requirements. In 3rd WBPM, 2009, Brazil
- Thom, L. H., Lazarte, I. M., lochpe, C., Priego-Roche, L.-M., Verdier, C., Chiotti, O. & Villarreal, P. D. (2011). On the Capabilities of BPMN for Workflow Activity Patterns Representation. In R. M. Dijkman, J. Hofstetter & J. Koehler (eds.), *BPMN* (pp. 172-177). Springer. ISBN: 978-3-642-25159-7. doi: 10.1007/978-3-642-25160-3_18
- van Gorp, J., Prehofer, C. & Bosch, J. (2010). Comparing practices for reuse in integration-oriented software product lines and large open-source software projects. *Software: Practice and Experience*, 40, (pp. 285-312). doi: 10.1002/spe.955
- Warmer, J. & Kleppe., A. (1999). *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley
- Weitlaner, D., Guettiner, A., & Kohlbacher, M. (2013). Intuitive Comprehensibility of Process Models. *S-BPM ONE 2013*, pp. 52-71. doi: 10.1007/978-3-642-36754-0_4
- Wen, Y., Zhao, H. & 0001, L. L. (2011). Analysing security requirements patterns based on problems decomposition and composition. *RePa* (p./pp. 11-20), IEEE. ISBN: 978-1-4577-1020-9
- Withall, S. (2007a). *Software Requirement Patterns*. O'Reilly
- Withall, S. (2007b). *Software Requirement Patterns*. Microsoft Press
- Wouters, B., Barjis, J., Maponya, G., Maritz, J., & Mashiri, M. (2009). Supporting Home Based Health Care in South African Rural Communities Using USSD Technology. In R. C. Nickerson & R. Sharda (eds.), *AMCIS* (p./pp. 410), Association for Information Systems. url: <http://aisel.aisnet.org/amcis2009/410/>
- Yang, Y., Bhuta, J., Boehm, B. & Port, D. N. (2005). Value-Based Processes for COTS-Based Applications. *IEEE Software*, 22, 54--62.
- Yoshioka, N., Washizaki, H., & Maruyama, K. (2008). A survey on security patterns. *Progress in Informatics, Special Issue: The future of software engineering for security and privacy*, pp. 13
- Zhao, X., & Zou, Y. (2011). A business process-driven approach for generating software modules. *Software: Practice and Experience*, 41, (pp. 1049–1071). doi: 10.1002/spe.1068

Chapter III

Theory and Practicality of Modeling in the Domain of Micro-businesses

Having reviewed the literature and identified the main contributions both in theory and in practice in the previous chapter, this chapter analyzes and discusses work related to the practical use of the fundamental symbols, languages, concepts, and elements of the models that can be useful candidates to be used to strengthen our proposal in the next chapter. In particular, the focus is on the modeling of the business processes and their associated software components in the domain of micro-businesses.

Since the objectives of this thesis are to come up with a proposal which is comprehensible and technically relevant both for micro-business owners and software developers, we try to avoid the presence of too many concepts and elements in the models that would unnecessarily increase the cognitive load both for the modelers and those who would use the models (Chandler & Sweller, 1991). Our rationale is to propose candidate concepts and elements which are *easy-to-use*, *relevant for the practitioners* in the micro-business domain, and *comprehensible* for the micro-business owners with limited technical backgrounds.

On the same note of practicality, prior research on the trade-off between complexity and communication clarity (Gemino & Wand, 2005) further suggests that in practice, modeling method users would *prefer* efficient (i.e., non-complex) modeling methods over effective (i.e., highly expressive but complex) ones. This means that in industry, a user would choose a simpler and more efficient modeling method such as a simplified library of the Business Process Modeling Notation “BPMN” over using an effective and expressive modeling method such as the complete BPMN specification as originally defined by (Object Management Group, 2008). In addition, a recent survey also showed that when requirements engineering practices are employed, smaller businesses tend to adopt the easier techniques (Kassab, 2021).

1. THE PRACTICAL USE OF THE BUSINESS PROCESS MODELING NOTATION

The Business Process Modeling Notation “BPMN” aims to be a notation that is readily understandable by all business users, from the business analysts that create the initial drafts

of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes (Object Management Group, 2008).

In a study made by (Recker, 2008) about the factors explaining and predicting user acceptance of BPMN, he found that user acceptance of BPMN is primarily dependent on two factors: instrumentality (usefulness and performance of BPMN for process modeling) and easiness (complexity of creating BPMN models). Both instrumentality and easiness, in turn, relate to two main characteristics of any modeling language:

- (1) Expressiveness: the possibility to model everything that has to be shown in a model
- (2) Complexity: how hard it is to choose and specify the correct representations

These findings are a clear call for standardization bodies to produce standards that are not only technically sound but also likable and manageable by the people who are meant to use them in practice.

To assess the quality of the symbols and elements of BPMN, (Wahl & Sindre, 2005) evaluated BPMN using the Semiotic Quality Framework by (Krogstie & Sølvsberg, 2003). They concluded that BPMN excels in its comprehensibility because of its construct specializations and type aggregations which make it suitable for use for business process modeling in general. We evaluate the suitability of using BPMN in the domain of micro-businesses in the subsequent chapters.

BPMN talks about 38 different language constructs and attributes which are grouped into four basic categories of elements. They are:

- (1) Flow Objects: include events, activities and gateways and are the most basic elements for creating Business Process Diagrams.
- (2) Connecting Objects: used for connecting the Flow Objects through different types of arrows.
- (3) Swim lanes: group activities into separate categories for different functional capabilities or responsibilities such as various roles or departments.

(4) Artefacts: added to a diagram for additional information, e.g., relevant data, complementary notes.

Based on the Workflow Patterns Framework by (van der Aalst et al., 2003), (Wohed et al., 2005) evaluated BPMN based on its capability to express a series of control flow, data and resource patterns. They found that BPMN supports the majority of the control flow patterns, nearly half of the data patterns, and a few resource patterns. This means that BPMN is meant more for modeling flows than showing data and resource structures.

In the Appendix B.1 of this thesis, some of the most basic BPMN elements from the complete BPMN specification are explained in detail. We only include these basic BPMN elements in our proposal to minimize the cognitive load required by the users because too much theoretical information curtails usefulness. Our selection of the most basic BPMN elements make our proposal more practical in the domain of micro-businesses.

Understanding the trade-offs between theoretical limitations and practical advantages of BPMN help us in the reasoning for using BPMN in the micro-business domain. A theoretical model by (Recker et al., 2006) predicted nine different propositions regarding the limits and shortcomings of BPMN. The empirical investigation, however, revealed that not all the theoretical predictions constitute critical problems in process modeling practice. Despite these theoretical limitations of BPMN, there have been studies of BPMN use in practice which show its advantages. For instance, BPMN is currently being supported by more than 60 commercial and academic process modeling products and is finding rapid adoption in industry (Recker, 2010).

According to a study made by (Recker, 2010), BPMN has quickly become a de facto standard for graphical process modeling. No other notation has seen such an uptake in such a short time as BPMN has. It is widely supported by both free and commercial process modeling tools such as SparxSystems, itp-commerce, Websphere, Sungard, Intalio, Tibco, IBM, Pega, and Telelogic. BPMN has integrated into the curriculum of education providers like Queensland University of Technology, Widener University, Howe School of Technology Management, and part of the offerings of modeling coaches and consultants such as BPM-Training.com, BPMInstitute.org, and Object Training. Other standardization bodies such as the Workflow Management Coalition have also revised their standard development efforts to include BPMN in 2008.




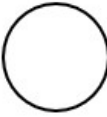


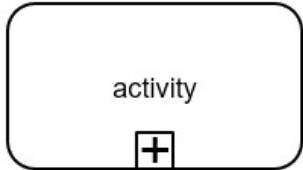


In a study of BPMN modeling in practice, a typical BPMN diagram contains less than 10 symbols and that the frequency of symbol use follows a long-tail distribution, similar to the use of words in natural language (zur Muehlen & Recker, 2008). A possible conjecture that follows from this observation is that users deliberately reduce the complexity of the language by restricting its vocabulary and consequently the rules governing the unused parts of the vocabulary. Following such conjecture in the micro-business domain, limiting the use of symbols and elements in BPMN to just the most basic and practical ones would allow the notation to be even more readily understandable to micro-business owners and the software developers they work with, just like restricting language in its most basic and practical form.

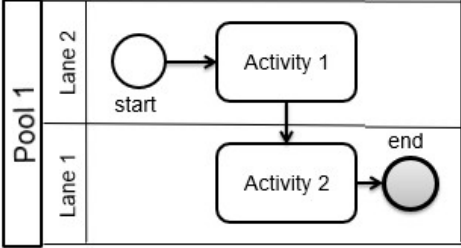





BPMN uses intuitive shapes and icons for its graphical elements so that they can be readily understandable for everyone. In our case, *everyone* refers to every user in the micro-business domain. There is a trend of companies using mostly BPMN core elements (Recker, 2010). These core elements are Normal Flow, Task, Start/End Event, Pools, and Data-Based Decisions. For use in our proposal, we explain these core shapes and icons of BPMN and some other basic elements as specified in the original BPMN (Object Management Group, 2008) in Appendix B.1 of our thesis.

In addition, as a response to the shortage of BPMN training and skilled BPMN users in the market nowadays as described by (Recker, 2010), a User Guide and tutorials for BPMN are found in the Appendices C.2 and C.3 of this thesis and are also made available to the public for those who intend to use our proposal in practice. The User Guide and tutorials are also a concrete response to the research question posed by (Recker, 2010): "What is the beginner BPMN course supposed to look like?".

The purpose of the User Guide and tutorials are not to make BPMN experts out of the users but to get them started in the use of BPMN. The key is to make the first-time users understand basic BPMN elements without overcomplicating it. On the same note, understanding the BPMN elements in the User Guide and tutorials would be key to understanding our proposal in the next chapter. A table of the BPMN elements found in our User Guide and tutorials are listed below:

Table III.1 Candidate BPMN Elements for this Thesis

Symbol	Name
	Sequence Flow
	Default Flow
	Conditional Flow
	Start Event
	End Event
	Activity
	Activity with Sub-Process
	Data Object
	Data Store

	Pools and Lanes
	Exclusive Gateway
	Inclusive Gateway
	Parallel Gateway
	Event-Based Gateway
	Complex Gateway

2. PRACTICALITY WITH THE UNIFIED MODELING LANGUAGE ON THE SAME NOTE

Like BPMN, the Unified Modeling Language “UML” has achieved widespread adoption in industry practice (Dobing & Parsons, 2006) (Recker, 2010). UML is complex and some researchers even claim that UML is 2-11 times more complex than other modeling methods (Siau & Cao, 2002). However, there are also studies that acknowledge the complexity of UML but also note that UML is more often used informally and not as intended by its developers (Siau & Cao, 2001) (Siau et al., 2005) (Dobing & Parsons, 2006) (Siau & Tian, 2009).

(Siau et al., 2005) introduced and coined the terms “theoretical complexity” and “practical complexity” of modeling methods and argued that the use of modeling methods in the field might be very different from the use as advocated by the developers of such methods. A possible conjecture is that the practical complexity of BPMN and UML is expected to be lower than the theoretical complexity (Recker et al., 2009). Based on this conjecture, we expect that our practical use of UML would also be lower than its theoretical complexity and that UML would also be suitable for use in the domain of micro-businesses.

We could use this analogy of theoretical complexity and practical complexity in operating a motorized vehicle. When applying for a license to operate a motorized vehicle, understanding the theoretical complexity on how the vehicle works, its maintenance, and also how it should be used on public roads could go on and on, for several pages. A theoretical driving manual can go into so much detail, even up to the explanation of how to convert psi to bar for measuring tire pressure or how braking distance does not increase linearly as the speed of your vehicle increases. It could become quite complex for someone who is new to motorized vehicles.

The practical complexity of operating a motorized vehicle does not necessarily involve strict mathematical formulas or sophisticated physics theory. A person can simply enter the vehicle, switch the vehicle on, select the right gear, drive from point A to point B, park the vehicle, and then leave the vehicle. You only need to understand some practical matters to use the vehicle for its purpose.

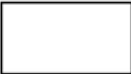
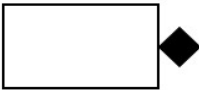
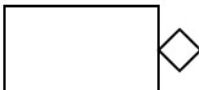
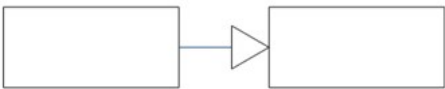
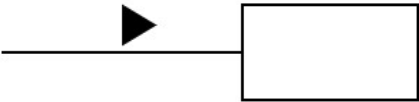
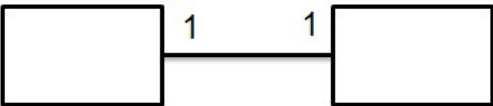
We use UML version 2.2 (Object Management Group, 2009) and select the elements that we use in our proposal as discussed in the next chapter. The UML specification models behavior or structure. For our use, among the structural models, we use the most basic elements of the class diagrams and the most basic elements of component diagrams in order to avoid increasing the cognitive load for the stakeholders involved in the use of our models. Understanding the basic elements of these two UML specification models are fundamental in comprehending our proposal:

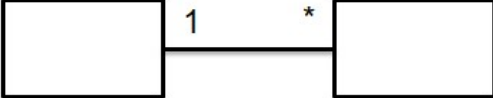
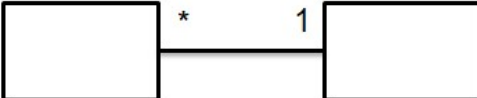
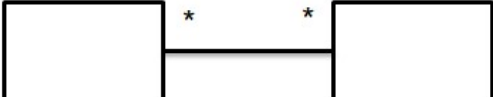

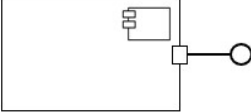
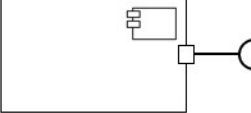
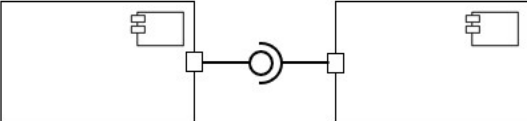
- (1) For Class Diagrams: a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods) (Gamma et al., 1995).

(2) For Component Diagrams: components are considered autonomous, encapsulated units within a system or subsystem that provide one or more interfaces (Object Management Group, 2009).

In a similar fashion, we explain the UML elements that we use in our proposal in the Appendix B.3 of this thesis. A User Guide and tutorial for the UML elements are provided for those who intend to apply our proposal in practice. Below is a table of the most basic UML elements that are discussed in the User Guide, tutorials, and used in our proposal in the next chapter.

Table III.2 Candidate UML Elements for this Thesis

Symbol	Name
	Class
	Container (Composition)
	Container (No Composition)
	Inheritance
	Action
	One-to-One Relationship

	One-to-Many Relationship
	Many-to-One Relationship
	Many-to-Many Relationship
	Component
	Provided Interface
	Required Interface
	Connecting Interfaces






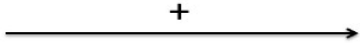
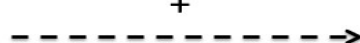
3. PRACTICAL DIAGRAMMING AND MODELS FOR NON-FUNCTIONAL REQUIREMENTS

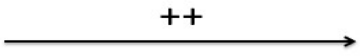
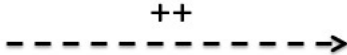

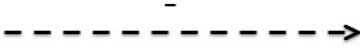
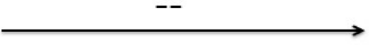
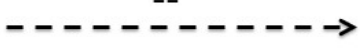



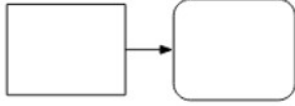
To understand how Non-Functional Requirements “NFRs” are used in practice, it is important to understand how to diagram and illustrate Softgoal Interdependency Graphs “SIGs.” SIGs were first proposed by Lawrence Chung, the head of our research team in the United States of America, in the NFR Framework (Chung et al., 2000). SIGs are used to diagram and illustrate NFRs.

While BPMN is used to model processes and UML is used to model components in practice, SIGs are used to model NFRs in real-world cases (Supakkul et al., 2010; Chung et al., 2011; Chung et al., 2013). The practical use of SIGs in the real world makes SIGs an ideal choice for use in the micro-business domain. In a similar fashion, SIGs per se may be complex with all of its theoretical elements but we use SIGs in a practical way to make it suitable for the micro-business domain. Hence, we choose a selection of the most basic elements in SIGs for use in our proposal in this thesis in order to minimize the cognitive load for the stakeholders involved in the use of our proposal. A more detailed discussion of SIGs can be found in the original manuscript of (Chung et al., 2000).

In Appendix B.2 of this thesis, we provide a detailed explanation of the SIG elements that we use in our proposal. There is also a detailed explanation of SIGs in the User Guide and tutorials for those who would be putting our proposal into practice. Below is a table of the most basic SIG elements we have selected for use in our proposal in the next chapter.

Table III.3 Candidate SIGs Elements for this Thesis

Symbol	Name
	Softgoal
	Refinement "AND"
	Refinement "OR"
	Direct / Explicit Relationship
	Indirect / Implicit Relationship
	Positive Direct Dependency
	Positive Indirect Dependency

	Strong Positive Direct Dependency
	Strong Positive Indirect Dependency
	Negative Direct Dependency
	Negative Indirect Dependency
	Strong Negative Direct Dependency
	Strong Negative Indirect Dependency
	Operationalizing Method
	Operationalization Target Link
	Target System
	Design Decision Link to Functional Requirement

4. A PRACTICAL CHARACTERIZATION AND DESCRIPTION OF PATTERNS IN SOFTWARE

Aside from the notations and languages that are theoretically complex at the onset, the idea of patterns has also become complex and when applied in practice, rely heavily on expert development skills (Serrato-Barrera et al., 2020). There are so many different characterizations and descriptions of patterns in software that the idea of using patterns could be more confusing than helpful for a micro-business owner and software developer who simply want to get things done without complicating matters.

4.1 Characterization of Patterns

For example, one of the first characterizations of patterns in software was made by (Alexander et al., 1977) and he characterizes a pattern as a description to a problem that occurs again and again in our surroundings, and also a solution for this problem, in a way that the solution could be used a million times without changing it even twice. This characterization is simple and straightforward and could be readily understood by the users in the micro-business domain. However, many researchers after him believed that this characterization was insufficient and needed to be refined even more.

One of the more popular characterizations of patterns was made by (Gamma et al., 1995) as mentioned in the related literature in the previous chapter. They say that patterns are a set of classes and objects which are ready to solve a design problem in a particular context. Such a definition has become very popular among software practitioners but may not be the most practical definition to use in the micro-business domain because of a lot of technical jargon, e.g., class, object.

There are also popular characterizations of patterns such as the one made by (Riehle and Züllighoven, 1996). They say that a pattern is an abstraction of a concrete form that repeats itself in various contexts. Although this definition may seem easier to understand for micro-business owners than the definition of (Gamma et al., 1995), the word “abstraction” could lead to confusion among micro-business owners.

Limiting the use of technical jargon is essential to arrive at a practical characterization of patterns in software for micro-businesses. Aside from the straightforward characterization of (Alexander et al. 1977), two more characterizations could be suitable for the micro-business domain. One of the characterizations is from (Fowler, 1997) and he says that a pattern is an idea that has been useful in a particular context and probably useful in others. Like the characterization of (Alexander et al., 1977), it is a simple, straightforward, and uncomplicated way of talking about patterns in the micro-business domain.

There is another characterization of patterns in software that would be apt for the micro-business domain and would even be simpler and more straightforward than those provided by (Alexander et al., 1977) and (Fowler, 1997). It is a characterization that even the least technically-inclined micro-business owner would understand. It is as simple and classic as a characterization that can be found in the official dictionary of your day-to-day language. The (Real Academia Española, 2003) defines a pattern as a model which serves as an example to make another same thing. The various definitions and characterizations of patterns in software made by different researchers are shown in Table III.4.

Table III.4 Characterizations of Patterns in Software

Author	Characterization
Alexander et al., 1977	Every pattern describes a problem that occurs again and again in our surroundings, and also a solution for this problem, in a way that the solution could be used a million times without changing it even twice
Gamma et al., 1995	Patterns are a set of classes and objects which are ready to solve a design problem in a particular context
Riehle and Züllighoven, 1996	A pattern is an abstraction of a concrete form that repeats itself in various contexts
Fowler, 1997	A pattern is an idea that has been useful in a particular context and probably useful in others

Real Academia Española, 2003	A pattern is a model which serves as an example to make another same thing
------------------------------	--

On top of characterizing a pattern for the micro-business domain, it would also be important to characterize what kind of pattern we would be proposing. There are some popular pattern classifications such as the one made by (Riehle and Züllighoven, 1996). They mainly classify patterns as: Conceptual Patterns, Coding Patterns, and Design Patterns.

Conceptual patterns help in constructing the conceptual model of the software system. Coding patterns help in writing the code for the software system. Design patterns help in designing the software system. There are also Analysis patterns as described by (Fowler, 1997) which help in analyzing the software system.

The technical characterizations and classifications of a pattern in software possibly add to the cognitive load to the users of our proposal. Hence, we try to be as practical as possible in the micro-business domain by referring to a pattern in software by using natural language: **reusable solutions to recurring requirements** in the micro-business domain. The patterns we propose would help in **analyzing** and **designing** micro-business software as will be explained in the next chapter.

4.2 Description of a Pattern

The description of a pattern in software in the micro-business domain must be comprehensible and structured for the practical use of micro-business owners and software developers. A pattern description made by (Meszaros and Doble, 1998) could be used as a starting point for describing patterns in software for the micro-business domain. They say that patterns in software are described with the following mandatory elements:

- (a) Name: the identifier which the pattern is referred to and fitting for what it models
- (b) Context: the state and circumstances surrounding the pattern
- (c) Problem: what the patterns plans to solve
- (d) Factors: constraints and considerations to bear in mind when using the pattern
- (e) Solution: how the problem is solved considering various factors

They also say that patterns in software are also described with optional elements such as:

- (a) Indicators: these are signs, symptoms, or hints which suggest applying the pattern
- (b) Resulting Side Effects: these are outcomes that could occur when the pattern is applied and could be resolved using other patterns
- (c) Related Patterns: other patterns that may solve the problem
- (d) Examples: a demo on the use of the pattern
- (e) Code samples: lines of code showing the application of the pattern
- (f) Rationale: a thorough description of the purpose of the pattern and why it works in a certain situation
- (g) Aliases: alternative names which the pattern is known
- (h) Acknowledgments: list of contributors to the pattern

From both these mandatory and optional elements, there is still a lot of information that unnecessarily increases the cognitive load for the users of the pattern. Hence, we simplify the description of patterns in software so that it would be more practical and suitable for the micro-business domain. Such users are more focused on getting things done with the least complexity possible. We present our pattern description in detail in the next chapter.

5. CONCLUSIONS

In this chapter, on the basis of a discussion and analysis of previous related work on the modelling of business processes and their associated software components in the domain of micro-businesses, we have presented the fundamental candidate elements, concepts, and characterizations in the models that we plan to use in our proposal in the next chapter. The main motivation for the selection of the chosen elements and characterizations is to **minimize the cognitive load for the stakeholders who are mainly involved in requirements elicitation and analysis**. Reducing unnecessary cognitive load allows our proposal to be practical and suitable for the micro-business domain where stakeholders are not normally exposed to technical jargon and would rather use their natural language to express and understand their software requirements. Stakeholders in the micro-business domain are also focused on getting things done with the least complexity as possible.

CHAPTER REFERENCES

- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language*. Oxford University Press, New York.
- Chandler, P., & Sweller, J. (1991). Cognitive Load Theory and the Format of Instruction. *Cognition and Instruction* (8:4), (pp. 293-332)
- Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering*. Boston, Dordrecht, London. Kluwer Academic Publishers.
- Chung, L., Supakkul, S., Subramanian, N., Garrido, J. L., Noguera, M., Hurtado, M. V., Rodríguez, M. L. & Akhlaki, K. B. (2011). Goal-Oriented Software Architecting. In P. Avgeriou, J. Grundy, J. G. Hall, P. Lago & I. Mistrík (ed.), *Relating Software Requirements and Architectures*, pp. 91-109. Springer. ISBN: 978-3-642-21000-6.
- Chung, L., Hill, T., Legunsen, O., Sun, Z., Dsouza, A. & Supakkul, S. (2013). A goal-oriented simulation approach for obtaining good private cloud-based system architectures. *Journal of Systems and Software*, 86, (pp. 2242-2262). doi: 10.1016/j.jss.2012.10.028
- Dobing, B., & Parsons, J. (2006). How UML is Used. *Communications of the ACM* 49:5, (pp. 109-113)
- Gemino, A., & Wand, Y. (2005). Complexity and Clarity in Conceptual Modeling: Comparison of Mandatory and Optional Properties. *Data & Knowledge Engineering* 55:3, (pp. 301-326)
- Fowler, M. (1997). *Analysis patterns: Reusable Object Models*. Addison Wesley Longman, Inc.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley.
- Kassab, M. (2021). How Requirements Engineering is Performed in Small Businesses? 29th International Requirements Engineering Conference Workshops (REW), 2021, IEEE, (pp. 220-223), doi: 10.1109/REW53955.2021.00041.
- Krogstie, J. & Sølvsberg, A. (2003). *Information Systems Engineering - Conceptual Modeling in a Quality Perspective*. Kompendiumforlaget, NTNU, Trondheim, Norway.
- Meszaros, G. & Doble, J. (1998). A pattern language for pattern writing. In Martin, Riehle and Buschmann (eds.), *Pattern Languages of Program Design 3*, pp. 529-574. Reading, MA, Addison-Wesley
- Object Management Group, Inc. (2008). *Business Process Modeling Notation Version 1.1*. Last accessed on March 10, 2011 at <http://www.omg.org/spec/BPMN/1.1/PDF>
- Object Management Group, Inc. (2009). *Unified Modeling Language Version 2.2*. Last accessed on March 10, 2011 at <http://www.omg.org/spec/UML/2.2/Superstructure/PDF/changebar>
- Real Academia Española. (2003). *Diccionario de la Lengua Española*. 22nd Edition. Espasa Calpe

- Recker, J. (2008). Understanding Process Modelling Grammar Continuance: A Study of the Consequences of Representational Capabilities. Faculty of Information Technology, Queensland University of Technology, Brisbane.
- Recker, J., zur Muehlen, M., Siau, K., Erickson, J., & Indulska, M. (2009). Measuring method complexity: UML versus BPMN. In: 15th Americas Conference on Information Systems, 6-9 August, 2009, San Francisco, California.
- Recker, J. (2010). Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, Vol. 16 No. 1, (pp. 181-201), <https://doi.org/10.1108/14637151011018001>
- Riehle, D. & Züllighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Software Systems*, 2 (1): 3-13
- Serrato-Barrera, R., Rodríguez-Gómez, G., Pérez-Sansalvador, J.C., Pomares-Hernández, S., Flores-Pulido, L., and Muñoz, A. (2020). Software system design based on patterns for Newton-type methods. *Computing* 102, pp. 1005–1030
- Siau, K., & Cao, Q. (2001). Unified Modeling Language: A Complexity Analysis. In *Journal of Database Management*, 12:1, (pp. 26-34)
- Siau, K. & Cao, Q. (2002). How Complex Is the Unified Modeling Language? In *Advanced Topics in Database Research*, Vol. 1 (pp. 294-306)
- Siau, K. & Tan, X. (2005). Improving the Quality of Conceptual Modeling Using Cognitive Mapping Techniques. In *Data & Knowledge Engineering* 55:3, (pp. 343-365)
- Siau, K. & Tian, Y. (2009). A Semiotics Analysis of UML Graphical Notations. *Requirements Engineering* 14:1, (pp. 15-26)
- Supakkul, S., Hill, T., Chung, L., Tun, T., & Sampaio do Prado Leite, J.C. (2010). An NFR Pattern Approach to Dealing with NFRs. In *Proceedings of the 18th IEEE International Requirements Engineering Conference RE* (pp. 179-188). IEEE. doi: 10.1109/RE.2010.31
- van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. & Barros, A.P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14 (1), 5-51
- Wahl, T. & Sindre, G. (2005). An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In *Proceedings of the CAiSE'05 Workshops*. Volume 1, Castro, J. and E. Teniente, Eds., (pp. 533-544), FEUP, Porto, Portugal.
- Wohed, P., van der Aalst, W.M.P., Dumas, M. & ter Hofstede, A.H.M. (2005). Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives. In *BPM Center Report No. BPM-05-26*. BPMcenter.org.
- zur Muehlen, M., & Recker, J. (2008). How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Léonard, M., and Bellahsene, Z. (Eds.) *Advanced Information Systems Engineering - CAiSE 2008*, Montpellier, France. (pp. 465-479). Springer

Chapter IV

Micro-business Requirements Patterns “ μ BRPs”

In this chapter, we propose the ***Micro-business Requirements Patterns (μ BRPs)*** which are *patterns of recurring requirements models in micro-business software projects*. This proposal is a response to the research questions of coming up with a comprehensible and technically relevant requirements proposal for micro-businesses. The μ BRPs aim to help micro-business owners express and comprehend their software requirements better and aim to assist software analysts/developers when aligning on requirements and technical matters such as software analysis, design, and component reuse.

Reusing recurring requirements models in real-world micro-businesses *grounds* our μ BRPs as they are *based on actual industry practice*. This is a strong argument which could be used by industry practitioners when applying μ BRPs. As will be shown in our μ BRPs, we focus on *facilitating the real-world communication* between the micro-business owners and the analysts/developers since it is not guaranteed that everyone in the micro-business domain would understand the technical jargon which engineers have been accustomed with throughout their years of education and work experience.

We discuss μ BRPs step-by-step in this chapter in the following subsections. First, we provide our characterization of a micro-business, describing it in relation to the other characterizations of micro-businesses mentioned in the first chapter. Then, we present the conceptual model which covers the main concepts and the relationships among them, the description of μ BRPs with its corresponding parts (models and notes), a demonstration of μ BRPs using a real-world example, and a supporting tool which is available for the users of μ BRPs. Finally, we discuss the activities for managing the μ BRPs, including its creation and day-to-day use.

1. CHARACTERIZATION OF A MICRO-BUSINESS

As elaborated in Chapter 1, Section 1, there are various ways of defining or characterizing a micro-business, e.g., from micro-businesses based on headcount to micro-businesses based on the amount of capitalization involved. The different ways to characterize a micro-business can be confusing if we do not come up with a working characterization of a micro-business. Hence, we find it necessary to arrive at a working characterization of micro-businesses for clarification in this thesis.

Based on our ongoing research and from our industry experience working with micro-business software systems, it could be clearer if we would refer to a micro-business as the smallest kind of business, **using arithmetic criteria**. This could **avoid confusion** from the point of view of researchers and from the point of view of the practitioners who would use our proposal in industry. If we refer to micro-businesses with arithmetic, we could avoid problems when identifying them and filtering what a micro-business *is* and what *it is not* from the crowd. Arithmetic can be used as a guide to identify micro-businesses. However, of course additional analysis may be needed in order to determine if the proposal could be applied to other specific cases.

Hence, we refer to a micro-business by using the **combined characterizations** made by the **European Commission** and **our work** in this thesis. The European Commission refers to a micro-business as a business which employs fewer than 10 people and whose annual turnover or annual balance sheet total does not exceed 2 million Euros (European Commission, 2013). In Table IV.1, the European Commission differentiates micro-businesses from other enterprises using arithmetic.

Table IV.1 Definition of Small to Medium Sized Enterprises by the European Commission

Enterprise Size	Headcount	Annual Revenue	Annual Balance Sheet
Medium	< 250 people	≤ 50 million Euros	≤ 43 million Euros
Small	< 50 people	≤ 10 million Euros	≤ 10 million Euros
Micro	< 10 people	≤ 2 million Euros	≤ 2 million Euros

It is important for the European Commission to provide *arithmetic criteria* for determining whether a business *is a micro-business or not* because of the kind of help and assistance provided by the European Commission to them. Micro-businesses will have different aids and different registration and application procedures as opposed to Medium-sized businesses.

*In addition, we also use arithmetic as a guide to determine whether a micro-business could be one or not. If a micro-business has a software implementation of **less than 10-man days** and a software system design that involves **less than 10 software components** then we could characterize this business as a micro-business. (Macasaet et al., 2014; Macasaet et al., 2019).*

The idea of having an upper bound of 10-man days of software implementation for micro-businesses and an upper bound of 10 software components for software system design is to *limit the complexity* of the implementation when applying our proposal in practice. We have observed that when a software implementation exceeds 10-man days and more than 10 software components, the complexity of the software implementation usually increases significantly and this may need a different requirements approach than the one we are proposing in this thesis. It is important to note that if a micro-business is unable to adhere to the arithmetic criteria, an additional analysis may be needed to decide if our approach could still be applied or not.

As part of our characterization of a micro-business, it is also important to understand what a software component is. Software components could be characterized as defined units of computation or data which could be as small as a single procedure or as large as an entire application (Medvidovic and Taylor, 2000).

*We characterize a **software component** as an **encapsulation of a certain set of functions and data which vary in granularity as long as they could be updated, replaced, or modified without affecting other software components in a system.** Examples of such software components would range from an off-the-shelf customer management system, a website template, or a simple JavaScript line of code.*

A Commercial-off-the-Shelf “COTS” customer management system can be bought in-a-box from your local electronics store, installed on the computer at your office, and ready-to-go from there. You could replace, update, or modify this system without affecting other software

systems it is connected to. This same logic applies to a website template that you download from the internet, use for your website, and then replace it afterwards with another template. All this is done without affecting the original code of your website.

A simple JavaScript line of code can also exhibit the characteristics of a component. If the simple JavaScript line of code calculates Year 2020 Black Friday discounts with a special formula, then the line of code can be easily replaced with another that calculates Year 2021 Black Friday discounts without affecting the rest of the system.

The characterization of micro-businesses and its associated components could be used in a concrete, real-world example. For instance, let us have a first micro-business with the following characteristics: it is a bakery with five employees, five basic business processes being followed, and five hundred thousand euros of annual revenue. A second micro-business would be an online retail store with three employees, three basic business processes being followed, but has two and a quarter million euros of annual revenue. Both micro-businesses would like to implement a basic inventory system that only takes two-man days of effort and two software components. If you would use the ability of a business to collaborate on software projects as the yardstick to characterize a micro-business (Jantunen, 2010), then how would you characterize whether the first or second is a micro-business? Trying to answer this question *with arithmetic could quickly assist the stakeholders*.

If we apply our proposed characterizations then we could point out that the first business could be a micro-business. However, the second micro-business may have a bit more revenue than what the European Union would classify as a micro-business. Even after careful consideration and discussions, the second micro-business could still be classified as one. *Arithmetic could actually serve as a guide to proceed with the application of our proposal but there could always be exceptions*.

2. CONCEPTUAL MODEL

On top of providing a characterization for micro-businesses and their associated software components, it is also important to provide the context and an overview of all the concepts related to μ BRPs. The rationale for providing this perspective is *to provide an understanding of the concepts* in our approach, how they *relate to each other*, and how such perspective could *help make modeling requirements for micro-businesses more practical*. We show the conceptual model of our μ BRP proposal in Figure IV.1 which serves as a metamodel in our proposal.

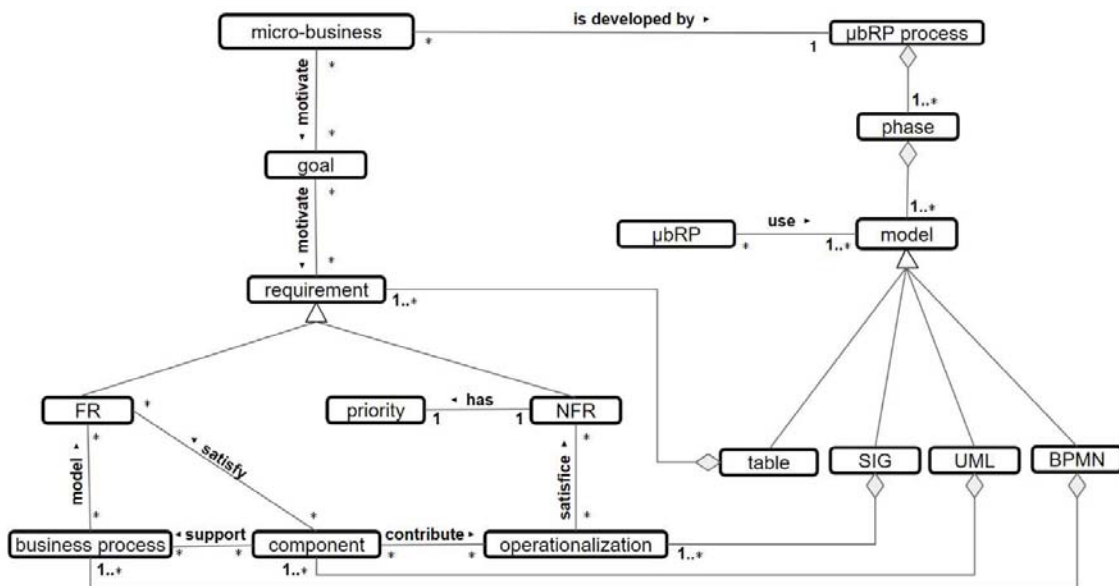


Figure IV.1 Conceptual model

We describe each and every concept in detail in the next paragraphs. As we have mentioned, we try to characterize a micro-business as a business in the real world that has less than 10 employees, less than 2 million euros in annual revenue or annual balance sheet, with a software implementation that requires less than 10-man days of effort and less than 10 associated software components. A possible example of a micro-business would be a restaurant with 3 employees, five hundred thousand euros in annual revenue, and in need of a point-of-sale software system that requires 2 software components and would take one man day to implement.

A **goal** is an objective that a micro-business would like to achieve. It is a motivation for the micro-business to exist. For example, a restaurant may want to reach an annual revenue target of six hundred thousand euros next year. This goal is set by the micro-business owner or any relevant stakeholder and is an objective or target that everyone working at the micro-business will strive for.

Goals can be refined. For instance, if the annual revenue target of the restaurant was six hundred thousand euros for next year, then the goal can be refined to be two hundred thousand euros of revenue next year coming from appetizers, three hundred thousand euros of revenue next year coming from main dishes, fifty thousand euros of revenue next year coming from desserts, and fifty thousand euros of revenue next year coming from beverages.

A **requirement** is a condition or capability needed by a user to solve a problem or achieve an objective (IEEE Computer Society, 1990). For example, in order for a restaurant to reach its goals of revenue for appetizers, main meals, desserts, and beverages, it would need to know how much customers have spent on each item. The requirement could be stated as follows: "I need to know how much my customers are spending on every item in my restaurant" or "As a micro-business owner, I want to know how much my customers are spending on items in my restaurant so that I am aware of my business reaching its revenue goals."

A **Functional Requirement**, abbreviated as FR, is a type of requirement that specifies an operation or function needed by the system and can be fully satisfied (Chung et al., 2000). For example, the requirement of knowing how much a customer spent on each item in the restaurant is a Functional Requirement because you can know, with absolute certainty, how much a customer spent on each item in the restaurant. If the customer spent five euros on a beverage in the restaurant, then the receipt and the records will show that the customer spent five euros on a beverage in the restaurant. There will be no grey area or anything debatable about this.

A **Non-Functional Requirement**, abbreviated as NFR, is a type of requirement that describes quality attributes of the system which can not be fully satisfied but only "satisfied," i.e., satisfied sufficiently (Chung et al., 2000). For example, in order for the restaurant to reach its annual revenue objective of six hundred thousand euros, it would need to improve customer experience. Part of the customer experience could include the speedy billing of customers. The requirement of having speedy billing of customers is a quality attribute of the system and is not a requirement that can be completely satisfied because it would be impossible to

ascertain whether a customer is having a speedy billing experience or not. Since experience varies from one person to another, it would be very difficult to pinpoint if this requirement is fully satisfied or not. Hence, this requirement can only be “satisficed,” i.e., satisfied sufficiently

A **priority** is the arrangement of requirements in terms of their importance. In practice, this would be the ranking of importance of NFRs made by the micro-business owner. Priorities could be useful in managing trade-offs among NFRs. For example, the micro-business owner of the restaurant may want to speed up the billing of customers with the software and make it a top priority in order to improve the overall customer experience. Integrating faster and more ways of billing customers could mean that the system may be more complex and difficult to maintain. Hence, the micro-business owner may have to prioritize the trade-offs of having few and fast ways of billing customers over a variety of ways, both fast and slow, of billing customers.

Micro-business Requirements Patterns, abbreviated as **µBRPs**, are *patterns of recurring requirements models in micro-business software projects*. In the case of the restaurant micro-business owner, there could be an µBRP for the way digital payments are collected from customers. For instance, the credit/debit card payment method has evolved from swiping to contactless over the past few years. The micro-business owner of the restaurant could take advantage of a contactless payment µBRP for his/her business.

A **µBRP process** is a procedure for developing micro-business requirements patterns. A phase is a stage or a step in the development of the µBRP process. As will be explained later in this chapter, these phases would be the observation of micro-businesses, the creation of the µBRP, the use of the µBRP, and then the reuse of the µBRP in other instances in the micro-business domain. For example, the contactless payment µBRP for restaurants would have started as an observation by software developers that kept repeating itself over and over again in various micro-business restaurants. Eventually, the µBRP could be created, used, and reused for other restaurants.

A **model** is a representation of some system whose form and content are chosen based on a specific set of concerns (Object Management Group, 2010). The representations can be in graphical form, textual form, or a combination such as a graph or a table. For example, you could show the contactless payment µBRP for restaurants using a table by listing down its requirements and how they could be met. You could also show the contactless payment µBRP for restaurants using a diagram with the flow of activities from the customer selecting his or

her contactless card up to the point where the waiter or waitress asks if the customer needs a receipt for the meal.

A **table** is an informal representation of requirements in a tabular template. A template is a combination of placeholders and linguistic formulas used to describe something in a particular domain, e.g., templates facilitate communication among practitioners and provide a helpful guide for beginners (Segura et al., 2017). In the micro-business restaurant, you could list down the requirements of the micro-business owner and how they would be met. In a μ bRP, this would be made up of a tabulation of questions, options, choices, and priorities.

Softgoal Interdependency Graphs (Chung et al., 2000), abbreviated as SIGs, is a semi-formal notation for modeling NFRs. In the restaurant micro-business, you could represent the quickness of the billing of customers with a softgoal model.

An **operationalization** is a measurable (or estimate-able) element which satisfices refined Softgoals. They are solutions in the form of operations, processes, (infra)structures, that would satisfice the NFRs (Chung et al., 2000) and contribute to the reuse or development of software components that satisfy the FRs. In the restaurant micro-business, for instance, the speediness of customer payment softgoal could be operationalized using a touch-free payment device which forms part of the software system. The touch-free payment device would have measurable characteristics such as processor speed that would then contribute to the performance of the software components of the system.

The **Unified Modeling Language** (Object Management Group, 2009), abbreviated as UML, is a semi-formal notation used to model software components in this thesis. In the restaurant micro-business, UML could be used to model the software components that would facilitate the collection of payments from customers in a contactless manner.

A (software) **component** is 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 (Medvidovic and Taylor, 2000). An example of a component in the restaurant micro-business would be the contactless payment component interfacing as a provider to a required interface connection in the point-of-sale system.

The **Business Process Modeling Notation** (Object Management Group, 2008), abbreviated as BPMN, is a semi-formal notation to model business processes in this thesis. A

business process is a step-by-step procedure of a business which achieves a particular objective. The BPMN models in this thesis attempt to model the business processes from the point-of-view of the customer (customer-centric view). An example of a business process in the restaurant micro-business which can be modeled with BPMN would be the step-by-step procedure of activities for collecting payment from a customer in a contactless manner. A summary of the concepts as definitions is provided in Table IV.2.

Table IV.2 Definition of Concepts in the Conceptual model

Concept	Definition
Micro-business	A business in the real world that has less than 10 employees, less than 2 million euros in annual revenue, with a software implementation that requires less than 10-man days of effort and less than 10 associated software components
Goal	An objective that a micro-business would like to achieve
Requirement	A condition or capability needed by a user to solve a problem or achieve an objective (IEEE Computer Society, 1990)
FR	A functional requirement of the system which can be fully satisfied (Chung et al., 2000)
NFR	A non-functional requirement about the quality of the system which can not be fully satisfied but only "satisficed," i.e., satisfied sufficiently (Chung et al., 2000)
Priority	The arrangement of elements in terms of importance. This would be the ranking of

	importance of NFRs made by the micro-business owner.
µbRP	The abbreviation for Micro-business Requirements Pattern which are patterns of recurring requirements models in micro-business software projects
µbRP process	A procedure for managing µbRP
Phase	A stage or step in the µbRP process
Model	A representation of some system whose form and content are chosen based on a specific set of concerns (Object Management Group, 2010)
Table	An informal representation of requirements in a tabular template. In an µbRP, this would be the tabulation of questions, options, choices, and priorities.
SIG	Softgoal Interdependency Graph (Chung et al., 2000) is a semi-formal way of modeling NFRs
Operationalization	A measurable (or estimate-able) element which satisfies refined Softgoals. They are solutions in the form of operations, processes, structures, that would satisfy the NFRs (Chung et al., 2000) and further connect to the FRs.
UML	The Unified Modeling Language (Object Management Group, 2009) is a semi-formal way of modeling software components

Component	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 (Medvidovic and Taylor, 2000)
BPMN	The Business Process Modeling Notation (Object Management Group, 2008) is a semi-formal notation of modeling micro-business processes
Business Process	A step-by-step procedure of a business which achieves a particular objective

After defining the concepts, let us now discuss the relationships among them. The **Micro-business - Goal** relationship is characterized by having many Goals motivating many Micro-businesses. In the example of the restaurant micro-business, the micro-business owner could have many goals, one of which would be that he or she wants to achieve six hundred thousand euros of annual revenue in the next year. Another micro-business such as a clothes retail store could also have this goal of achieving six hundred thousand euros of annual revenue next year.

The **Goal - Requirement** relationship is characterized by having many requirements that are motivated by many Goals. There are techniques that aid in turning goals into requirements (Kotonya & Sommerville, 2003) (Respect-IT, 2007) and vice-versa (Cardoso et al., 2011). An example of how goals are turned into requirements is provided on pp. 128 of (Kotonya & Sommerville, 2003) and in the goal modeling section of (Respect-IT, 2007) and an excerpt is shown below in Figure IV.2. Our conceptual model stems from the review, study, and analysis of these previous works.

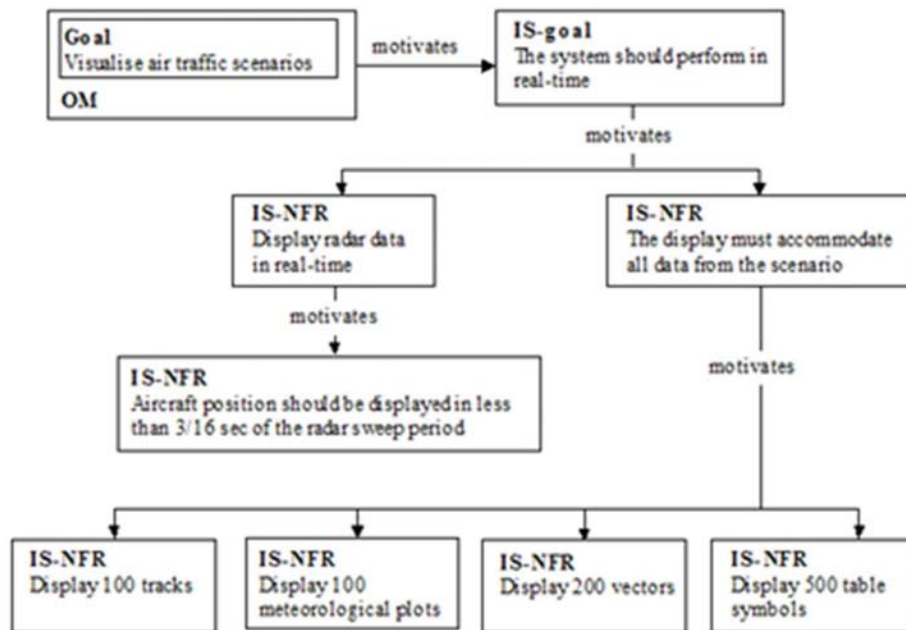


Figure IV.2 Turning Goals into Requirements by Kotonya and Sommerville, 2003

An example of turning goals into requirements in the restaurant micro-business would be the goal of obtaining six hundred thousand euros of annual revenue next year, being refined into the goal of obtaining three hundred thousand euros of annual revenue next year from main dishes, and then finally into the requirement of being able to identify how much customers spent on main dishes.

The **Requirement - FR** relationship is characterized by the possibility that a requirement could be a functional requirement. At the same time, the **Requirement - NFR** relationship is characterized by the possibility that a requirement could be a non-functional requirement. An example of this in the micro-business restaurant is that if you come up with a list of requirements for this micro-business, a requirement would either be functional or non-functional. The requirement of being able to identify how much customers spent on main dishes is a functional requirement while the requirement of having speedy billing is a non-functional requirement.

The **NFR - Priority** relationship can be characterized by the non-functional requirement never being able to be fully satisfied and instead being prioritized to further clarify the requirement. An NFR could have a priority. For example, in the restaurant micro-business, the

speediness of billing could take priority over the variety of billing methods for customers. This way, the focus is more on faster payments from customers over the ability of the customer to pay with several methods.

The **NFR - Operationalization** relationship can be characterized by many Operationalizations satisficing many NFRs. An example in the micro-business restaurant would be the operationalization of the quickness of customer payment NFR. There could be several touch-free devices that could contribute to the quickness of customer payment and at the same time, a touch-free device could also be an operationalization that satisfies another NFR such as variety of customer payment.

The **FR - Business Process** relationship can be characterized by many Business Processes modeling many Functional Requirements. An example in the micro-business restaurant would be the business process of collecting payment with contactless methods as a model for several functional requirements such as the customer must be able to pay with his or her credit card in a contactless manner and another requirement that the customer must be able to pay with his or her mobile phone in a contactless manner.

The **FR - Component** relationship can be characterized by many Components satisfying many FRs. An example of this in the micro-business restaurant would be the requirement that the customer must be able to pay with his or her credit card in a contactless manner and the software components which satisfy this requirement are the point-of-sale system and the contactless payment component. In addition, a software component such as the point-of-sale system can satisfy several FRs such as the customer must be able to pay with his or her credit card in a contactless manner and that the micro-business owner must be able to see what items customers are paying for the most.

The **Business Process - Component** relationship can be characterized by many Components supporting many Business Processes. In the micro-business restaurant example, the business process could be the process of collecting payment in a contactless manner and the supporting components would be the point-of-sale system and the contactless payment component. In addition, the point-of-sale system can support several business processes such as the process of collecting payment in a contactless manner and the process of collecting order information made by each customer.

The **Component - Operationalization** relationship can be characterized by many Components implementing many Operationalizations. In the micro-business restaurant example, speediness of customer payment could be operationalized with a touch-free device which will contribute to the speediness of the point-of-sale software system.

The **Micro-business - μ BRP** Process relationship can be characterized by many micro-businesses being developed by many μ BRP processes. In the restaurant micro-business example, the μ BRP process of collecting payment in a contactless manner could be used to develop several micro-businesses. Some of which could be the restaurant micro-business itself or a clothes retail store micro-business that would also like to accept contactless forms of payment when customers purchase at their store.

The **μ BRP Process - Phase** relationship can be characterized by a μ BRP process made up of many phases. In the restaurant micro-business example, the μ BRP process could be made up of phases such as: observation of the payment collection in a contactless manner, creation of the payment collection in a contactless manner μ BRP, and the use and reuse of the payment collection in a contactless manner μ BRP.

The **Phase - Model** relationship can be characterized by a Phase being modeled in many ways. In the restaurant micro-business example, the use and reuse of the payment collection in a contactless manner μ BRP could be modeled, depending on the phase, using a table to help decide on choices and priorities, a BPMN diagram to model the payment process, or a UML activity diagram to show the selected component to be reused.

The **μ BRP - Model** relationship can be characterized by a defined μ BRP resulting in several usable models as mentioned in the Phase-Model relationship. In the restaurant micro-business example, there could be a point-of-sale μ BRP that could be modeled using a table, BPMN, UML, and SIGs diagrams.

The **Table - Requirement** relationship can be characterized with requirements that can be placed in a table in many different ways. In the restaurant micro-business example, the requirement that the customer must be able to pay with his or her credit card in a contactless manner could be placed in a table in many different ways. You could state the requirement as-is, in the form of a User Story as done in Agile Methodologies, i.e., "As a role, I would like to do an action, for this reason," or in a question-answer form as will be discussed in Section 3.1.2 of this Chapter.

The **BPMN - Business Process** relationship could be characterized with business processes being modeled in many ways using BPMN. In the restaurant micro-business example, the business process of collecting payment in a contactless manner could be modeled using BPMN in many different ways, from left to right, top to bottom, and even using different icons to express the process to the readers.

The **SIG - Operationalization** relationship could be characterized by operationalizations being modeled in many ways using SIGs. In the restaurant micro-business example, the NFR of the speediness of customer payment which is operationalized with a touch-free payment system can be modeled using SIGs in many different ways, from refinements from one NFR such as the speediness of customer payment or a refinement from several other NFRs such as the ability of the customer to pay in several different ways.

The **UML - Component** relationship could be characterized by a component that could be modeled in many ways using UML. In the restaurant micro-business example, the contactless payment component could be modeled in several different ways in UML. The contact payment component could be modeled as a component with a provided interface or a required interface, depending on how it connects to the other components in the system. Also, some UML component diagrams are more detailed than others and the modeler could decide how much detail should be put in the UML component diagram, depending on the kind of readers or users of the diagram. The relationships among all these concepts are provided in Table IV.3.

Table IV.3 Relationships in the Conceptual model

Relationship	Definition
Micro-business - Goal	Many Goals motivate Many Micro-businesses
Goal - Requirement	Many Requirements are motivated by Many Goals. Requirements could be turned into goals (Cardoso et al., 2011) and vice-versa (Kotonya & Sommerville, 2003) (Respect-

	IT, 2007). A detailed description of this relationship is explained in the text.
Requirement - FR	A requirement can be a functional requirement
Requirement - NFR	A requirement can be a non-functional requirement
NFR - Priority	Since a non-functional requirement can not be fully satisfied, they could be prioritized instead to further clarify the requirements and manage trade-offs. An NFR can have a priority.
NFR - Operationalization	Many Operationalizations can satisfice Many NFRs
FR - Business Process	Many Business Processes are modeled for Many Functional Requirements
FR - Component	Many Components satisfy many FRs
Business Process - Component	Many Components support many Business Processes
Component - Operationalization	Many Components could implement many Operationalizations
Micro-business - μ bRP Process	Many Micro-businesses can be developed with many μ bRP processes
μ bRP Process - Phase	A μ bRP process is made up of many phases
Phase - Model	A Phase can be modeled in many ways

µbRP - Model	A µbRP can be modeled in many ways
Table - requirement	Requirements can be placed in a table in many different ways
BPMN - business process	A business process can be modeled using BPMN in many ways
SIG - operationalization	An operationalization can be modeled using SIGs in many ways
UML - component	A component can be modeled using UML in many ways

3. THE µbRP

This section will provide a detailed description of a µbRP, its structure and characteristics, using the restaurant micro-business example. To help describe solutions, an adaptation of SIGs for the micro-business domain and a proposed catalog of operationalizing methods are also described in this section.

3.1 The Description of a µbRP

A µbRP is made up of:

- (1) **Name:** A word or set of words by which a µbRP is known, addressed, or referred to. The name is usually placed in the first section of the table or in the BPMN model of the µbRP.
- (2) **Context:** A brief way of describing the µbRP in business language by stating some of its business activities, e.g., restaurant micro-business makes food for customers. The context description(s) is usually placed in the first section of the table.
- (3) **Keywords:** A word or words by which a µbRP can be indexed and then searched in a repository. This is usually found in the first section of the table.

- (4) **Problem:** The description of the problem in natural language including the main goals and requirements that have to be satisfied (or satisfied) with the application of the pattern solution. The problem can be presented informally with tables and notes and semi-formally with BPMN. There are factors, e.g., constraints (e.g., maximum payment time), considerations/conditions (e.g., restaurant peak hours), circumstances (e.g., ability to hire additional people during peak hours), NFRs (e.g., responsiveness), etc., in the problem that affect and guide to the feasible alternatives, i.e., choices of options/modes, for the solution.
- (5) **Solution:** The tables, models, and notes that plan to satisfy (or satisfy) the requirements of the micro-business and eventually determine the system and software to be implemented for the micro-business based on certain factors. The solution involves a table with FRs and BPMN models that show options/modes on how to satisfy the FRs, NFRs identified and ranked made by the micro-business owner and software developer/analyst to help prioritize the NFRs. A catalog of operationalizing methods is also proposed to help in modeling the NFRs. The operationalizing methods support systems and software components which could possibly be reused and contribute to faster implementation. Notes are also made throughout the requirements analysis to help in providing solutions for the problems.

The parts that make up a μ BRP are explained in further detail in the next paragraphs. To provide an overview, the parts are shown in Figure IV.3 and summarized in Table IV.4, which are later explained in detail in the following paragraphs.

Description of a Micro-business Requirements Pattern "µbRP"

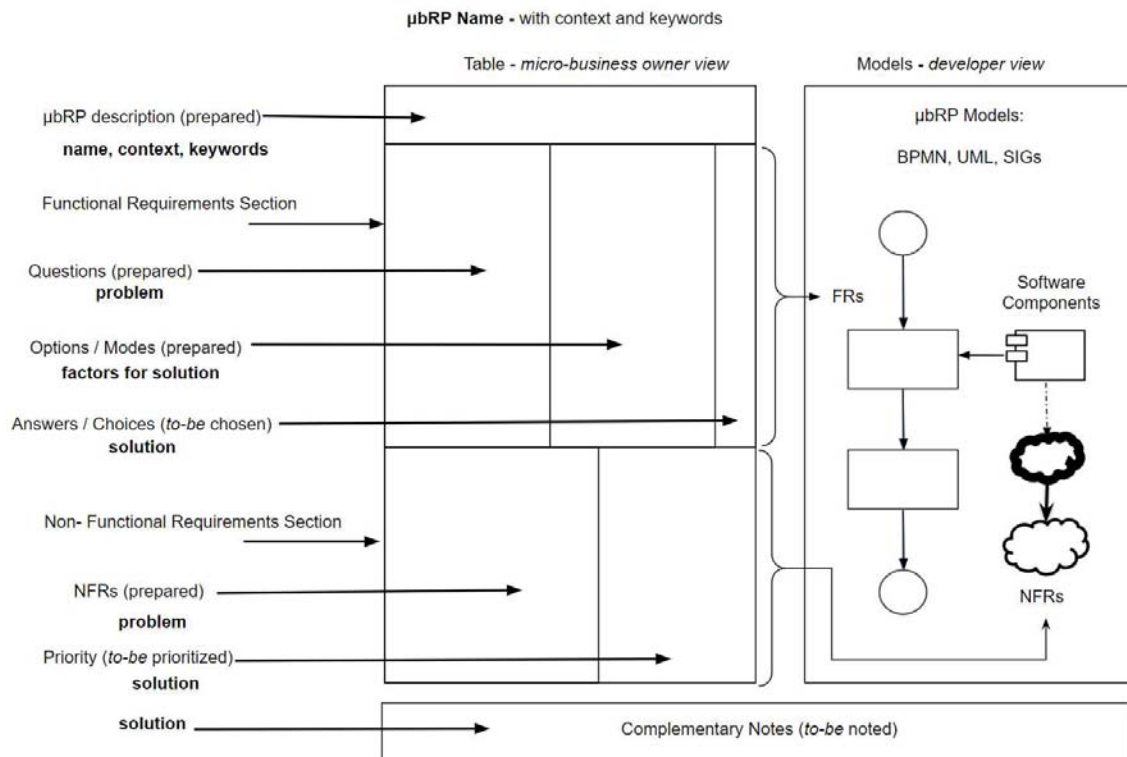


Figure IV.3 Overview of the Description of a µbRP showing prepared and to-be parts

Table IV.4 Description of the parts of a µbRP

Part	Description
Prepared Parts	Parts of the µbRP that are ready before the requirements elicitation meeting
To-be Parts	Parts of the µbRP that will be chosen, decided, or noted during the requirements elicitation meeting
Description Table Section	Provides the name, overview, context, and keywords of the µbRP
Name	A word or set of words by which a µbRP is known, addressed, or referred to
Overview/Context	A brief way of describing the µbRP in

	business language by stating some of its business activities
Keywords	A word or words by which a μ BRP can be indexed and then searched in a repository
Functional Requirements Table Section	Questions, possible answers, and responses which are easy to understand and fill up for the micro-business owner. The purpose of this part is to understand the “hard” problem better.
Questions	Functional requirements which are in a question form that is easy to understand for micro-business owners
Options/Modes	Possible ways or manners of functional requirements which a micro-business owner can select from. These alternatives are also influenced by factors.
Answers/Choices	Possible ways or manners of functional requirements which a micro-business owner has selected to support the micro-business. The selection is used for the instantiation of the pattern.
Non-Functional Requirements Table Section	Priorities which are set by the micro-business owner given the constraints. The purpose of this part is to understand the “soft” problem better.
List of NFRs	List of quality requirements of the system that are important but however, can not be fully satisfied but instead, “satisfied” or met “good enough”
Priority of NFRs	The rank of priorities for the micro-business owner, software analyst/developer, and other stakeholders
Complementary Notes	Additional information provided by the micro-business owner and developers that will aid in the software implementation
BPMN Models	Business process models that help developers understand the μ BRP solution
SIGs Models	Models of Non-Functional Requirements that help developers understand the μ BRP solution
UML Models	Software component models that help developers understand the μ BRP solution

3.1.1 μ BRP Table

We first describe the **μ BRP Table** which is **an informal representation of requirements in a tabular template**. A **template** is a combination of placeholders and linguistic formulas used to describe something in a particular domain, e.g., templates facilitate communication among practitioners and provide a helpful guide for beginners (Segura et al., 2017).

Micro-business owners see the μ BRP Table during requirements elicitation. In requirements engineering, although there is a clear distinction between the roles of a requirements analyst and a software developer, this role is *normally played by one person*, usually the software developer, in micro-business software projects. This usually happens due to the *resource and budget constraints of micro-business projects* which normally cannot afford to have a dedicated requirements engineer in their implementations (Azar et al., 2007).

Table IV.5 shows a shortened version of the FR section of a requirements table for the restaurant micro-business example. The complete table can be found in Appendix A.9. The table is brought to the requirements elicitation meeting and is made up of three main sections. The topmost section is the description section of the μ BRP where the following details can be found: the **name** of the μ BRP, a brief description of the **context** of the μ BRP, and **keywords** of the μ BRP. Keywords are indexed in μ BRP repositories so that software developers may search for them easily. Although there is no limit as to the number of keywords that can be placed, creators of the μ BRP tables are recommended to keep this at a minimum for precision and searchability purposes. The topmost section is *prepared*, meaning that it was ready before the requirements elicitation meeting between the software developer and the micro-business owner.

Table IV.5 Restaurant µbRP FR Table Section (shortened)

Restaurant µbRP		
description: µb sources ingredients, µb produces food for customers to eat, and then customer pays.		
keywords: ingredients, food, restaurant, delivery		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb gather the ingredients for the food to be served?	buy at market / have ingredients delivered by third party / other select as many that apply	
(b) How can customers order food at the restaurant µb?	in-house menu / phone / website / mobile app / other select as many that apply	
(c) How do customers eat the food from the restaurant?	in-house / take away / delivery select as many that apply	
(d) How can customers pay the restaurant µb?	credit / debit / Paypal / other, select as many that apply	
(e) What roles do the µb-side users have on the system?	admin / manager / user / other select as many that apply	

3.1.2 Functional Requirements Table Section

The middle section of the µbRP table is the functional requirements section and its purpose is to elicit the functional requirements of the micro-business. This section refers to a detailed specification of the requirements that the µbRP is trying to solve, in line with the general description of the **problem**. In the first column of the middle section, functional requirements are expressed in a business-like, question-answer format. Micro-business owners prefer to use their natural language and sketches to express their requirements instead of using technical software jargon (Macasaet et al., 2011). The questions and answers are made so that they are comprehensible for micro-business owners in a straightforward way. Like the top section, the first column in the middle is *prepared* before the requirements elicitation meeting.

We propose a technique to transform functional requirements into a business-like, question-answer format. When transforming functional requirements to business-like, question-answer format, the first step is to express the functional requirements in declarative, straightforward sentences, e.g., “record cash sale” and “display total cash sales” is transformed into “the user is able to record a cash sale and then display cash sale totals.” Then, when the sentences are constructed, they are turned into a question form which can be answered in simple, straightforward ways, e.g., the question “does the micro-business owner need to record and display cash sales?” can be answered with a “Yes” or “No” response. Figure IV.4 shows the transformation of functional requirements to business-like, question-answer format. However, direct specification of the functional requirements is another

alternative, e.g. “record cash sale”, where the writer can choose between alternatives according his/her preferences.

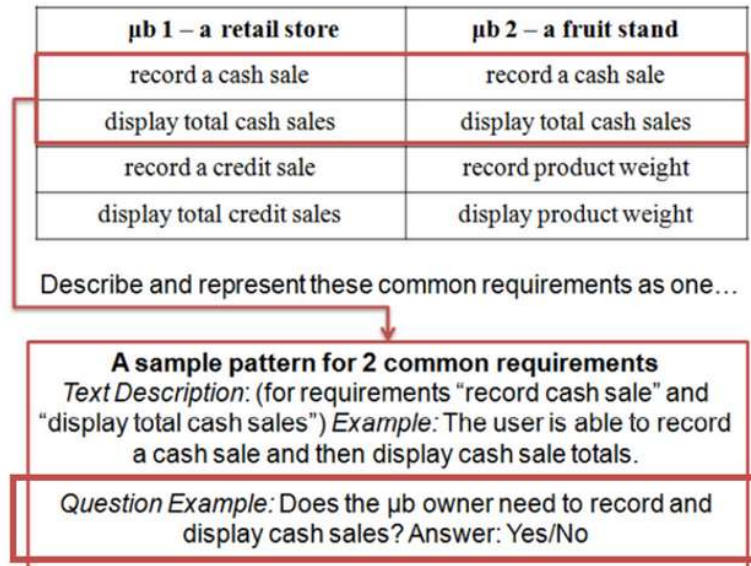


Figure IV.4 Transformation of FRs to business-like, question-answer format

In the second column of the middle section of the µbRP table, the possible answers to the business-like, question-answers, are found. These candidate answers correspond to the options or modes of the pattern which point to the **solution**. For example, if a process is to “query site visitor data” then some possible options or modes would be to “query log files” or to “query page tag files.”

For further clarity, another example of a pattern option or mode could be seen in the restaurant micro-business requirements table in Table IV.5. For question (d) How can customers pay the restaurant?”, the possible modes of payment are by credit card, debit card, PayPal, etc. It is easy to understand the concept of modes by relating it to the phrase “modes of payment,” something heard on a day-to-day basis. A customer is required to pay for what is purchased but *the manner or mode* for paying is something that can vary depending on the situation. Like the first column of the middle section, the second column of the middle section is *prepared* before the requirements elicitation meeting. However, it is important to note that in the future, modes of payment are going to change and *some prepared parts of the pattern will have to change and vary with the times*.

In the third column of the middle section of the μ BRP table, the responses of the micro-business owner are placed under the column “choices.” Based on the choices for the solution, the μ BRP will guide developers to **solutions** as will be explained. The final answers of the micro-business owner are required for the different instantiations of the pattern. The instance of the pattern is its application in a real-world micro-business case. The third column of the middle section is a part *to-be* chosen, meaning that the values under this column are chosen by the micro-business owner with the software developer *during* the requirements elicitation meeting. Figure IV.5 provides a model of the relationships among a question, mode/option, and choice/answer.

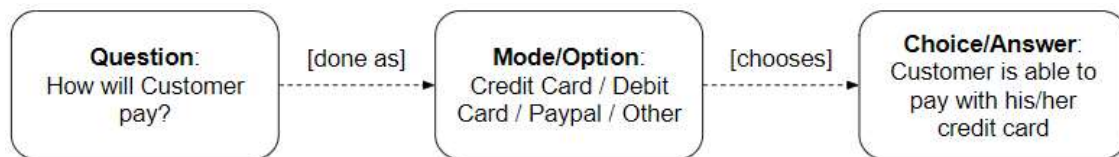


Figure IV.5 From question, to mode/option, to choice/answer

3.1.3 Non-Functional Requirements Table Section

The bottom section of the μ BRP table is the NFR section. This section also refers to the **problem description** (even challenges) that the μ BRP is trying to address (or confront). In every μ BRP, the most commonly occurring NFRs in the given micro-business are enumerated and grouped for the micro-business owner, the micro-business customer, and the software developer. These priorities are ranked from the most important to the least important; where 1st is the label used in the table for the most important priority, and 2nd, 3rd, etc... is used up to the least important. NFRs are ranked instead of given a definite answer since these requirements are “satisficed” instead of satisfied. While the NFRs are prepared, they are *to-be* prioritized by the micro-business owner with the software developer *during* the requirements elicitation meeting for pattern instantiation. Table IV.6 shows a shortened version of the NFR section of the restaurant micro-business requirements table.

Table IV.6 Restaurant µbRP NFR Table Section (shortened)

Non-Functional Requirements Section, to be ranked in terms of priority by µb owner	
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.	
- Micro-business side -	
	µb Ranking
How important is _____ to the µb owner?	
system speed (responsiveness)	
affordability of software system (cost)	
deployment time (short implementation period)	
user-friendliness of the software system, admin side (usability)	
ease of maintaining the software system	
security of the software system (data protection)	
exactness (preciseness) of data	
How important is _____ to the customers of the µb?	
quickness (reponsiveness) of the food service	
quickness (responsiveness) of customer payment	
quickness (reponsiveness) of the software system (online restaurant site/app)	
user-friendliness of the software system, customer side (online restaurant site/app)	
security of the software system (transactions made on online restaurant site/app)	
availability of the food on the menu (availability of ingredients)	
delivery time of the food (fast food delivery)	

The priorities decided in the NFR section help in *managing trade-offs in the solution*. The relationship of the NFRs to the activities allow the developers and the users to see *dependencies* which would later on *help in understanding priorities and trade-offs during implementation*. More priority will be given to the operationalizations that are related to softgoals with higher priority. For example, if speedy customer payment *takes priority over* the variety of customer payment, then the designed solution will prioritize - operationalizations with lesser payment components, also in relation with lesser choices for options/modes in the functional requirements, that perform faster than others - *instead of* - operationalizations with many payment components for variety which could result in slower customer payments.

3.1.4 BPMN, UML, and SIGs Models

BPMN models business processes, SIGs model NFRs, and UML models software components. These models help stakeholders in requirements elicitation and analysis and developers in modeling, designing, and implementing the solution in the µbRP. The point-of-view is an important aspect for using different models since a single model is not capable of providing various points of views in equal measure at the same time (Kalenborn, 2010). Hence, specific models are used for the µbRPs to provide further clarity, optimal viewing, and various perspectives for its stakeholders which are usually the owners, analysts, and developers involved in the micro-business software project.

The BPMN models are used to further describe the business processes involved in the μ BRPs. In particular, **the BPMN models are used to show the flow of activities for the functional requirements**. In Figure IV.6, the flow of activities from the placement of the order of the customer to providing feedback is modeled.

First, the customer orders food and this is recorded in a sales system. Then the restaurant micro-business uses ingredients in the inventory to make food. If there are not enough ingredients, the restaurant micro-business must source them from suppliers. Monitoring the availability of ingredients can be done through an inventory management system. Purchasing of ingredients from suppliers can be done through a procurement system. Once the food is prepared, the customer receives the food and then pays for it. In some cases, the customer pays for the food when it is ordered. Finally, the customer provides feedback based on his or her experience with the restaurant micro-business. Customers, the micro-business, and the suppliers could be able to log in to the restaurant micro-business system if allowed.

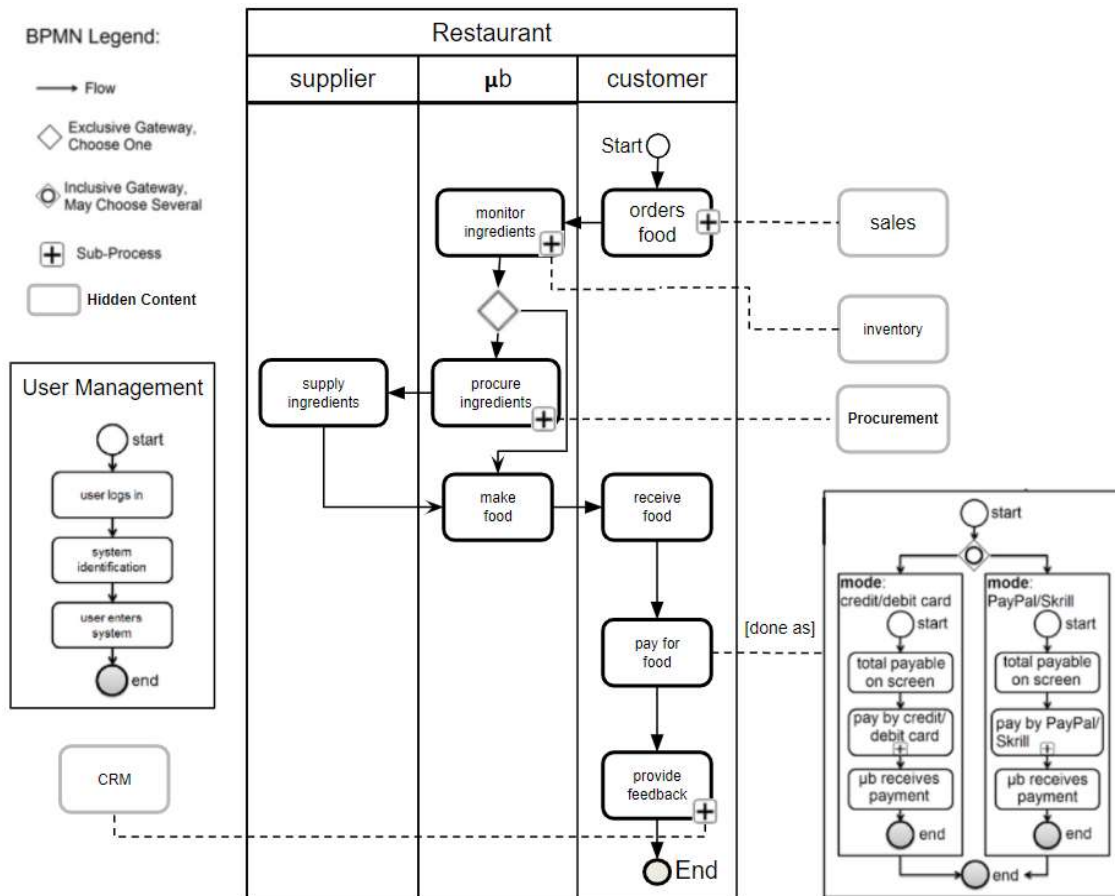


Figure IV.6 BPMN Diagram for the Restaurant Micro-business (simplified)

Even if the BPMN models are considered readily understandable, (Mendling et al., 2010) suggest that the purposeful use of customized labels in business process models could help representations be more comprehensible (or even, more technically relevant). In Figure IV.6 in the customer BPMN pool lane, there is an activity “pay for food.” A customized label [done as] is used to indicate that there is a mode for this activity. Allowing the users of the models to see the modes of business process patterns is helpful in understanding the functional requirements and how they could be done.

When the business process pattern is done in practice, this is considered as an **instance** of the pattern. This means that the mode of paying for food is *performed in practice* when the customer pays for the food using a credit card or a debit card. During the instance of a pattern, e.g., performing a payment in practice, several dependencies come into play such as the software systems involved, the computing devices, and even the people who are part of collecting and making such payments. In addition to understanding the FRs which are business processes modeled in BPMN, *understanding the business processes alongside the NFRs provide better understanding as they pertain to holistic quality attributes of the entire system involved.*

Since BPMN is not intended to model NFRs (Zhao et al., 2012), the prioritized NFRs (or softgoals) are modeled using SIGs. SIGs were originally proposed to model NFRS (Chung et al., 2000). Combining both BPMN and SIGs through an “operationalization target link” allows developers and users to see how the NFRs *relate* to the activities in a business process.

For the restaurant micro-business, a prioritized NFR “quickness (responsiveness) of payment” is refined into an operationalization which is a measurable (or estimate-able) element that *satisfices* the NFR. In the model in Figure IV.7, operationalizations point to the NFRs they are satisficing. The operationalizations could satisfy the NFR in the form of operations, processes, or (infra)structures. A Mobile Payment Device could satisfice the NFR quickness of payment and in the case of a faulty Mobile Payment Device, it could negatively affect satisficing the NFR quickness of payment, e.g., make the payment slower or even impossible.

The Mobile Payment Device is modeled as an operationalization in SIGS. Operationalization icons are discussed in detail in the next subsection 3.2. From the operationalization, it would support the business activity of paying for food. This means that if the business process activity of paying for food would be done through a credit or debit card,

then the Mobile Payment Device would be supporting this business process activity. Without the Mobile Payment Device, it would be difficult or even impossible to perform a payment with a credit or debit card. The operationalization is modeled in SIGs and then connected with the business activity, pay for food, through an operationalization target link. The operationalization target link is modeled with a *dash-dot-dash* arrow, connecting the operationalization and the business activity. This is shown in Figure IV.7.

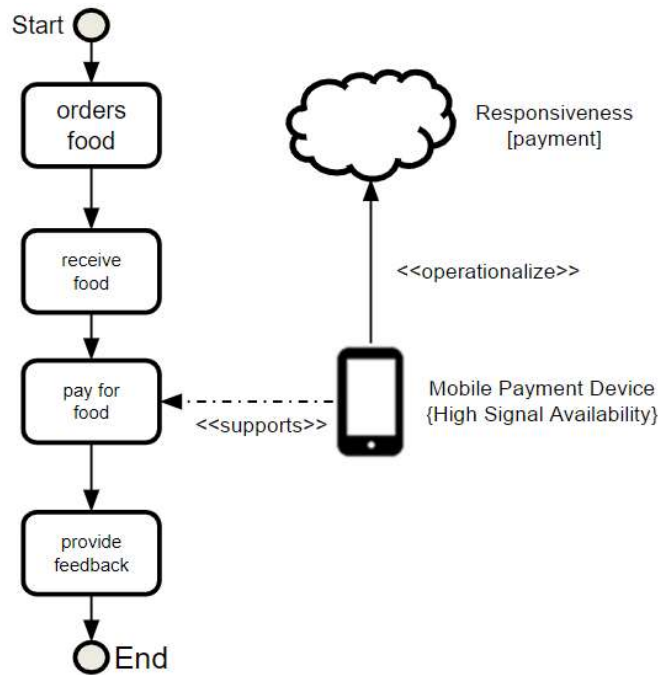


Figure IV.7 Combining BPMN and SIGs through an operationalization target link

In Figure IV.7, a broader, more holistic, perspective is provided for the business process activity “paying for food”. The metric of {high signal availability} for the mobile payment device is shown to directly relate to the NFR of the responsiveness of the payment through a solid directional arrow and also to the business process activity of the customer paying for food. Therefore, a viewer of this model would see that if the mobile payment device does not have high signal availability, the NFR responsiveness of payment and the execution of the payment for food by the customer can be negatively affected.

The perspective provided by the BPMN and SIGs models helps both the micro-business owner and software developer decide on possible operationalizations and how they will be implemented. In Section 3.2.2, a catalog of operationalizing methods is proposed to help the

micro-business owner and the software developer collaborate on possible operationalizations which could satisfy the NFRs. For example, when the NFR of responsiveness is made a top priority by the micro-business owner as shown in Figure IV.7, the software developer could refer to the catalog of operationalizing methods and from there, suggest a mobile processing payment device with high signal availability to satisfy the NFR of responsiveness and to support the business process of paying for food. The micro-business owner understands this relationship through the explanation of the software developer with the aid of the model. Hence, the micro-business owner can make better decisions to invest in mobile payment devices with high signal availability instead of constantly doubting about such an investment. Using the proposed catalog of operationalizing method icons is discussed in further detail Section 3.2.2.

Some micro-business owners already find the models with BPMN and SIGs sufficient for analysing requirements. Micro-business owners are usually not interested in knowing the technical implementation details such as the number of lines of code in a component or the programming language that is being used by the developers. Micro-business owners are more interested in the value that the software would bring to the business over the technical implementation details.

However, software developers could make use of more technical details to guide them during implementation. Although analysing requirements are important, software developers are also interested in ways to meet such requirements through software. For modeling the proposed software which could meet such requirements, software developers could express the associated software components in UML. When more models with UML are created by the developers, they are stored in a repository which can then be later referenced and reused by developers in future projects. When there are similar requirements and similar software components for meeting such requirements, there could be opportunities for reuse and speeding up implementation. Reusing components in repositories are discussed in more detail in Section 5.2.

In addition, the UML models bridge the gap between the SIGs and BPMN models, linking the FRs and the NFRs, providing an even more holistic view. In Figure IV.7, it is shown that the Mobile Payment Device supports the Payment for Food but this would lead to the next question for the software developer: “how could we implement that with software?” The UML models suggest solutions to this initial question.

In the restaurant micro-business, from the Mobile Payment Device which is represented as an operationalizing method (SIGs), an *operationalization target link* is used to represent how the Mobile Payment Device <<contributes>> to the performance of the Payment Component which is expressed in UML. This means that if the Mobile Payment Device does not have high signal availability, the way the Payment Component functions will be affected. In extreme cases where there is no availability of signal, the Payment Component may also cease to function.

From the Mobile Payment Device, the operationalization target link contributes to the Payment Component expressed in UML, and then continues to support the specific business process of Payment for Food. As shown in the model, the Payment Component <<supports>> the business process of Payment for Food because through the Payment Component, the Payment for Food through the use of a credit or debit card can be made. Without the Payment Component, such activity between the micro-business and the customer may not be possible.

Figure IV.8 shows the relationships from NFRs (payment responsiveness), to operationalizations (Mobile Payment Device), to UML (Payment Component), and finally to BPMN (Payment for Food business process activity). The relationships in Figure IV.8 are based on the relationships in the conceptual model in Figure IV.1 which is found at the beginning of this chapter.

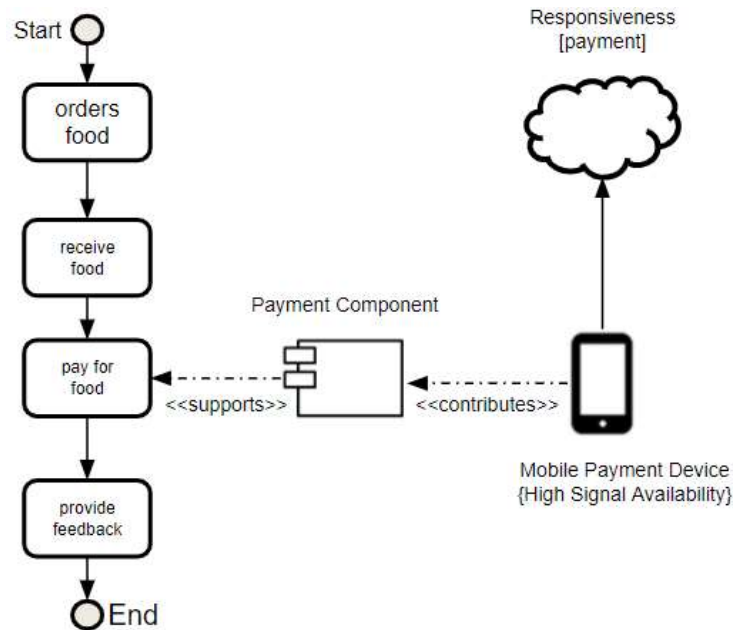


Figure IV.8 Combining UML with BPMN and SIGs

The purpose of the UML model is to *provide technically relevant information* for software design to software developers, aiding them in software component reuse. Information such as the required and provided interfaces is shown for every software component as shown in Figure IV.9. In addition, the components should also include other notes such as what kind of FRs it satisfies and NFRs it can satisfy, based on the models. This information guides the software developer when searching for other associated software components in the repositories, aiding in reuse. These technical diagrams are created by software developers who have previously developed the associated software components. If there are no components or UML diagrams in the repository, then these are developed / created by the software engineers. The process of creation, use, and reuse is discussed in more detail in Section 5, managing μ RPs.

In Figure IV.9, if the software developer is using the Payment Component, the UML model shows that it is providing an interface to the Point-of-Sale POS component. The UML model also shows that the POS Component requires the Payment Component for certain functions. Hence, when the software developer uses the Payment Component, the possibility of (re-)using another component such as the POS Component is shown in the UML model as well.

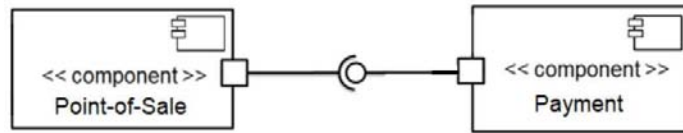


Figure IV.9 A UML Component Diagram showing required and provided interfaces

3.1.5 Complementary Notes

Even if requirements are listed down in a table and modeled, they *could continue to be ambiguous and/or complex* for many micro-business owners and software developers. To avoid confusion and prevent misunderstandings, μ BRP **complementary notes** could be made by the software developer **and** micro-business owner to ensure that the requirements are met. These complementary notes can be *made at any point in time* in the requirements process and then *compiled afterwards*. The μ BRP complementary notes are part of the μ BRP solution as they are vital guides for the micro-business owner and the software developer for implementation. They are *to-be* noted in a sense that they are written down depending on each particular micro-business software project implementation. Below are examples of complementary notes for the restaurant micro-business example.

Complementary Notes for the Restaurant Micro-business

- The operating hours of the restaurant are everyday except Monday, from 12 noon to 11 in the evening. The system should be available and working at these times.
- Peak hours of the restaurant are from 1230 pm to 230 pm and 7 pm to 10 pm. Expect the system to have more activity during these peak hours.
- The restaurant micro-business is able to hire additional staff during peak hours.

3.2 Adapting SIGs models for the micro-business domain

Since the proposal of SIGs in the NFR Framework in 2000 (Chung et al., 2000), there have been hardly any industrial empirical studies regarding its comprehensibility and adaptability. In the domain of micro-businesses, our studies on SIGs as detailed in the next chapter, would be one of the first to be done. This *initial* study was done from June 2013 to December 2013, where we conducted and recorded sixteen one-on-one interviews with micro-business owners (in Manila (Philippines), Dallas, Texas (United States of America), and Granada (Spain)) to find out if SIGs were comprehensible to them.

This *initial* study had favorable results. Although the SIG samples were not too complex, micro-business owners were still able to provide relevant responses when we asked them for their interpretations. All sixteen interviewees provided us with valid interpretations of the SIG diagrams. Most of them described how required infrastructure could be traced to business goals through NFRs. In fact, the proposed SIGs adaptations were inspired by valuable comments from the micro-business owners interviewed, such as, “why are they (the operationalizing methods) still clouds (in bold)?”

In the 90's, it would have been difficult to imagine a micro-business on a remote island using a computing device to sell their goods. Nowadays, even the most isolated micro-businesses in remote islands are using mobile computing devices to manage their orders and inventory (Macasaet et al., 2019). Given the increased reliance of businesses on computing devices and evolving computing paradigms, the way micro-businesses will operate will involve more and more seamless integration of both the physical and digital worlds (Georgakopoulos & Jayaraman, 2016). This evolution is *not* expected to slow down and is even expected to speed up in the coming years (Tan & Wang, 2010).

3.2.1 Operationalizing Methods

It is unlikely that the application of generalist requirements techniques (such as SIGs) be directly applicable in the micro-business domain without any modifications (Aranda et al., 2007; Solemon et al., 2009; Bürsner & Merten, 2010). We propose to adapt and update the operationalizing method models to be *more iconic*, resulting in *intuitive* models which are more *comprehensible* for the micro-business owner and *technically relevant* for the developers working on the software projects. This adaptation also serves as an *update* for modeling in

the context of the evolving computing paradigms of today. Figure IV.10 shows the traceability from micro-business (soft) goals (NFRs) to operationalizations, such as infrastructure, to FRs.

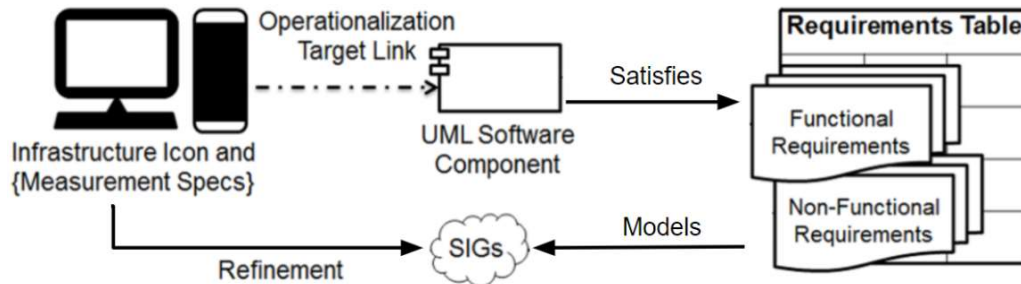


Figure IV.10 Adapting SIGs for Micro-business Software Systems

First, a μ BRP table is used to determine both the FRs and NFRs to be respectively satisfied and satisfied in the micro-business software system. Then, the NFRs (in SIGs) are refined into operationalizing methods. Operationalizing methods are measurable (or estimate-able) elements which satisfice refined softgoals. Originally, the operationalizing method is modeled with a cloud-in-bold element as specified in the NFR Framework. In Section 3.2.2, we propose the use of familiar icons with possible measurement specs in braces “{ }” for micro-businesses. The icons can be easily represented digitally with copyright-free icon libraries or even drawn by hand during informal requirements elicitation sessions.

FRs and the NFRs are linked naturally with μ BRPs. It is crucial to cover this aspect in order to provide a broader, more holistic view, and appropriate solution. When both the micro-business owner and the software developer have collaborated on refining the NFRs (which are modeled with SIGs), they have to come up with possible solutions that would satisfice the NFR softgoals. Depending on the priorities of the micro-business, such solutions could be made up of operationalizations, processes, and software systems. For instance, if the micro-business owner feels that having onsite storage is safer and more secure than cloud storage then the operationalizing method chosen would be an onsite server over a cloud server. The operationalizations of the NFR softgoals provide more concrete mechanisms in the target system where the FRs and NFRs meet (Chung et al, 2000).

Operationalizing methods target systems with operationalization target links. The systems could be software components modeled with UML components diagrams or business processes modeled in BPMN. The software components or business activities satisfy the FRs in the μ RP table.

3.2.2 Catalog of Operationalizing Methods

(Mairiza et al., 2010) propose a catalog for classifying the most popular and frequently occurring NFRs which could be performance, reliability, usability, security, and maintainability. By refining these popular NFRs into operationalizing methods, we propose a catalog of the most popular recurring operationalizing methods in the micro-business domain, as specified in Table IV.7, specifying operationalizing methods, examples, infrastructure icons, metrics for software developers, and management notes for micro-business owners. The purpose of this catalog is to improve the way operationalizing methods are modeled with SIGs, resulting in better collaboration for possible solutions between micro-business owners and software developers: from clouds-in-bold to *more familiar icons that the users could recognize*, apt for the evolutions of the computing paradigms of today.

Since the proposed operationalizing methods are derived from the most popular and frequently occurring NFRs, most of the common NFRs that appear in the μ RP tables would have commonly occurring operationalizing methods that relate to them. For instance, responsiveness is a commonly occurring NFR in the μ RPs and this NFR would usually be related to an operationalizing method that has a measurement spec with speed involved, e.g., a fast processing device with high processing power, internet connection with high bandwidth, etc. The succeeding paragraphs discuss these commonly recurring operationalizing methods that relate to commonly recurring NFRs for micro-businesses.

The first operationalizing method is the stationary processing device. It is a non-movable unit such as a desktop or pc and is represented with both a monitor and a desktop tower unit icon. Developers can consider the specs to ensure proper performance while micro-business owners can consider warranty and power consumption in their notes. In the micro-business restaurant, this could be a desktop pc functioning as a cashier system where the records of all the payments of the customers are stored.

The second operationalizing method is the mobile processing device. It is a movable unit such as a mobile phone or laptop and is represented with a smartphone icon. Like the

stationary processing device, developers can consider the specs to ensure proper performance while micro-business owners can consider warranty and power consumption in their notes. In the micro-business restaurant, this could be a mobile payment device which the waiters and waitresses carry around the dining tables to collect payments after meals.

The third operationalizing method is the display device. It is a device that displays information such as flatscreen television or a desktop monitor and is represented with a monitor icon. Developers can consider the resolution of the display device while micro-business owners can consider the lifetime and warranty of the display unit in their notes. In the micro-business restaurant, this could be the monitor that is attached to the desktop pc.

The fourth operationalizing method is the networking device. It is a device that enables the transfer of data between devices such as a router or a satellite dish and is represented with a broadcasting router icon. Developers can consider the data transfer rates while the micro-business owners can consider the geographical location in their notes. In the micro-business restaurant, this could be the router that is used at the locale to connect to the internet. The restaurant may need to connect to the internet because of cloud-based email or because of some other software application hosted in the cloud.

The fifth operationalizing method is the virtual security element. It is an element that protects the software from virtual threats like a virus or malware. An example of such an element is anti-virus software or a firewall and is represented with a virtual shield icon. Developers can consider scan speed and frequency while micro-business owners can consider the risks if a virtual threat happens in their notes. In the micro-business restaurant, this could be an updated Operating System that prevents malware attacks.

The sixth operationalizing method is the physical security device. It is a device that protects the system from physical threats like a thief. An example of this device would be a surveillance camera or a physical lock and is represented with a lock icon. Developers can consider the capture resolution of the video or the durability of the physical lock while the micro-business owners can consider the risks if a physical attack happens in their notes. In the restaurant micro-business, this could be a steel gate at the front door which is closed at night to prevent any intrusions from thieves.

The seventh operationalizing method is the virtual third party. It is a party which provides virtual services for the software to function such as cloud data storage and is represented with

a broadcasting building icon. Developers can consider transfer rates to and from the virtual third party while micro-business owners can consider the service level agreements and alternative back-ups in their notes. In the micro-business restaurant, this could be a cloud storage provider on the internet where the micro-business stores its marketing media such as pictures of the food and other promotional material.

The eighth operationalizing method is the physical third party. It is a party which provides tangibles for the system to function such as utilities and is represented with a building icon. Developers can consider how much of the utilities they can use and at what rate while micro-business owners can consider the type of contract and the track record of the third party in their notes. In the micro-business restaurant, this could be a meat supplier who the micro-business has to interact with almost daily to get his or her fresh supply of ingredients for the meals.












The ninth operationalizing method is the human resource. These are human resources that contribute to the software system and are represented with a human stick figure icon. The developers can consider the skill set of the human resources supporting the software system while the micro-business owners can consider the contracts they have with the human resources in their notes. In the micro-business restaurant, this could be the waiters and waitresses who are serving the customers at their tables when they order food.

The tenth operationalizing method is the logistics. This is a means which can transport tangible items between locations such as a transport truck and is represented with a transport truck icon. The developers can consider transport capacities and delivery frequency while micro-business owners can consider the appropriate transport means in their notes. In the micro-business restaurant, this could be an owned vehicle which picks up ingredients from a supplier that does not deliver.

There are still operationalizing methods to-be-determined **TBDs** and can even be determined on a case-to-case basis by the developers and the micro-business owners using SIGs for their representations. If the operationalizing method is still to be determined, we would even encourage the users to improvise and model the operationalizing methods as they see fit for their models. Both the developers and the micro-business users can always use the original cloud-in-bold icon to represent an operationalizing method

On any of these icons which represent operationalizing methods, labels are used (as recommended by (Mendling et al., 2010)), indicating exactly what they are (or planned “to-be”), accompanied by a metric (in braces “{“ ”}”) for which satisficing/satisfying could be measured or estimated. The way these labels are placed can be seen in Figure IV.10.

Table IV.7 Catalog of Operationalizing Methods

	Operationalizing Method	Examples	Developer Specs	Micro-business Notes	Icon
1	Stationary Processing Device definition: this is a non-movable data processing unit (may have display device)	desktop pc, stationary server	Overall CPU performance CPU+RAM+GPU+DD Input-Output-Network-OS Windows-OSX-Linux	Warranty/Return policy? How long do you plan to use this device? Overpowered/just right?	
2	Mobile Processing Device definition: this is a movable data processing unit	mobile phone, tablet, laptop	Overall MPU performance CPU+RAM+GPU+DD Input-Output-Network iOS-Android	Warranty/Return policy? How long do you plan to use this device? Overpowered/just right?	
3	Display Device definition: this device displays information	flatscreen, monitor,	Resolution SD/HD/4K/3D Power Consumption Lifespan, pixel burnout rate	Warranty/Return policy? How long do you plan to use this device?	
4	Networking Device definition: this device enables the transfer of data between devices	router, dish, antenna,	Response Time, Ping, Data Tranfer Rates, Thoroughput, Streaming (multimedia)	How far are devices from each other? Can the devices communicate without interferences?	
5	Virtual Security Element definition: this element protects the software system from virtual threats	antivirus, firewall, SSL, WEP, WPA, PIN	launch speed, scan speed, memory utilization, bit encryption	How will the software system be protected virtually? What are you protecting the system from?	
6	Physical Security Device definition: this device protects the software system from physical threats	surveillance, CCTV, lock and chain	capture resolution (id-ability), capture duration records, material types	How will the software system be protected tangibly? What are you protecting the system from?	
7	Virtual Third Party definition: this is a third party which provides virtual services for the software to function	internet, cloud application, dedicated server	Data Tranfer Rates, Bandwidth consistency, Jitter, Latency, Packet loss	How do non-tangible elements affect the software system? What is the SLA? Is there a back-up?	
8	Physical Third Party definition: this is a third party which provides tangibles for the system to function	utilities, goods supplier, maintenance, wired phone	utility per dollar, maintenance rates	How do tangible elements affect the software system? What is the contract? Track record of third party?	
9	Human Resource definition: these are the human resources that contribute to the software system	cashier, admin, user, web visitor	Resumé, Skill Set-Match, Experience Level-Match, Loyalty, Ethics, Legal considerations	Capability/Motivation? Incentive scheme? Labor laws/HR policies?	
10	Logistics definition: these are means by which tangible items are transported between locations	transport truck, delivery truck	mileage per gallon/liter, transport capacities	How will tangible items be transported from one place to another?	
	TBD's definition: these are operationalizing methods which remain to be determined	TBD	TBD	TBD	

4. A REAL-WORLD EXAMPLE OF AN μ BRP IN PRACTICE

Since μ BRPs are based on real-world practice, we would like to further describe our proposal using a real-world example. Let us put this micro-business example into context then: At the end of 2020, there were 1,107,145 micro-businesses with 1-9 employees (Spain SME Statistics, 2021). It contracted by 2.3% from the previous year. The closure of almost 30,000 micro-businesses during the global pandemic of 2020 shows how difficult it is for micro-businesses to stay afloat these days. Micro-businesses need to consider new constraints such as social distancing, mask wearing, washing of hands, curfews, and lockdowns in their locales. In addition, if micro-business owners become ill with fever, colds, or any symptoms of COVID-19, they will have to quarantine and even close shop if required.

Given these constraints of physical contact, the ordinary brick-and-mortar retail store is now more than ever forced to go online. These types of micro-businesses are left with almost no choice but to embrace evolving computing paradigms. They must familiarize themselves with infrastructure and devices that enable them to do business without any physical contact, preventing the spread of COVID-19 and allowing them to stay in business during these trying economic times.

Our real-world micro-business example is a retail shop owner in Spain who wants to stay in business today. His plan to stay in business is to convert his physical retail shop into an online retail shop, save on locale costs, and then source and ship orders faster than his competition. He wants his software system to have accurate, real-time online sales orders and inventory information so that he could make better and faster logistics decisions. The online retail shop of the micro-business must be as responsive as possible, both from the customer and administrator perspectives, to improve the shipping of products to customers. The micro-business owner is not interested in owning the software and maintaining it and would rather focus on the business instead of worrying about software matters. Instead, the developer will be billing the micro-business owner a monthly subscription for the use of the software system, i.e., online retail shop system.

4.1 Requirements Elicitation

The *first step* for the micro-business owner and the software developer is *to meet and clarify the requirements during elicitation*. Note that in Requirements Engineering, there is a clear distinction between the role of requirements engineer and software developer. However, due to resource constraints in micro-businesses, this role is usually done by one person, the software developer.

The software developer brings a μ BRP table to the requirements elicitation meeting with the micro-business owner as shown in Table IV.8. This table is meant to *speed up requirements elicitation* through *better communication* with the micro-business owner *without compromising technical details*. The table is *straightforward* and *could even be answered by the micro-business owner himself* even without the aid of a software developer. However, it is *recommended* that the software developer elicit the requirements together with the micro-business owner.

The requirements elicitation table **before** the requirements elicitation meeting is shown in Table IV.8. This table would be the **same** for several online retail shops. Since a μ BRP is a pattern of recurring software requirements in micro-businesses, the recurring software requirements of online retail shops are included in the table.

Table IV.8 Online Retail Shop µbRP Requirements Elicitation Table **Before** Elicitation Meeting

online retail shop µbRP		
description: µb sells items online, customer pays online, and then µb ships item to the customer.		
keywords: online shopping, online payment, online activity monitoring		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb customers find the product they want at the online shop?	search engine / filters / product catalog / offers / other, select as many that apply	
(b) How are the products of the µb presented on the online shop?	image / text / video / review (see (c)) select as many that apply	
(c) Can an µb customer leave product reviews on the online shop?	yes, optional (see account in (f)) / no	
(d) How does the online shop accept payments? (credit card pattern can be found at (Macasaet et al., 2011))	credit / debit / PayPal / Skrill / other select as many that apply	
(e) Will the online shop offer a tracking service for product deliveries?	yes (see (l)) / no	
(f) How will the online shop collect visitor/shopper data?	server-based / client-based / customer account-based / other / none, select as many that apply	
(g) Are admin privileges needed by µb-side users?	yes / no	
(h) Does the online shop have a physical retail store location?	yes (see (l)) / no	
(i) What other µb information is available on the online shop?	company info / physical store locations / FAQs / other, select as many that apply	
(j) Is the online shop linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	
(k) Is the online shop linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	
(l) Is the online shop linked to a logistics management system?	yes / no	
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
system speed (responsiveness)		
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
quickness (reponsiveness) of the software system (online shop)		
user-friendliness of the software system, customer side (online shop)		
security of the software system (transactions made on online shop)		
availability of the product in the inventory (inventory stock levels)		
delivery time of the product (fast product delivery)		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		

The Online Retail Shop µbRP table shows its three main sections. The description section, the topmost section, is the section which shows the following details: the name which is the “Online Retail Shop µbRP,” a brief description of the µbRP which is “µb sells items online, customer pays online, and then µb ships item to the customer,” and then finally the keywords of the µbRP which are online shopping, online payment, and online activity monitoring.

In the functional requirements section, the middle section, of the Online Retail Shop µbRP table, we find the business-like, question-answer format. The first question (a) “How will the µb customers find the product they want at the online shop?” until the last question (l) “Is the online shop linked to a logistics management system?” are all expressed as *questions in a straightforward, easy-to-understand language for micro-business owners*.

In the second column of the functional requirements section of the Online Retail Shop µbRP table, the possible options or answers to the business-like, question-answers, are found. Shown are the possible options or answers to the question (a) “How will the µb customers find the product they want at the online shop?” in the column right after it which shows the possible options: search engine, filters, product catalog, offers, and others, being able to select as many that apply to the question. The question (d) “How does the online shop accept payments?” has the possible options or **modes of payment** in the second column after the question which are payment by credit card, PayPal, etc. From the first and second columns of the functional requirements section, the processes of the micro-business are provided.

In the third column of the middle section of the Online Retail Shop µbRP table, the responses of the micro-business owner are shown under the column “choices.” For the question (a) “How will the µb customers find the product they want at the online shop?”, the choices are in the third column of the table which are: search engine, filters, and catalog.

The requirements elicitation table **after** the requirements elicitation meeting is shown in Table IV.9. This table would be **different** for several online retail shops, depending on the choices made by each micro-business owner. The non-functional requirements of the Online Retail Shop micro-business are discussed in further detail in Section 4.3.

Table IV.9 Online Retail Shop µbRP Requirements Elicitation Table **After** Elicitation Meeting

online retail shop µbRP		
description: µb sells items online, customer pays online, and then µb ships item to the customer.		
keywords: online shopping, online payment, online activity monitoring		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb customers find the product they want at the online shop?	search engine / filters / product catalog / offers / other, select as many that apply	search engine, filters, catalog
(b) How are the products of the µb presented on the online shop?	image / text / video / review (see (c)) select as many that apply	image, text, review
(c) Can an µb customer leave product reviews on the online shop?	yes, optional (see account in (f)) / no	yes
(d) How does the online shop accept payments? (credit card pattern can be found at (Macasaet et al., 2011))	credit / debit / PayPal / Skrill / other select as many that apply	credit, debit, PayPal, Skrill
(e) Will the online shop offer a tracking service for product deliveries?	yes (see (l)) / no	yes
(f) How will the online shop collect visitor/shopper data?	server-based / client-based / customer account-based / other / none, select as many that apply	server and customer-based
(g) Are admin privileges needed by µb-side users?	yes / no	yes
(h) Does the online shop have a physical retail store location?	yes (see (i)) / no	no
(i) What other µb information is available on the online shop?	company info / physical store locations / FAQs / other, select as many that apply	company info, FAQ
(j) Is the online shop linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	yes
(k) Is the online shop linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	yes
(l) Is the online shop linked to a logistics management system?	yes / no	yes
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
(m) system speed (responsiveness)		3rd
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		5th
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
(m) quickness (reponsiveness) of the software system (online shop)		2nd
user-friendliness of the software system, customer side (online shop)		
security of the software system (transactions made on online shop)		4th
availability of the product in the inventory (inventory stock levels)		
(n) delivery time of the product (fast product delivery)		1st
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		1st
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		2nd

→ The letters in parantheses in this Table are linked to encircled letters in the Figures

4.2 Modeling

BPMN is used for modeling the business processes in the functional requirements section. Providing this point-of-view helps both the micro-business owners and the software developers understand the functional requirements. The BPMN model in Figure IV.11 shows the functional requirements, its modes, and options.

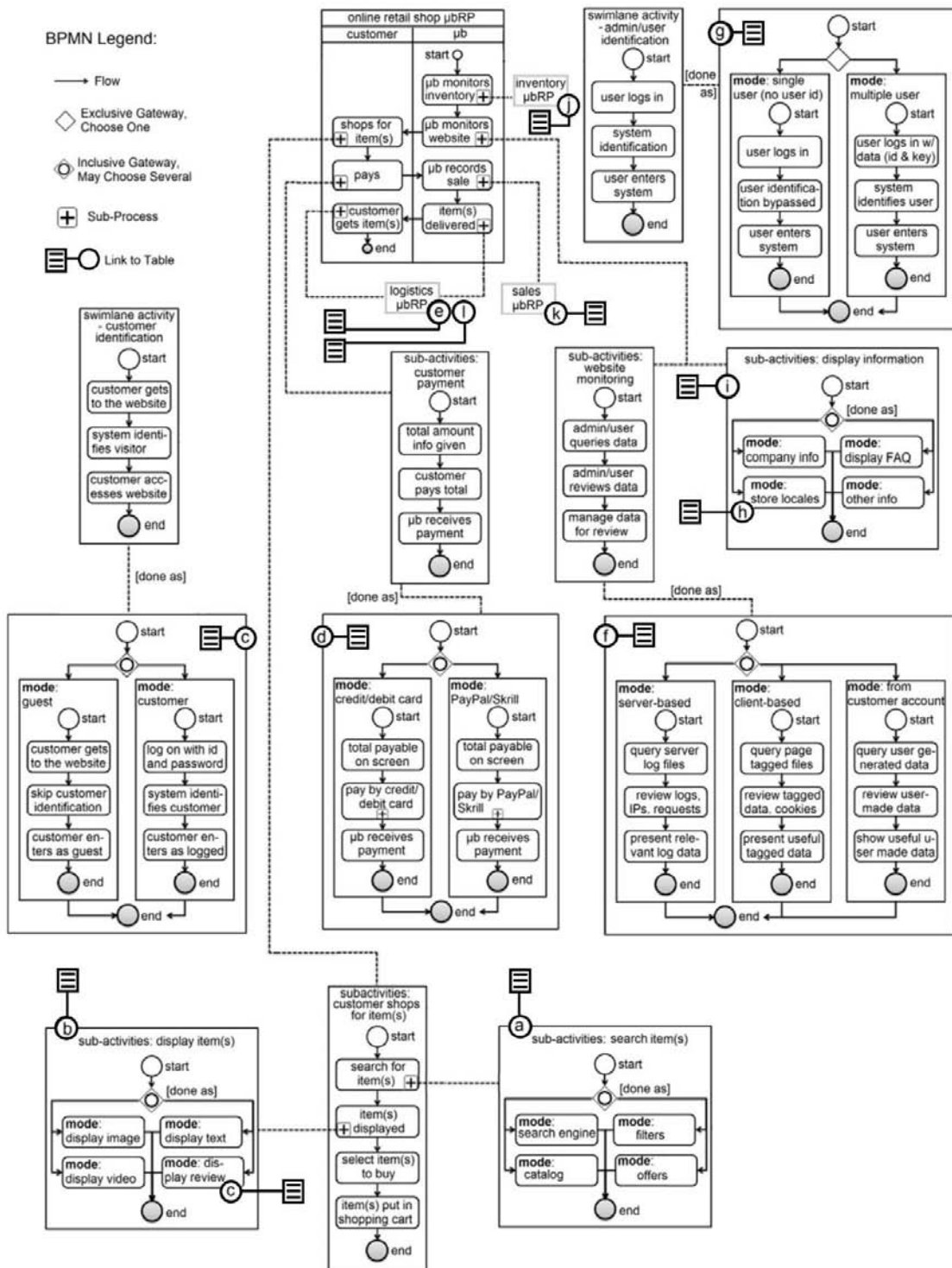


Figure IV.11 Model of the modes or “options” of the Online Retail Shop μRP

The purposeful placement of custom labels in business process models (in this case, the BPMN models) is recommended to make the models more comprehensible or maybe even more technically relevant (Mendling et al., 2010). Using custom labels, a basic overview of *how μ RP tables link to models* is provided in Figure IV.12 with only requirements (a) and (m). The BPMN model for the complete requirements from (a) to (n) are provided in Figure IV.11.

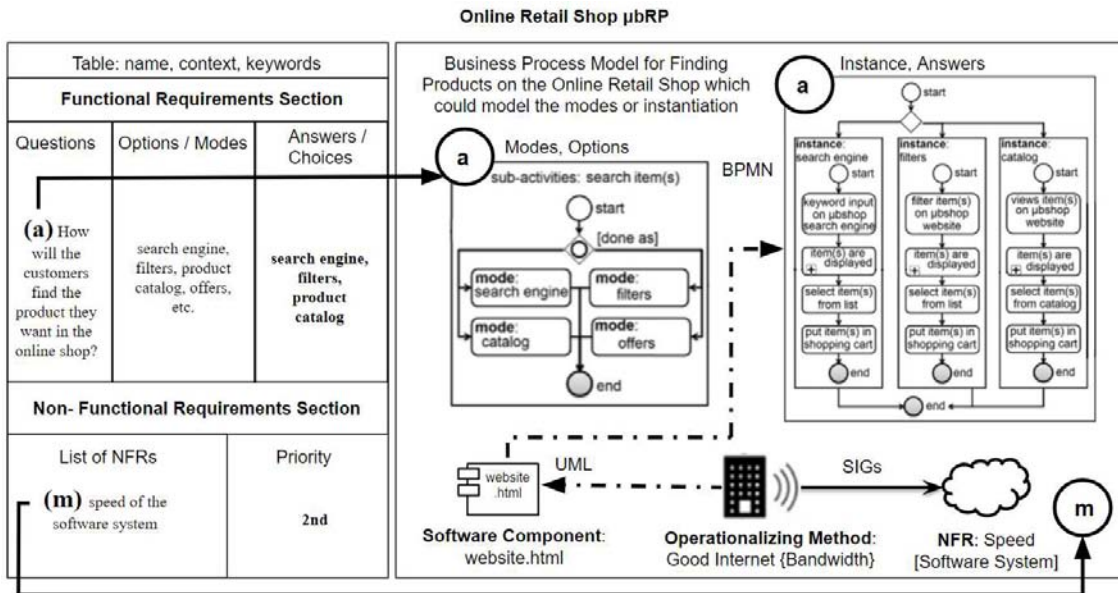


Figure IV.12 Linking μ BRP Tables and Models

Figure IV.12 shows several custom labels which help the viewers understand the links between tables and figures. The first custom label is the letter in parentheses which is found in the μ BRP tables. Table IV.8 and Table IV.9 show these letters in parentheses before the questions in the first column in the functional requirements section. For example, the question: “How will the customers find the product they want in the online shop?” is preceded by the letter “a” in parentheses. This means that there is an area in a figure such as Figure IV.11 where this functional requirement is modeled in BPMN. The area in the figure has an encircled letter-in-bold that corresponds to this question. In Figure IV.11, the viewer must look for an encircled letter “a” in bold and see how this functional requirement is modeled in BPMN.

In Table IV.8 and Table IV.9, there are modes in the second column of the functional requirements section. For question (a), these modes are to find products via search engine, filters, product catalog, offers, and others. In Figure IV.11, the modes of functional requirement

(a) are modeled in BPMN in the area indicated with an encircled letter “a” in-bold. There is also a [done as] label in the BPMN model to indicate the modes/options. In the succeeding chapter where we evaluate the μ BRPs, our observations show that the custom labels, e.g. [done as], as shown in these figures provide indications and important notes for developers who would like to reuse μ BRPs and also their associated software components.

In Table IV.8 and Table IV.9, the answers, choices, and priorities made by the micro-business owner are shown in the last columns of the functional requirements section and the non-functional requirements section. For question (a), the answers/choices are search engine, filters, and catalog. In Figure IV.13, *only the choices/answers* made by the micro-business owner for functional requirement (a) are modeled in BPMN in the area indicated with an encircled letter “a” in-bold. These choices correspond to instances which are discussed in further detail in Section 4.3.

4.3 Instantiation

The **choices made by the micro-business owner** are used in the **instantiations** of the pattern. The micro-business choices are reflected in the last column of the functional requirements section in Table IV.9. The **instances** of a pattern are the results of the pattern in practice in the real-world when the micro-business owner and the software developer have agreed on modes or options in which the pattern will be done (as), i.e., “how will the option be done?” The instance of a pattern could be understood by taking the example of the modes of payment and then envision what happens when a customer actually pays. In question (d) “How does the online shop accept payments?”, the micro-business has chosen that the customer can either pay with a credit card or pay with an e-wallet like PayPal or Skrill. The BPMN diagram shows that the micro-business owner *has chosen* that the customers *can only pay with these two options*: credit card or e-wallet. Which means that in the real-world **instance**, if the customer would have wanted to pay using a bank transfer, it would not have been possible because the micro-business owner did *not* choose this as one of the possibilities to accept payment.

In the bottom section of the Online Retail Shop μ BRP table in Table IV.9, the NFRs are listed. These NFRs are the most commonly occurring ones for Online Retail Shops and have been grouped in terms of priorities to the micro-business owner, the micro-business customer, and the software developer. These priorities are ranked from the most important to the least

important; where 1st is the label used in the table for the most important priority, and 2nd, 3rd, etc... is used up to the least important.

The most important NFR for the online retail shop would be “(m) the timely delivery of the product to the customer.” In Figure IV.13 in the area with an encircled letter in-bold “m”, this NFR is modeled with SIGs and refined as “(n) speed of the software system” since the timely delivery of the product is directly dependent on the speed of the software system. The refined NFR is also considered the 2nd most important NFR for the micro-business. The NFRs are ranked instead of given a definite answer since these requirements are “satisficed” instead of satisfied. This provides *insight on what quality attributes are most important and provides guidance on the choices to make when trade-offs have to be made.*

In Figure IV.13, the non-functional requirement “speed of the software system” is located with an encircled letter “m” in-bold which means that it is a model that corresponds to letter “m” in the non-functional requirements section in Table IV.9. The non-functional requirements are modeled **after** prioritizations have been made by the micro-business owner. This is a practical approach because modeling NFRs before prioritizations are made is time-consuming and may not be necessary because some NFRs are *not that* important based on actual prioritizations of the micro-business owner. The operationalizing method “Good Internet” is a refinement for the NFR “speed of the software system” and is modeled with the operationalizing method icons proposed in Section 3.2.2. Good internet supports how the website will function which is modeled in UML. Finally, the website supports the instance of the business process of how customers look for products which is modeled in BPMN. This BPMN model is located with an encircled letter “a” in-bold. The flow from an NFR to a BPMN model is explained in detail in Section 3.1.4.

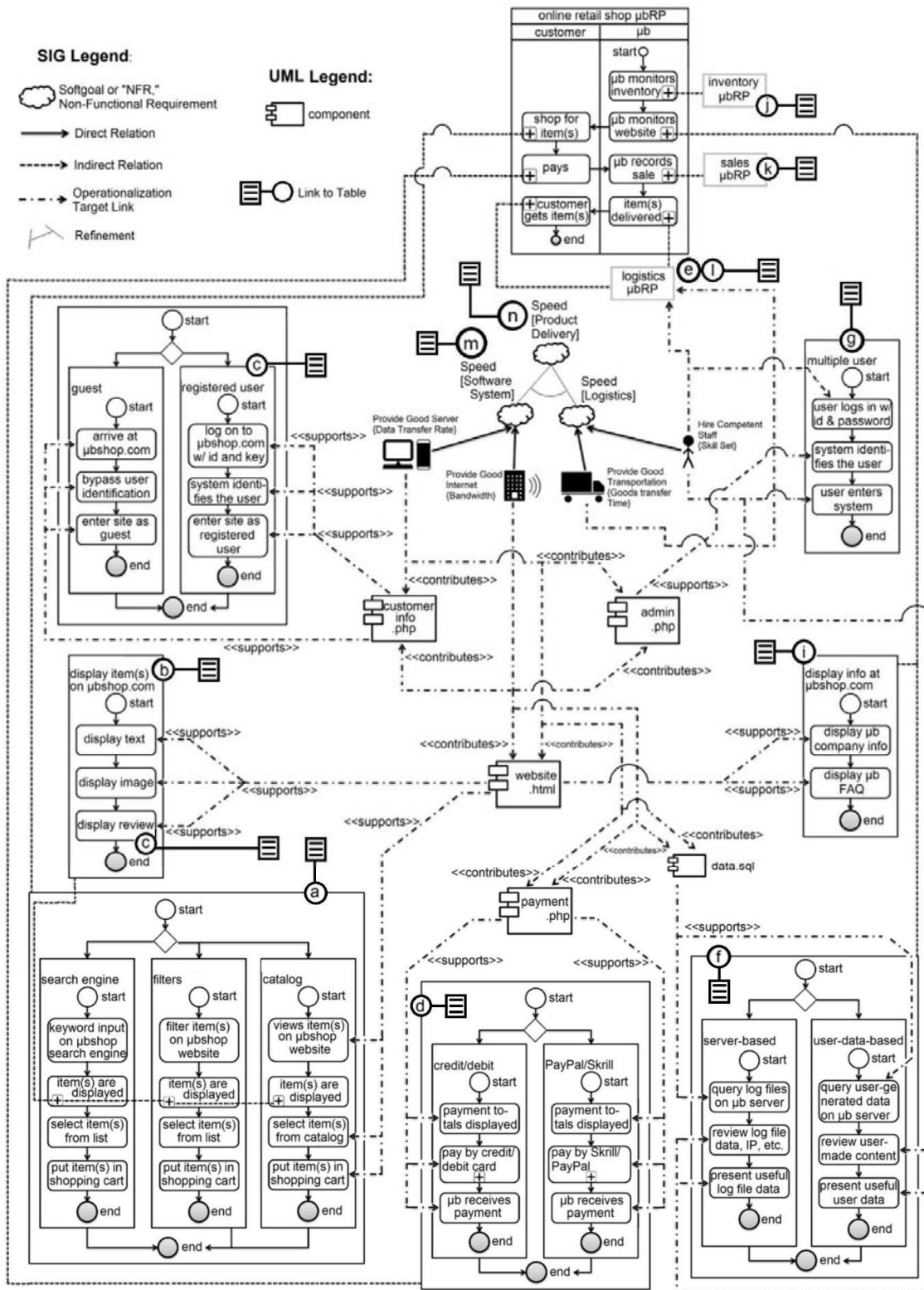


Figure IV.13 Model of an Online Store μbRP instantiation with “choices” and “priorities”

As discussed in the previous chapter and as detailed in Appendix B.2, NFRs are modeled with a cloud and then lines connect the NFRs to the refined NFRs. The catalogue adapting SIGs for the micro-business domain presented in Section 3.2.2 is used to model the operationalization methods. Figure IV.13 shows three NFRs namely: the speed of product delivery which is identified in the figure with an encircled letter in-bold “n”, refined into two NFRs which are the speed of the software system which is identified in the figure with an encircled letter in-bold “m”, and the speed of logistics. The letters “m” and “n” are encircled **in-bold** because **they correspond directly to prioritized NFRs in Table IV.9**. Both the speed of the software system and the speed of logistics contribute to the speed of product delivery for the customer. This SIGs model shows that importance has to be given to these two NFRs to ensure that the most important NFR is satisfied.

From the refined NFRs, operationalization methods point to the NFRs that it satisfies and an operationalization target link is used to point to a system or software which would depend on such operationalization. From the catalog presented in Section 3.2.2, the operationalizing methods stationary processing device (server) and virtual third party (internet provider) are used and point to the NFR speed of the software system. This means that in order to satisfy the NFR speed of the software system, there must be a good server and internet provider in place. The operationalizing method models of logistics (transporter) and human resource (staff) point to the NFR speed of logistics. This means that in order to satisfy the NFR speed of logistics, there must be good transporters and competent staff to carry out these tasks.

From the operationalizations, operationalization target links point to software systems or business processes. In Figure IV.13, the good server operationalization points to the customer info component and the website component. This means that in order for the customer info component and the website component to function properly, it needs to be supported with a good server. The customer info component supports the business process of identifying customers when logging on to the Online Retail Shop system while the website component supports the business process of how customers search for products on the Online Retail Shop. From the NFRs in Table IV.9 to the SIGs models in Figure IV.13 to the BPMN models which correspond to the business processes of the FRs in Table IV.9, the μ BRPs come full circle. Should the software developer and micro-business owner prefer more models for the 4th, 5th, etc. NFRs, they are able to continue modeling, depending on a case-to-case basis.

4.4 Software Design

For keeping the models manageable, not all associated components are placed in the models which include BPMN and SIGs as shown in Figure IV.13. **After** creating the SIGs models, the software developer creates UML models to include within the BPMN and SIGs as shown in Figure IV.13. Then, a UML component deployment diagram with more details is created that would further guide in the implementation of the system as shown in Figure IV.14. As will be explained in the next section, Section 5, managing μ BRPs, based on the design in the UML, the software developers will either create these software components or reuse software components if they already exist in a repository.

Figure IV.14 shows the website component which was also shown in Figure IV.13. More detail about the website component is provided such as the required interfaces for it to function. Through the UML diagram, the software developer is trying to explain the relationships among the software components which later on provide guidance on any opportunities to (re-)use such components. For the website component, such relationships are detailed in the following paragraphs.

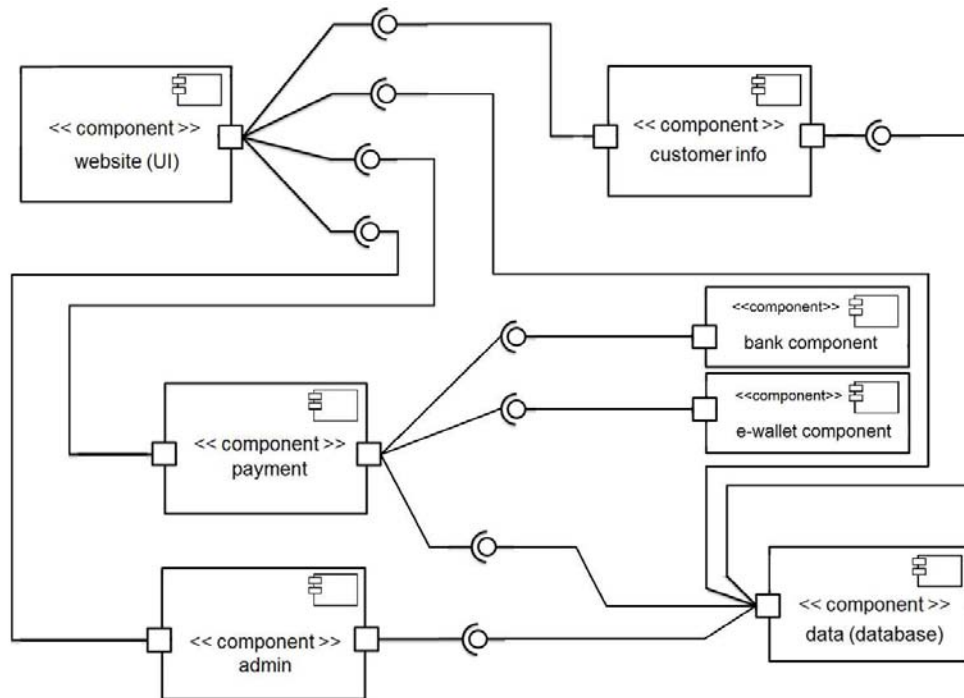


Figure IV.14 A UML model from the instantiated μ BRP (in Figure IV.13)

For the website user interface component to function, it will need input from the customer information component, the database component, the payment component, and the admin component. In the online retail micro-business example, the website user interface could be an entire front-end written in JavaScript which would need input from the other components.

For the customer information component to function, it will need input from the database. In the online retail micro-business example, this customer information component could be a module which is focused on collecting and providing customer information between the user interface and the database.

For the payment component to function, it will need input from the bank component, the e-wallet component, and the database. In the online retail micro-business example, this payment component could be a module which is focused on facilitating the payment of users from financial institutions such as banks or e-wallets so that users could order items from the online retail shop and have them delivered to their homes.

For the admin component to function, it will need information from the database. In the online retail micro-business example, the admin component could be a module which manages the users who can log into the system and check balances, orders, and status of shipments.

The bank component provides output to the payment component. In the online retail micro-business example, the bank component could be a module that interfaces with various banks so that users could use any credit or debit card they have associated with a bank to make their payments on the online retail shop.

The e-wallet component provides output to the payment component. In the online retail micro-business example, the e-wallet component could be a module from an e-wallet company such as PayPal which would allow the user to make a payment through a portal on the website user interface.

The database provides output to the customer information component, the website user interface component, the payment component, and the admin component. In the online retail micro-business example, the database could be a relational database using Structured Query Language SQL.

4.5 Complementary Notes

Throughout the requirements process, notes are taken by the micro-business owner and the software developer/analyst. They are compiled as complementary notes and would be included as follows:

- There will be discussions regarding the speed and reliability of the hosting servers and internet providers.
- Investments in reliable computers are to be taken into consideration.
- From the point of view of the software developer, the top priority is to profit from the project on a long-term basis.
- The software which will be developed is not going to be owned by the micro-business owner. Instead, the micro-business owner will be billed for the developed software on a subscription basis.
- If the functioning of the software is disrupted due to issues related to the global pandemic, then the software developer will be obliged to immediately resolve the issue as soon as humanly possible, without endangering his health and the health of others.
- The software developer will not be held responsible if the shipping of goods to customers is affected by issues related to the global pandemic. The micro-business owner will be responsible to resolve these issues within his organization and with his third parties. Since the speed of the delivery is a top priority, the speed of shipping takes priority over other NFRs.
- If a global lockdown is implemented which means that either the micro-business owner or the software developer is unable to perform his duties and obligations, the subscription fees will be waived during the period of the global lockdown and any damages or fines related to the malfunctioning of the software will be exempt from liabilities on the part of the developer. By the inability to perform duties and obligations, we mean for instance that all shipping routes are made inactive or all computer servers are shut down, e.g., circumstances which totally prevent either party from functioning even at bare minimum of his operations.

5. MANAGING μ BRPs

This section explains how μ BRPs are managed. A detailed explanation on how μ BRPs are created, (re-)used, and then applied into practice is provided. In addition, explanations on how the associated software components are used alongside the μ BRPs are provided within the context of Component Based Software Engineering “CBSE.”

5.1 Creating μ BRPs

Before being able to (re-)use a μ BRP in practice, it must first be created. In Figure IV.15, a step-by-step process for creating and then using μ BRPs is shown.

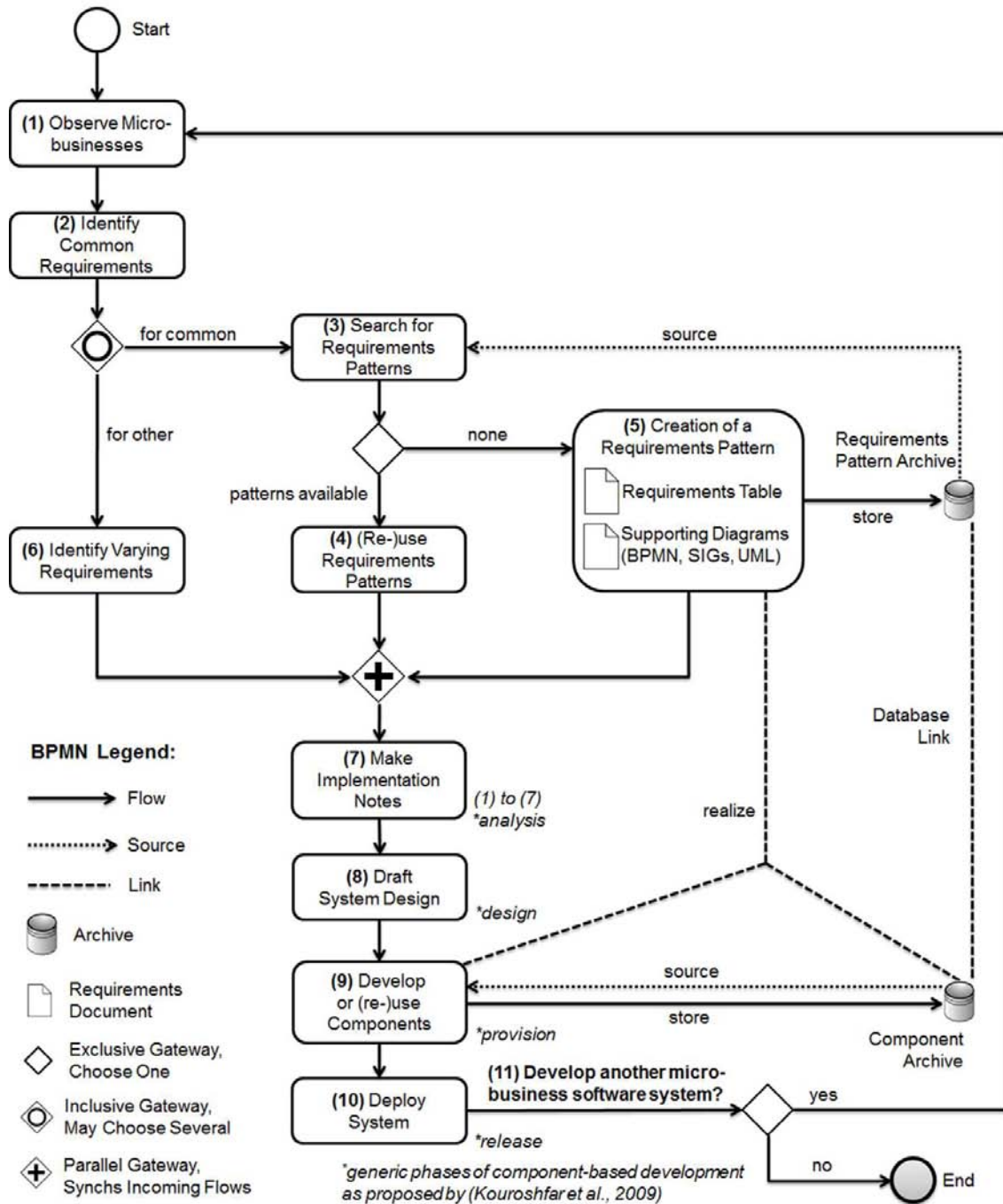


Figure IV.15 A BPMN model of the process of managing µBRPs

First, we start off by creating an μ BRP. (1) Observing micro-businesses is the first step of the process. The goals and requirements of micro-businesses are listed down. Upon listing down the requirements, (2) common, recurring requirements are identified. If there is an existing repository of μ BRPs, developers (3) search for μ BRPs using keywords in the repository and if there are relevant results, i.e., common, recurring requirements, (4) then μ BRPs are (re-)used. However, if there is no existing repository or if there are no relevant μ BRPs, then a (5) μ BRP is created.

As defined, a μ BRP is a pattern of recurring requirements models in micro-business software projects and the minimum number of common requirements to constitute such a pattern is two. This is because if the number is less than two then there would be nothing in common and there would be no pattern in the first place.

Figure IV.16 shows an example of how common, recurring requirements are grouped and then diagrammed. This is the basis of all the (recurring) requirements in the μ BRPs. The figure shows that Hypothetical μ A and Hypothetical μ B have two common requirements: record a cash sale and display total cash sales. From these two common requirements occurring in both Hypothetical μ A and Hypothetical μ B, they are combined and described in one (recurring) requirement. The pattern is called "Record-Display Cash Sale." It is accompanied with a text description, i.e., for requirements: "record cash sale" and "display total cash sales." From the text description, a BPMN diagram can be created to further explain the (recurring) requirement as shown in Figure IV.16. At this point, SIGs and UML models are still not created because they become more relevant after filling up requirements tables.

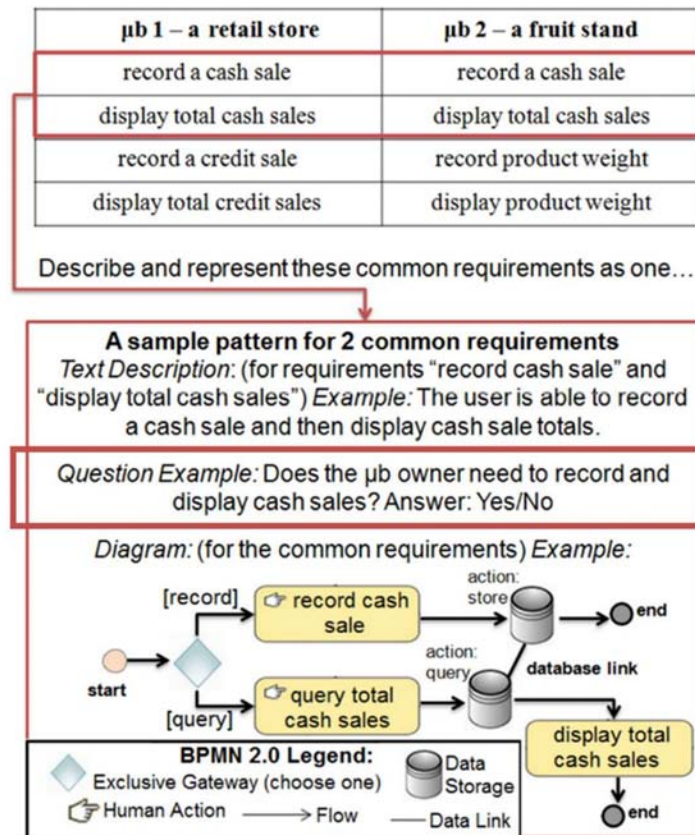


Figure IV.16 Grouping and Modeling Common Requirements and Turning Requirements into a Question Format that can be Answered with Yes or No, and corresponding BPMN diagram

Creators of the µbRPs have to ensure that the µbRP names are as unique as possible and that they are not repeating a µbRP that has already been created. During creation, they can double-check this in the repositories, assuming a repository already exists. Software development companies using wiki tools such as Confluence (Atlassian, 2021) follow basic guidelines for having unique names and titles for their wiki pages so that users can find them more efficiently later on. Such discipline can be applied to µbRP creation and archiving as well. Software developers are free to use whatever naming and archiving methodology that best suits their team practices.

5.2 Using μ RPs

The use of the μ RPs would either come from step (4) μ RPs are (re-)used, step (5) μ RPs are created, or both steps: (re-)used and created μ RPs. We will proceed with the steps and see how the μ RPs are used in practice.

After dealing with the common, recurring requirements in steps (4) and (5), step (6) varying requirements are identified and then (7) complementary notes are made. Aside from the BPMN models which correspond to the FRs in the requirements tables, the SIGs and UML models are prepared for the (8) design and (9) reuse or development of software components and their architecture. The SIGs models correspond to the prioritized NFRs while the UML models correspond to the associated software components to the μ RPs.

Finally, the (10) micro-business software system is deployed and eventually, (11) future systems would benefit from the growing repositories of μ RPs and associated software components. As seen in Figure IV.15, the process of creating and using μ RPs is iterative, meaning that the more times it is done, the more chances that there will be a growing and evolving library of μ RPs and associated software components for the software developers. In the case of μ RPs where micro-business owners make similar choices, even more opportunities for reuse become apparent. For example, if there is an existing repository of components that satisfy FRs and satisfy NFRs, then these components could be found with keywords (FRs, NFRs, operationalizations, etc.) or notes and then can be re-used. However, if the components are not those that directly comply with the requirements, then new components may have to be developed.

Managing μ RPs are in line with Component Based Software Engineering "CBSE" practices. (Kouroshfar et al., 2009) propose generic phases of component-based development based on seven popular component-based development methodologies. These generic phases are analysis, design, provision, and release. Activities (1) to (7) correspond to **analysis**, (8) and (9) correspond to **design** and **provision** of components respectively, and (10) corresponds to **release**. These generic phases are marked with an asterisk "*" in Figure IV.15.

In the context of component-based software development work by (Kouroshfar et al., 2009), we consider the association of software components to the operationalizations from the NFRs

in the μ BRPs as part of the **analysis** phase – where the requirements of the system are elicited and a preliminary project plan and schedule is outlined. In Figure IV.15, the analysis phase is reflected in steps (1) through (7).

In the **design** phase, the components of the system are identified and based on the dependencies and interactions among these components, specifications are made for each component. This is when the system is designed and is step (8) in Figure IV.15. This is explained in more detail in Section 3.1.3 about component modeling using UML.

In the **provision** phase, the required components (from the analysis and design phases) are (re-)used from the component repository or written from scratch. This is step (9) in Figure IV.15.

In the **release** phase, the components are assembled for deployment which is shown as step (10) in Figure IV.15. The proposition of using CBSE is based on the functionality of the software components (analysis phase) and their fit for the implementation (design phase). There are other non-functional attributes and characteristics of software components such as (re-)usability, efficiency, and maintainability and we acknowledge that this is **not** yet specified in this proposal and have made it **part of our future work**.

6. TOOL SUPPORT FOR μ BRP MODELING

The μ BRP models involve several languages and notations, i.e., BPMN, SIGs, and UML, and a modeling tool would come in handy to support the software developers creating and maintaining these μ BRP diagrams. We have developed Requirements Engineering Tools “RE-Tools” (RE-Tools, 2013) which is capable of modeling BPMN, SIGs, UML, and additional labels, e.g., encircled letters in bold, [done as], all at the same time, all in one model.

RE-Tools has already surpassed 1,700 downloads internationally. We have done tool demonstrations in conferences (Supakkul & Chung, 2012; Supakkul et al., 2013), conducted several training sessions in industry (PSRI, 2018; Sabre, 2018; Virus, 2018; Everywhere, 2018; Desarrollo TIC, 2018), and have encountered publications by other researchers using the tool (Veerappa & Harrison, 2013). To improve RE-Tools, we constantly collect user feedback and incorporate improvements incrementally. Figure IV.17 shows a greyscale screenshot of RE-Tools, which depicts the simultaneous visualization of multiple notations, i.e., BPMN, SIGs,

and UML. This tool is available for download, is open-source, and can be tested by any software developer who needs tool support when using μ BRPs. The download link can be found in the μ BRP User Guide in the Appendix C.2.

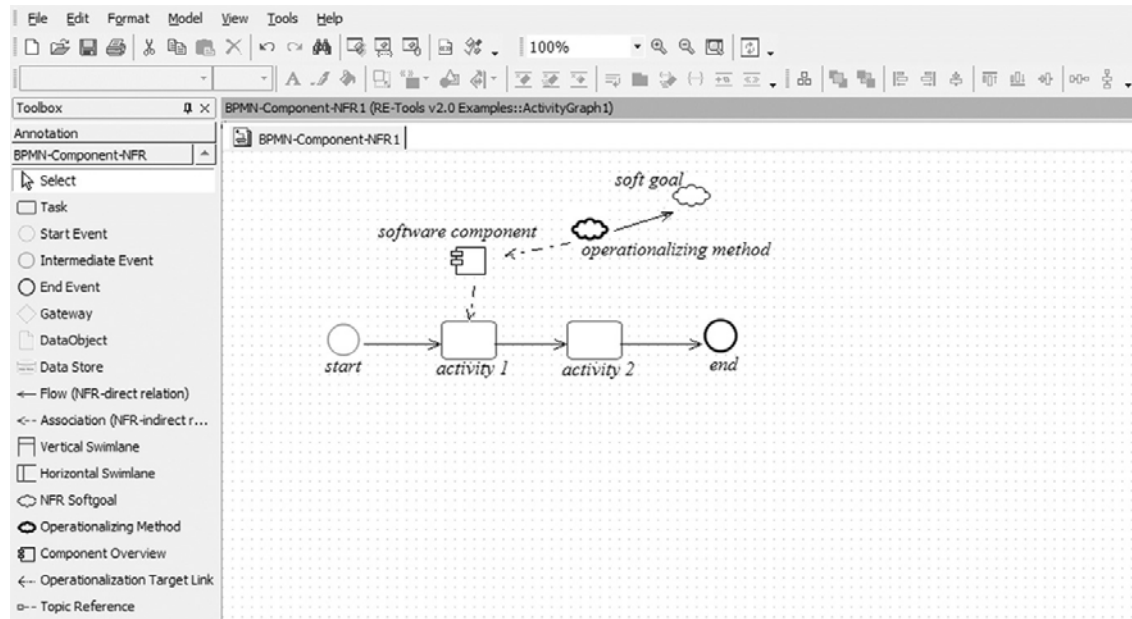


Figure IV.17 A screenshot of RE-Tools, showing multiple notations, BPMN, SIGs, and UML at the same time

7. CONCLUSIONS

In this chapter, we have provided our characterization and conceptualization of micro-businesses in terms of goals, requirements, software components, etc. In particular, we have presented a conceptual model for μ BRPs, explaining each and every concept and the relationships among them. We have proposed μ BRPs, which are recurring software requirement models in micro-businesses, with a description including tables, notes, and models in BPMN, UML, and SIGs. We have presented an adaptation and update of SIGs to make them more suitable in the micro-business domain. To bring the proposal of μ BRPs into practice, we have presented the μ BRP in a real-world example in order to validate the proposal. We have presented, step-by-step, the process of creating and using μ BRPs in practice. Finally, we have presented the diagramming tool, RE-tools, which allows the

simultaneous modeling of BPMN, UML, and SIGs. In the next chapter, we are going to discuss how we have evaluated μ BRPs in practice using Action Research.

CHAPTER REFERENCES

- Aranda, J., Easterbrook, S. & Wilson, G. (2007). Requirements in the wild: How small companies do it. Requirements Engineering Conference, 2007. RE '07. 15th IEEE International (p./pp. 39-48), ISBN: 978-0-7695-2935-6
- Atlassian. (2021). Atlassian. Last accessed on January 30, 2021 on www.atlassian.com
- Azar, J., Smith, R.K., & Cordes, D. (2007). Value-oriented requirements prioritization in a small development organization, IEEE software, vol. 24, no. 1, (pp. 32–37)
- Bürsner, S. & Merten, T. (2010). RESC 2010: 1st Workshop on Requirements Engineering in Small Companies, in workshop proceedings of Requirements Engineering for Software Quality REFSQ 2010, ICB-Research Report no. 40, October 2010, p. 128-130. url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Cardoso, E., Almeida, J., Guizzardi, R., & Guizzardi, G. (2011). A Method for Eliciting Goals for Business Process Models Based on Non-Functional Requirements Catalogues. *International Journal of Information System Modeling and Design*, 2 (2), (pp. 1-18) doi: 10.4018/jismd.2011040101
- Chung, L., Nixon, B., Yu, E., & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering*. Boston, Dordrecht, London. Kluwer Academic Publishers
- Desarrollo TIC. (2018). Desarrollo TIC. Last accessed on December 15, 2018, at <http://www.desarrollotic.com>
- Everyware. (2018). Everyware Technologies. Last accessed on December 15, 2018, at <http://www.everywaretech.es>
- European Commission. (2013). User Guide to the SME Definition. Last accessed on April 3, 2021 at https://ec.europa.eu/regional_policy/sources/conferences/state-aid/sme/smedefinitionguide_en.pdf
- Georgakopoulos, D. & Jayaraman, P.P. (2016) Internet of Things: from internet scale sensing to smart services. *Computing* 98(10), pp. 1041–1058
- IEEE Computer Society. (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard
- Jantunen, S. (2010). 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*, (pp. 131-140). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Kalenborn, A. (2010). Modelling by Example: Requirements engineering during the bidding stage of dialog-oriented software projects. In *Proceedings of the 1st Workshop on RE in*

Small Companies RESC, (pp. 158-166). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Kotonya, G. & Sommerville, I. (2003). *Requirements Engineering: Processes and Techniques*. England. John Wiley and Sons Limited.

Kourosfar, E., Shahir, H. Y. & Ramsin, R. (2009). Process Patterns for Component-Based Software Development. In G. A. Lewis, I. Poernomo & C. Hofmeister (eds.), CBSE (pp. 54-68). Springer. ISBN: 978-3-642-02413-9. doi: 10.1007/978-3-642-02414-6_4

Macasaet, R., Chung, L., Garrido, J., Rodriguez, M., & Noguera, M. (2011). 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*, (pp. 50-56). ACM New York, NY, USA. doi: 10.1145/2181101.2181114

Macasaet, R., Noguera, M., Rodriguez, M., Garrido, J., Supakkul, S., & Chung, L. (2012). 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*, Article 7, (pp. 1-6). ACM New York, NY, USA. doi: 10.1145/2424563.2424573

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S. & Chung, L. (2013). 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*, (pp.1-12), IEEE. ISBN: 978-1-4673-2912-5. doi: 10.1109/RCIS.2013.6577703

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2014). 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 5* (4), (pp. 71-90), IGI-Global.

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2019). *Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations*. *Journal of Universal Computer Science JUCS 25* (7), (pp. 764-787).

Mairiza, D., Zowghi, D. & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. *Proceedings of the 2010 ACM Symposium on Applied Computing* (p./pp. 311--317), New York, NY, USA: ACM. ISBN: 978-1-60558-639-7

Mendling, J., Recker, J., & Reijers, H. (2010). On the usage of labels and icons in business process modeling. *International Journal of Information System Modeling and Design*, 1 (2), pp. 40-58. doi: 10.4018/jismd.2010040103

Medvidovic, N. & Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26, 70--93.

Object Management Group, Inc. (2008). *Business Process Modeling Notation Version 1.1*. Last accessed on March 10, 2011 at <http://www.omg.org/spec/BPMN/1.1/PDF>

Object Management Group, Inc. (2009). *Unified Modeling Language Version 2.2*. Last accessed on March 10, 2011 at

<http://www.omg.org/spec/UML/2.2/Superstructure/PDF/changebar>

Object Management Group, Inc. (2010). MDA Foundation model. Needham, MA, USA

PSRI. (2018). Pentathlon Systems Resources Incorporated. Last accessed on December 15, 2018 at <http://www.pentathlonsystems.com>

RE-Tools. (2013). RE-Tools. Last accessed on October 9, 2013 at https://personal.utdallas.edu/~chung/Sam_Supakkul/RE-Tools/index.html

Respect-IT. (2007). KAOS Tutorial Version 1.0. Retrieved on January 15, 2013, from <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>

Sabre. (2018). Sabre Incorporated. Last accessed on December 15, 2018 at <http://www.sabre.com>

Segura, S., Durán, A., Troya, J., and Cortés, A.R. (2017). A Template-Based Approach to Describing Metamorphic Relations. *IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*, 2017, pp. 3-9, doi: 10.1109/MET.2017.3

Solemon, B., Sahibuddin, S., & Ghani, A.A.A. (2009). Requirements engineering problems and practices in software companies: An industrial survey. *Advances in Software Engineering*, Springer, 2009, pp. 70-77

Spain SME Statistics. (2021). Spanish Ministry of Commerce, Industry, and Tourism. Last accessed on June 30, 2021 at <http://www.ipyme.org/Publicaciones/CifrasPYME-enero2021.pdf>

Supakkul, S., & Chung, L. (2012). The RE-Tools: A Multi-notational Requirements Modeling Toolkit. In *Proceedings of the 20th International Requirements Engineering Conference RE* (pp. 333-334). IEEE. doi: 10.1109/RE.2012.6345831

Supakkul, S., Chung, L., Macasaet, R., Noguera, M., Rodriguez, M., & Garrido, J. (2013). 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*, (pp. 128-130). Last accessed on October 9, 2013 at http://ceur-ws.org/Vol-978/paper_23.pdf

Tan, L., & Wang, N. (2010). Future internet: the internet of things. In: 2010 3rd international conference on advanced computer theory and engineering (ICACTE), vol 5, (pp. 376-380). IEEE doi: 10.1109/ICACTE.2010.5579543

Veerappa, V., & Harrison, R. (2013). Assessing the maturity of requirements through argumentation: A good enough approach. *Automated Software Engineering ASE*, (pp. 670-675). IEEE. doi: 10.1109/ASE.2013.6693131

Virus. (2018). Virus Worldwide. Last accessed on December 15, 2018 at <http://www.virusworldwide.com>

Zhao, L., Letsholo, K., Chioasca, E., Sampaio, S., & Sampaio, P. (2012). Can business modeling bridge the gap between business and information systems? In *Proceedings of the 27th annual ACM symposium on applied computing SAC* (pp. 1723-1724). ACM New York, NY, USA. doi: 10.1145/2245276.2232054

Chapter V

Evaluation of μ BRPs

This chapter will discuss how μ BRPs have been evaluated. First, we discuss why μ BRPs are suitable for micro-business software projects in terms of comprehensibility, timeliness, and affordability. Then, we discuss how we have been evaluating μ BRPs using Action Research and Grounded Theory. Finally, we show how the evaluation approach for the μ BRPs has also been applied in other software development contexts.

1. THE SUITABILITY OF USING μ BRPs IN MICRO-BUSINESS SOFTWARE PROJECTS

The suitability of using μ BRPs in micro-business software projects could be characterized in terms of its contributions to the goals of comprehensibility, timeliness, and affordability. Comprehensibility refers to the ability of a micro-business owner (or software developer) to interpret and understand the μ BRPs correctly. Timeliness refers to the timely implementation of a software project. Affordability refers to the ability of a micro-business owner to afford (purchase) software.

For μ BRPs to be suitable in the micro-business domain, they have to be readily comprehensible by micro-business owners. We use one-on-one interviews for evaluating the comprehensibility of the μ BRPs for micro-business owners. We discuss these interviews in detail in Section 2.1.

μ BRPs are suitable for software developers because they are technically relevant and practical to use. This relevance and practicality are reflected in the timeliness and affordability goals. The number of man days expended in a software project directly affects the timeliness goal. The labor cost in a software project directly affects the affordability goal (where labor cost in a software project is derived by multiplying the number of man days expended in a software project by the daily software developer rate). The number of components reused in a software project affects the timeliness goal (by reducing the number of man days expended in a software project), which then affects the affordability goal (by reducing the cost equivalent of man days expended in a software project).

Action Research is when a research proposal is applied into practice and the researchers are actively participating. (Goldkuhl, 2008) (Goldkuhl, 2012) (Bilandzic & Venable, 2011).

Evaluating μ BRPs using Action Research was an ideal choice because we needed real-world data, specifically for evaluating man days, number of software components reused, and cost in real-world software development companies with micro-business projects. Also, the use of μ BRPs required training and influence from the researchers as done in Action Research. In case studies and field experiments, the influence of the researcher has to be minimized or even non-existent. Action Research is discussed in detail in Section 3.

Complementing Action Research with survey-type studies has been done by (Lee, 2002) and we do this in a similar fashion with the one-on-one interviews. A graphical representation which depicts the Action Research, as described above, evaluating the suitability of μ BRPs in micro-business software projects, can be conveniently expressed using SIGs as shown in Figure V.1.

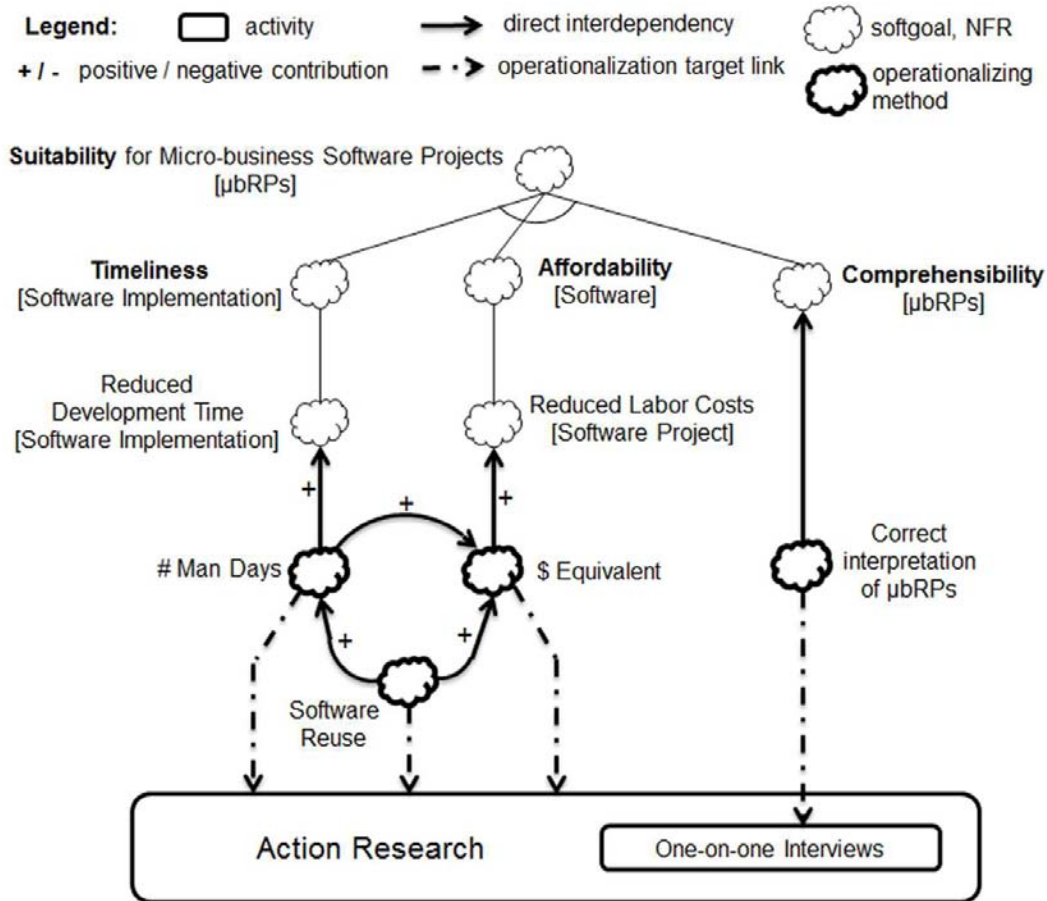


Figure V.1. Evaluating the suitability of µbRPs for micro-business software projects

2. EVALUATING COMPREHENSIBILITY

In reference to Figure V.1 – evaluating suitability, using µbRPs are suitable when they are comprehensible to the micro-business owners. If the µbRPs are not comprehensible to the micro-business owners then it would not improve the communication of requirements from and to the software developers. If a micro-business owner is unable to communicate his/her requirements correctly to the software developers then the success of the entire software project is jeopardized, misunderstood requirements being a major cause of project failure (Happel, 2010).

With the help of the participating software development companies, we have conducted one-on-one interviews in order to further understand whether the tables and models in our

proposal are “comprehensible enough” for micro-business owners. In every “project under study” in the Action Research, a participating software developer showed a table and a model, like the one shown in Figure V.2, representative of the project, to the micro-business owner and asked them for their interpretation of the documents and if they had any extra comments to make. Since the tables of the μ BRPs are straightforward and are already in “business-like” language, we did not make an evaluation form for the μ BRP tables. Instead, we focused on evaluating the comprehensibility of the μ BRP models for the micro-business owners.

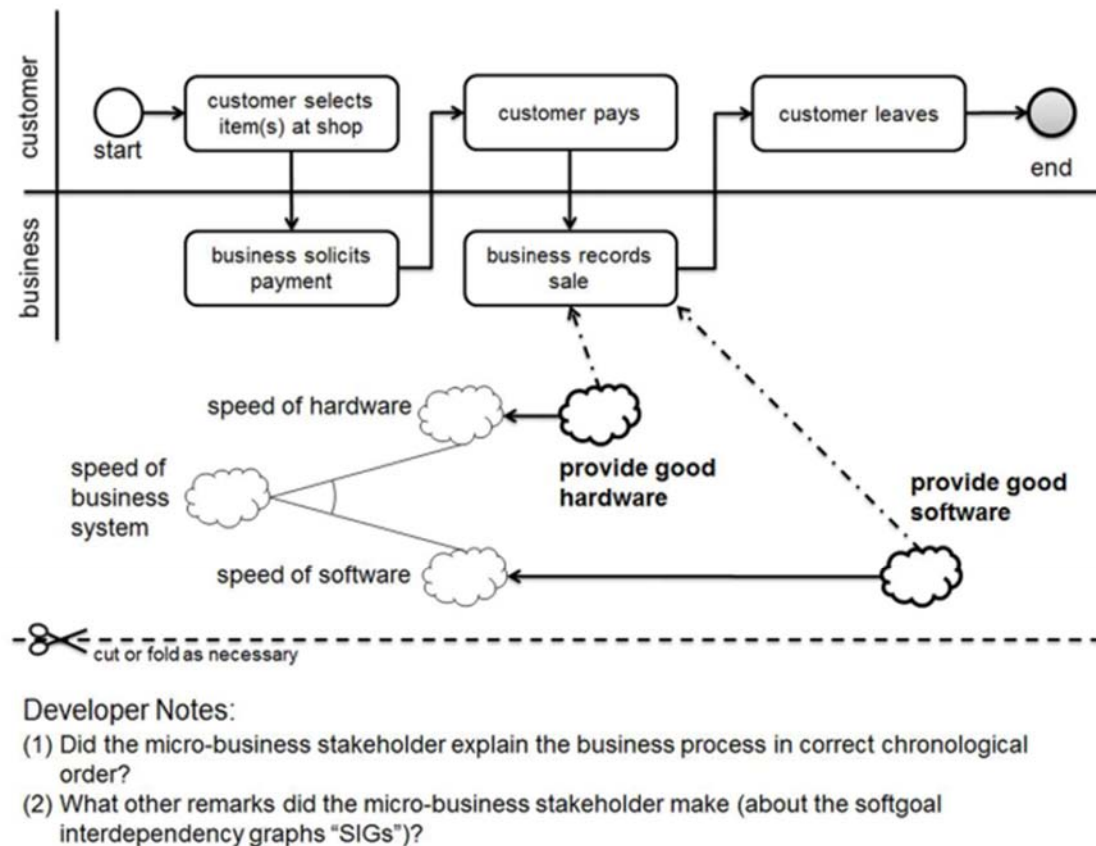


Figure V.2. Sample Model Comprehensibility Form

A sample μ BRP diagram comprehensibility form, in English, is shown in Figure V.2 and can also be found in Appendix C.1. The forms were also prepared in other languages, i.e., Spanish “Castellano,” the local language spoken in Granada, Spain and Filipino “Tagalog,” the local language spoken in Manila, Philippines.

There are two main questions in the μ bRP diagram comprehensibility form. The first question pertains to the explanation of the business process diagram (expressed in BPMN) in correct chronological order by the micro-business owner. The result of this question is either “correct” or “incorrect.” The second question is an open-ended question, geared towards the interpretation of the NFRs and infrastructure (expressed in SIGs) in the diagrams. The results of this question vary.

2.1 Compilation of Interview Results

From the period of June 2013 to December of 2013, we have compiled the results of 16 one-on-one interviews where stakeholders from the micro-business projects in the Action Research provided responses. Six micro-businesses were from Manila, where responses were documented in both English and Tagalog. Six micro-businesses were from Granada, where responses were documented in Spanish. Four micro-businesses were from Dallas (Texas, United States of America), where responses were documented in English.

The four micro-businesses from Dallas only participated in the evaluation of μ bRP diagram comprehensibility and not in the full Action Research (where timeliness and affordability were also evaluated). We decided to include some one-on-one interviews in Dallas because the area is also considered a micro-business hub and gathering relevant data in the area was an opportunity that we could immediately take advantage of. In 2011, Dallas had 42,068 micro-businesses based on a headcount of less than 10, representing approximately 69% of total businesses in the area (United States Census Bureau, 2011).

The recording of the responses was written on the forms by the researchers in the local languages and then translated with great care by the researchers into English for collation and presentation in this thesis, as shown in Table V.1. The responses to the second question (relating to NFRs) are condensed for presentation purposes in Table V.1, highlighting the most important input provided by the micro-business stakeholder.

Table V.1

Summary of the Results of the 16 one-on-one Interviews with Micro-business Owners

Software Developer	Micro-business Project	Location	Interviewee #	Results from Question 1 – BPMN interpretation	Results from Question 2 – other remarks, mainly related to the interpretation of SIGs. The results in this column are condensed for presentation purposes in this table. They are provided in more detail in the later paragraphs.
PSRI	Sales Mgt	Manila	#1	Correct	Good internet service is important for software to work as planned
PSRI	Sales Mgt	Manila	#2	Correct	Employees have to be well-trained when using the software
PSRI	Sales Mgt	Manila	#3	Correct	Good transportation and trained staff improve product availability
PSRI	Sales Mgt	Dallas	#4	Correct	Powerful hardware maximizes software performance
PSRI	Sales Mgt	Dallas	#5	Correct	Experienced cashiers make sales transactions faster
PSRI	Sales Mgt	Dallas	#6	Correct	Purchasing a fast printer makes sales transactions faster
PSRI	Sales Mgt	Dallas	#7	Correct	Good hardware, software, and internet contribute to system speed
Virus	CRM	Manila	#8	Correct	Purchasing back-up database is advisable
Virus	CRM	Manila	#9	Correct	Good internet makes data collection and uploading faster
Virus	CRM	Manila	#10	Correct	Experienced CRM users improve customer data management
Everyware	Patient Mgt	Granada	#11	Correct	Well-trained system users improve collection of patient data
Everyware	Patient Mgt	Granada	#12	Correct	Faster hard drives and powerful computers make system faster
Everyware	Patient Mgt	Granada	#13	Correct	Inexperienced system users negatively affect the entire system
Desarrollo TIC	Online Shop	Granada	#14	Correct	System users must be trained well to manage the online shop
Desarrollo TIC	Online Shop	Granada	#15	Correct	Product availability depends on inventory & logistics management
Desarrollo TIC	Online Shop	Granada	#16	Correct	A reliable hosting provider is important for the online shop

2.2 Potential Benefits of using SIGs in µbRP Diagrams

From the one-on-one interviews, we also grouped the responses to question 2 (from the µbRP diagram comprehensibility form as presented in Appendix C.1) based on how the SIGs in µbRP diagrams aid in the comprehension of the requirements of micro-businesses. The SIGs in µbRP diagrams are beneficial in three ways.

First, SIGs are able to show that external factors, outside the responsibility of the software developer, can affect the performance of the software, as observed from the following responses to question 2 of the µbRP diagram evaluation forms.

“It is important to get a contract with a reliable internet service provider in order for the software to work as expected.” [Interviewee #1]

“The employees at the store have to be well-trained in using the software system.” [Interviewee #2]

“Purchasing powerful hardware for the system will maximize the performance of the software.” [Interviewee #4]

“Experienced cashiers at the counter of the point-of-sale (POS) system are able to make faster sales transactions with customers.” [Interviewee #5]

“Purchasing a fast printer for providing receipts to the customer will make sales transactions faster at the counter.” [Interviewee #6]

“In the case of system failure for an unforeseen reason, it would be advisable to have the CRM database backed-up on a separate server.” [Interviewee #8]

“Inexperienced users of the patient management system will affect the entire system in a bad way.” [Interviewee #13]

“A reliable hosting provider must be contracted in order for the online shop to function as planned.” [Interviewee #16]

Second, SIGs are able to show which NFRs and operationalizing methods are directly related to business process activities, as indicated by the operationalization target link (dash-dot-dash arrow). This is observed from the following responses to question 2 of the evaluation forms as follows.

“Experienced cashiers at the counter of the point-of-sale (POS) system are able to make faster sales transactions with customers.” [Interviewee #5]

“Purchasing a fast printer for providing receipts to the customer will make sales transactions faster at the counter.” [Interviewee #6]

“Having a good internet connection will make collecting and uploading customer-related data to the CRM database much faster.” [Interviewee #9]

“Managing customer data is improved when there are experienced CRM users operating on the system.” [Interviewee #10]

“If the users of the patient management system are well-trained then the collection of patient data onto the system is going to get better.” [Interviewee #11]

“Users of the online shop must be trained well in order to manage the online shop properly.”
[Interviewee #14]

Third, SIGs are able to show how child NFRs can positively (or negatively) contribute to their parent NFRs. This is observed from the following responses to question 2 of the evaluation forms as follows.

“In order to improve the availability of the products at the store, there must be good transportation of items from the warehouse to the store and the staff must be able to perform logistics tasks well.” **[Interviewee #3]**

“For the system to function as fast as possible, it is important to have good hardware, software, and a reliable internet connection at all times.” **[Interviewee #7]**

“The entire system will work faster if there are faster hard drives (database servers) and more powerful computers in place.” **[Interviewee #12]**

“The availability of the products which are sold at the online store depends on the proper management of inventory and transportation of the goods (logistics).” **[Interviewee #15]**

Further observations and discussions on how SIGs aid in the comprehensibility of the µbRP diagrams are continued in the next chapter.

3. EVALUATING TIMELINESS AND AFFORDABILITY

Using Action Research, the goals of timeliness and affordability are evaluated in this Section. Based on Figure V.1, the promotion of software (component) reuse, resulting in reduction of man days expended in software implementation and lowering of labor costs, contributes to the technical relevance and the suitability of μ BRPs in micro-businesses software projects.

Our research team has always valued the documentation of 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 μ BRPs are applied in practice. In addition, the use of μ BRPs 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, 2008; Goldkuhl, 2012; Bilandzic & Venable, 2011), the Action Research in this thesis is reported as-is, step-by-step, for clarification purposes. In order to improve the validity of Action Research, (Kock, 2004) recommends the use of one or more of the following: units of analysis, multiple iterations, and/or Grounded Theory. We apply all three recommendations as detailed in the following subsections: 3.1, 3.2, and 3.3.

3.1. Units of Analysis

We begin the Action Research by identifying the 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 “number of man days expended during software implementation.” It starts on the first day of requirements gathering and ends on the day the project is accepted by the micro-business owner (or equivalent stakeholder). One man day is

equivalent to one software developer who has worked for eight hours. Labor cost per project can be derived by multiplying the number of man days by the daily rate for developers.

The second unit of analysis is the "number of software components reused" where the characterization of a software component is as mentioned in the introduction. A third unit of analysis could be made and it would be "the kind of μ BRPs used." With this third unit of analysis, other μ BRP proposals could also be evaluated using the Action Research presented in this paper.

3.2 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 of 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, 2011). Granada had 55,578 micro-businesses which comprise approximately 96% of total businesses, based on a headcount of less than 10 (Spain SME Statistics, 2011).

Four software development companies with micro-business projects decided to learn and adapt the μ BRPs – (a) Pentathlon Systems Resources Incorporated (PSRI, 2018), (b) Virus Worldwide (Virus, 2017), (c) Everyware Technologies (Everyware, 2017), and (d) Desarrollo TIC (Desarrollo TIC, 2017). 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 similar nature as possible. In the "previous implementation," no μ BRPs 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, 2004).

Before the "implementations under study" took place, mandatory face-to-face training sessions involving two developers from each software development company were required.

The following were the training sessions that the developers went through and these sessions can be viewed in further detail in Appendix C.3. Each training session took no longer than one hour.

- (1) Tutorial 1: Basic BPMN and SIGs
- (2) Tutorial 2: Pattern Representation
- (3) Tutorial 3: Component Representation
- (4) Tutorial 4: Tool Support
- (5) Tutorial 5: Unified Modeling Language
- (6) Overview of μ BRPs and the User Guide
- (7) Participating on the Evaluation of μ BRPs

Afterwards, the developers could use the training material (PSRI Action Research, 2021), supporting tools (RE-Tools, 2014) and accompanying documentation (Supakkul & Chung, 2012; Supakkul et al., 2013), and could contact the researchers anytime via email, videoconference, or mobile for any μ BRP-related support.

Since PSRI has contributed in developing the μ BRPs under study since 2010, a fresh perspective involving two new hires developed a sales management system from scratch, their “previous implementation,” without using μ BRPs. Table V.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 V.2 also shows the exerted effort needed to set-up and maintain the μ BRPs for use in each of the participating companies. This is measured from the time when there had been no μ BRP knowledge up to the time when there were at least 10 μ BRPs that could be used. Set-up efforts consist of (1) training the software developers and analysts regarding the creation and the use of μ BRPs and (2) setting up the component libraries, μ BRP tables, and optional μ BRP illustrations.

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

Table V.2

Observing μ BRPs in Industrial Practice using Action Research																		
"reuse" refers to % of software components reused, "man days" refers to man days expended in deployment																		
Software Development Company	Micro-business Project	Previous Project (no μ BRPs used)		Projects Under Study												Savings average reduction in man days		
		reuse %	man days	1st		2nd		3rd		4th		5th		6th				
				reuse %	man days	reuse %	man days	reuse %	man days	reuse %	man days	reuse %	man days	reuse %	man days	reuse %	man days	
PSRI	Sales Mgt	0	18	20	10	30	8	30	8	30	8	30	8	30	8	30	8	9.33
Virus	CRM	20	34	50	26	50	24	50	22	50	24	50	24	50	24	50	20	10.67
Everyware	Patient Mgt	10	14	20	10	20	10	30	8	30	8	30	10	30	10	30	10	4.67
Desarrollo TIC	Online Shop	10	18	30	12	40	10	40	10	40	10	40	10	40	10	40	10	7.67

3.3 Grounded Theory

In order to take the results in Table V.2 in proper context, application of the third recommendation of (Kock, 2004) in validating Action Research, Grounded Theory, is discussed and applied in this subsection.

3.3.1 A Brief Background on Grounded Theory

Grounded Theory traces its roots back from sociologists (Glaser & Strauss, 1967). There are three basic "types" of Grounded Theory which are the original by (Glaser & Strauss, 1967) also referred to as "classical Glaserian Grounded Theory," a formalized and procedural one by (Strauss & Corbin, 1990), and one which clarifies ontological and epistemological ambiguities (Charmaz, 2006). We adapt the "classical Glaserian Grounded Theory" which has been recommended, used, and applied recently in the field of software engineering (Kock, 2004; Carver, 2006; Crabtree et al., 2009; Adolph et al., 2011; Macasaet, 2018). All references to Grounded Theory in this chapter 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 & Strauss, 1967).

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 explain what is going on. Using Grounded Theory in software engineering is useful when trying to answer the question, “What is going on here?”

(Schreiber & Stern, 2001) 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 as “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.

3.3.2 Grounded Theory in the Practice of Action Research

On a practical level, the Grounded Theory in this thesis is adapted for the specific needs of Action Research as recommended by (Kock, 2004). The results from Table V.2 may be used to develop causal models (Bagozzi, 1980; Davis, 1985) which are considered as the highest level of abstraction in Grounded Theory. The causal models link independent, moderating, intervening, and dependent variables (Arnold, 1982; Baron & Kenny, 1986; Creswell, 1994; Drew & Hardman, 1985). The variables may be classified in terms of units of analysis (as explained in Subsection 3.3.1), which can be measured or estimated numerically or non-numerically (Drew & Hardman, 1985; Gregory & Ward, 1974; Pervan & Klass, 1992). From the results in Table V.2, we build a causal model, shown in Figure V.3, which includes activities, variables (as units of analysis, represented as an operationalizing method using SIGs), and the possible grounded theories we could build from the data.

Table V.2 shows that the use of the μ BRPs 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 of μ BRPs. Some of the grounded theories that we can build from the data (and from memos) include (but are not limited to):

- (1) the use of μ BRPs improve component reuse
- (2) the use of μ BRPs improve communication between micro-business owners and software developers
- (3) improved communication improves project acceptance rates
- (4) improved communication reduces the number of man days expended in projects
- (5) training for using μ BRPs improves morale and the use of μ BRPs
- (6) the use of μ BRPs have a complimentary effect with other reuse methods
- (7) the use of μ BRPs increases the awareness of (software) reuse in general.

Such grounded theories provoke questions such as: Was the improvement in software component reuse due to the use of μ BRPs alone or due to a symbiosis between the use of μ BRPs 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 μ BRPs, how many reduced man days can be attributed to improved communication? When evaluating the use of μ BRPs in real world settings, is it possible to isolate human variables, which are indispensable in the field of software engineering?

Using SIGs, Figure V.3 shows us “what is going on here,” based on the grounded theories built from the Action Research data. The operationalizing methods in bold are used to depict project acceptance, man days expended, labor cost, and number of software components reused. These are the methods which contribute to the timelines and affordability goals, refined from the goal of suitability of μ BRPs for micro-businesses. From the operationalizing methods, target links depict the grounded theories which lead to the activities performed in micro-businesses which contribute to the success of the software projects. Within the activities, there are also grounded theories that link some activities together. Instead of trying to jump to conclusions, the grounded theories better prepare us to ask the right questions as we continue with our future work as discussed in the next chapter.

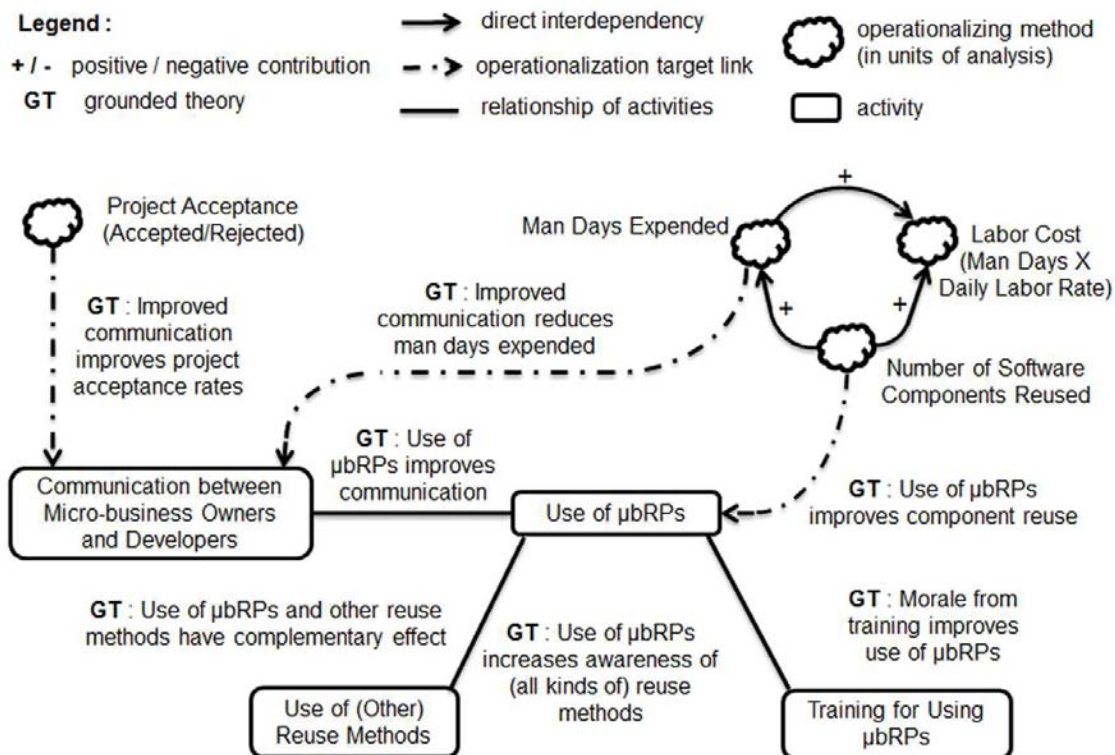


Figure V.3 A causal model showing grounded theories built from the Action Research data

4. APPLICATION OF OUR EVALUATION APPROACH IN OTHER CONTEXTS

As part of our future work, we believe that μ BRPs could be useful as a starting point for writing User Stories in an Agile context. Instead of starting from scratch, the μ BRPs could be used as a starting point for writing User Stories in micro-businesses using Agile methodologies for their software projects. For exploratory purposes in 2018, we asked Axiom Practice Management “AXPM,” a software company currently known as Greyfinch (Greyfinch, 2019), if they wanted to do Action Research in their current Agile methodology implementation. Greyfinch is a Software as a Service “SaaS” Company that has been in operation since 2012 and serves the healthcare industry in the United States of America, mainly in the states of Arkansas and Texas. In 2018, it had annual revenues of US\$ 80 Million and a headcount of a little more than 400 people.

The Action Research would allow us to probe and see the possibilities where µbRPs could also be applicable in an Agile context. The Action Research approach used to evaluate the µbRPs presented in this thesis (Macasaet et al., 2014; Macasaet et al., 2019) was used for the Action Research at Greyfinch. In sections 4.1 through 4.3, both the Agile methodology and Action Research used at Greyfinch are discussed in detail.

4.1 A Brief Background of the Scrum Framework

The Scrum Framework is a popular Agile framework being used nowadays and is the framework being used by Greyfinch to develop software. The Scrum Framework traces its origins from Professor Hirotaka Takeuchi and Ikujiro Nonaka and later on to Jeff Sutherland and Ken Schwaber who published the Scrum Framework (Scrum.org, 2018).

The Scrum Framework can be described as a framework that allows the management and development of complex products in an efficient, creative and focused manner. It aims to deliver products with the highest possible value. Scrum is a useful Project Management tool and can be used for software development as well as operations management within an organization. It also enables an organization to become “Agile” by maximizing responsiveness and adaptiveness to changing customer needs.

There are various roles in the Scrum Framework. The Product Owner or PO is responsible for maximizing the value of the work of the development team by being up-to-date with the market and ensuring the profitability of the product. Although the PO is responsible for understanding the business objectives of the product, they are allowed to delegate market research activities and surveys as long as objectives are not compromised for the Scrum Team.

The Development Team is made up of software developers, testers, business analysts, user experience designers, and testers. The Development Team is a team which is capable of delivering a working version of the software to the customers. The better the chemistry of the team, the more likely it is to succeed (Macasaet, 2017). There is sometimes confusion that the Development Team is only made up of software developers because of the name but the truth is that anybody who contributes to the development of the final product is part of the Development Team.

Finally, there is the Scrum Master who must ensure that the Scrum Framework is working in the organization. The Scrum Master ensures that those accountable for deliverables are made accountable. The Scrum Master is also responsible for removing any impediment that is hampering the Development Team from delivering the next working version of the product.

The events in the Scrum Framework are contained in a Sprint which usually lasts for one week or up to four weeks. At Greyfinch, the sprints were two weeks long. Sprint Planning is a Sprint Event which lasts eight hours for a four-week Sprint. In Greyfinch, the Sprint Planning session was four hours long every two weeks.

During Sprint Planning, the Development Team commits to do items found on a Product Backlog, an artifact where all the items to be developed are listed in terms of priority. These items are software requirements and usually referred to in Agile terminology as User Stories. User Stories usually have the form: "As a [role], I would like to [goal], so that [value]." For example, "As a User of a Sales System, I would like to Know my Sales, so that I know if I am meeting my Sales Targets for the current period." These User Stories are the "requirements" and are items in the Product Backlog. Each item on the Product Backlog is estimated with Story Points. Story Points are an estimate of the amount of effort needed to finish a product backlog item. This is an arbitrary estimate and does not correspond directly to man hours expended.

Every day, throughout the sprint, there is a Scrum Daily which lasts for a maximum of 15 minutes. Those attending briefly summarize what they did yesterday, what they are doing today, and if there is anything preventing them from doing what they have to do today.

At the end of the Sprint, there is a Sprint Review or a Sprint Demo where the Development Team presents their work to the Product Owner for acceptance. The Sprint Demo lasts for four hours long if the Sprint is for four weeks. In Greyfinch, the Sprint Demo lasts for two hours long for the two-week Sprints.

Finally, there is the Sprint Retrospective. The Sprint Retrospective lasts three hours long for an eight-week Sprint. At Greyfinch, the Sprint Retrospective lasts for one hour and thirty minutes. During the Sprint Retrospective, those who attend talk about what went right and wrong in the current Sprint and what could be improved in the next Sprint. All of the events in the Scrum Framework are time-boxed, meaning that the team members respect the maximum amount of time that can be spent in an event: at Greyfinch, four hours for Sprint Planning, 15

minutes for the Scrum Daily, two hours for the Sprint Demo, and one and a half hours for the Sprint Retrospective.

4.2 Just in Time Demos

The Scrum Framework has been successful for several teams however there are still teams who have not been successful (Lopez-Martinez et al., 2016). Some teams attempt to modify the Scrum Framework so that it could be more tailor-made to some software companies. This is called Scrumbut. Scrumbut is criticized because according to Scrum.org, those who intend to modify or change the Scrum Framework would not be able to take full advantage of what the Scrum Framework has to offer (Scrum.org, 2018).

Even if the Scrum Framework was working properly at Greyfinch, the team wanted to keep improving which led to the introduction of Just in Time “JIT” Demos. A JIT Demo is a demonstration of the work done by a developer to the Product Owner as soon as an item on the backlog is finished and both of them are available to meet (Macasaet, 2018). Normally, all finished items are presented by the developers at the end of a Sprint, during the Sprint Review or Sprint Demo. The completion of the items is based on how the developers have met the requirements as stated in the User Stories.

JIT Demos aim to have (1) faster feedback loops between the Product Owner and the Development Team and (2) demos with better quality. The trade-off that has to be made for JIT Demos is that there is no strict time-box that has to be followed which may result in more time being spent on them throughout the duration of the Sprint. An illustration of how JIT Demos have changed the Scrum Framework is shown in Figure V.4.

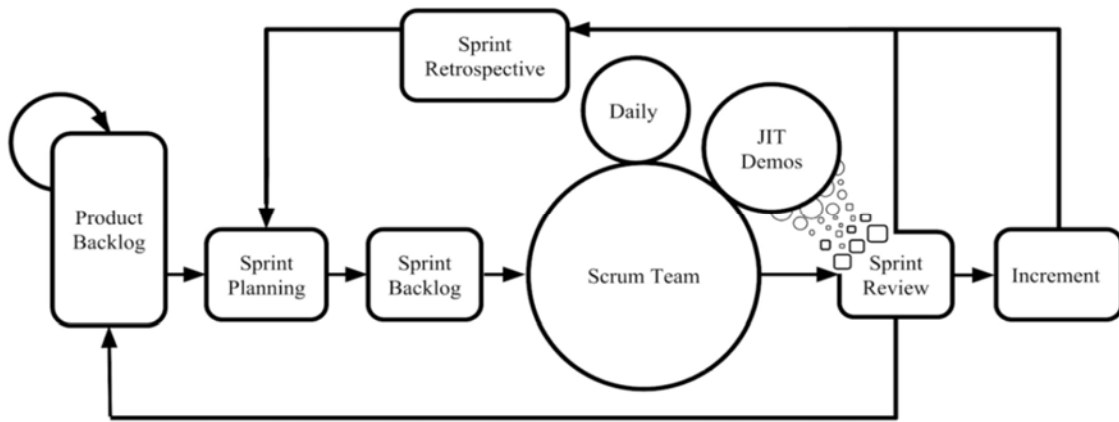


Figure V.4 Modification of the Scrum Framework in order to make way for JIT Demos (Macasaet, 2018)

4.3 Evaluation of Just in Time Demos

JIT Demos were evaluated in 2018 using the same approach we have used to evaluate μ RBPs. The units of analysis were both the Story Points finished in each Sprint and Product Owner Hours spent in each Sprint. The multiple iterations were several Sprints, the first five consecutive Sprints without using JIT Demos and the second five consecutive Sprints using JIT Demos. The results are shown in Table V.3.

Table V.3 Units of Analysis and Multiple Iterations with JIT Demos (Macasaet, 2018)

Sprint #	Demo Type	Story Points	PO Hours Spent
1	Standard	73	2.25
2	Standard	68	1.75
3	Standard	71	2.25
4	Standard	73	2
5	Standard	74	2.25
6	Standard	58	1.5
7	JIT	81	3.75
8	JIT	73	3.5
9	JIT	71	3.5
10	JIT	68	3.25
11	JIT	70	3.5
12	JIT	69	3.75

It is important to note that in Table V.3, POs spend an average of two hours in the Sprint Demo when the standard Scrum Framework is followed. In this case, the Development Team has to wait until the end of the Sprint to get feedback from the PO regarding whether a User Story is completed or not.

When Just in Time Demos are used, the POs spend almost double the amount of time during each Sprint. In this scenario, POs provide immediate feedback to the developers on the Scrum Team. There is a faster feedback loop for the Development Team to continue their work and they no longer have to wait until the end of the Sprint to start working on items on the Backlog.

From the units of analysis and the multiple iterations, a causal model of the Grounded Theory is presented in Figure V.5.

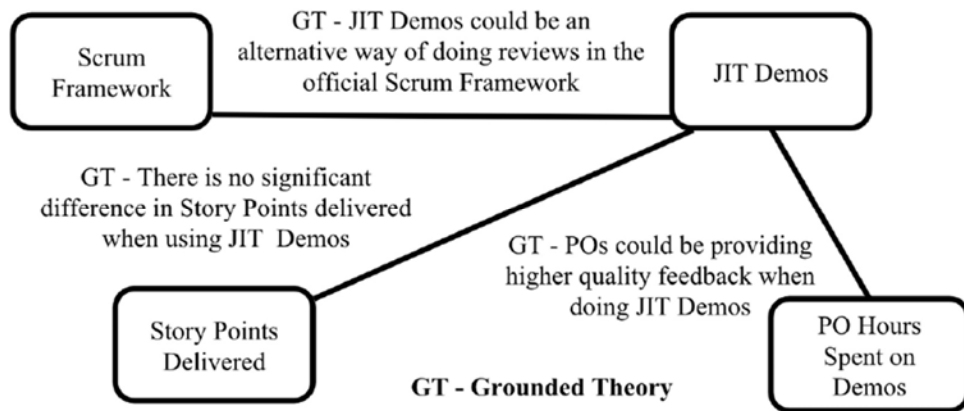


Figure V.5 Using Grounded Theory to make a Causal Model of JIT Demos
(Macasaet, 2018)

As a result of the causal model, some of the following questions for future research came up:

1. Does the time of the demo really have to be at the end of the Sprint (during the Sprint Review)?
2. Could the authors of the Scrum Framework consider JIT demos as an alternative within the Scrum Framework? Warnings are made about “Scrumbut” although industry cases vary from one to another and maybe exceptions could be made.
3. Is it better for POs to spend more time on JIT Demos instead of compressing all their feedback for all the demos during the Sprint Review Time Box?

These new research questions were relevant for Greyfinch to conduct future studies and further improve their software development practice. From the Action Research point-of-view, our research team has seen the value of the evaluation approach for exploring areas which have not been fully studied, e.g., JIT demos in Scrum, and coming up with research questions that could not have been formulated without the use of Action Research. For our future work, our research team has seen that we could continue applying Action Research in software companies using Agile methodologies and further explore the possible advantages that µBRPs could bring, such as how µBRPs could help in writing User Stories in Agile contexts.

5. CONCLUSIONS

In this chapter, we have presented the evaluation approach used for μ BRPs and how this evaluation approach could be applied in other contexts. As discussed, the evaluation approach has demonstrated value in industry by discovering more problems that need solutions and more questions that need answers. There is value in discovering the right questions to ask and the right problems to solve. As one famous thinker would say: "The greatest challenge to any thinker is stating the problem in a way that will allow a solution." - Bertrand Russell. In the next chapter, we discuss the observable strengths and weaknesses of μ BRPs based on the experiences from the evaluation approach as discussed in this chapter. We also discuss our future work in the next chapter.

CHAPTER REFERENCES

- Adolph, S., Hall, W. & Kruchten, P. (2011). Using Grounded Theory to study the experience of software development. *Empirical Software Engineering*, 16, (pp. 487-513). doi:10.1007/s10664-010-9152-6
- Arnold, H. (1982). Moderator variables: A clarification of conceptual, analytic, and psychometric issues, *Organizational Behavior and Human Performance*, (29) 2, (pp. 143-174), ISSN 0030-5073, doi:10.1016/0030-5073(82)90254-9
- Bagozzi, R. P. (1980). *Causal models in marketing*. New York, NY: Wiley. ISBN: 0471015164
- Baron, R. & Kenny, D. (1986). The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51, (pp. 1173—1182). doi:10.1037/0022-3514.51.6.1173
- Bilandzic, M. & Venable, J. (2011). Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics. *J. Community Informatics*, 7. Last accessed on October 9, 2013 at <http://ci-journal.net/index.php/ciej/article/view/786/804>
- Carver, J. (2006). 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* (pp. 42). Springer. ISBN: 978-3-540-71300-5. doi: 10.1007/978-3-540-71301-2_15
- Charmaz, K. (2006). *Constructing Grounded Theory: a practical guide through qualitative analysis*. London; Thousand Oaks, Calif.: Sage Publications.
- Crabtree, C. A., Seaman, C. B. & Norcio, A. F. (2009). Exploring language in software process elicitation: A Grounded Theory approach. *ESEM* (pp. 324-335), ISBN: 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5315984
- Creswell, J. W. (ed.) (1994). *A Qualitative Procedure in Research Design. Qualitative and Quantitative Approaches*. London and New Dheli: Sage.
- Davis, J. A. (1985). *The Logic of Causal Order* (Vol. 07-055). Beverly Hills, London, New Delhi: Sage.
- Desarrollo TIC. (2018). *Desarrollo TIC*. Last accessed on December 15, 2018 at <http://www.desarrollotic.com>
- Drew, C.J. & Hardman, M.L. (1985). *Designing and Conducting Behavioral Research*. Pergamon, New York, NY.
- Everyware. (2018). *Everyware Technologies*. Last accessed on December 15, 2018 at <http://www.everywaretech.es>
- Glaser, B. G., Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New York, NY: Aldine de Gruyter.

Goldkuhl, G. (2008). Practical Inquiry as Action Research and Beyond. In W. Golden, T. Acton, K. Conboy, H. van der Heijden & V. K. Tuunainen (eds.), ECIS (pp. 267-278). Last accessed on October 9, 2013 at <http://aisel.aisnet.org/ecis2008/118>

Goldkuhl G. (2012). From Action Research to practice research. *Australasian Journal of Information Systems*, 17, 2, (pp. 57-78). url: <http://dl.acs.org.au/index.php/ajis/article/view/688>.

Gregory, D. & Ward, H. (1974). *Statistics for Business Studies*. McGraw-Hill, London, England.

Greyfinch. (2019). Greyfinch.com. Last accessed on August 17, 2018

Happel, H.J., Maalej, W., & Seedorf, S. (2010). Applications of ontologies in collaborative software development. In I. Mistrík, J. Grundy, A. Hoek, & J. Whitehead (Eds.), *Collaborative Software Engineering* (pp. 109-129). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-10294-3_6

Kock, N. (2004). The three threats of Action Research: a discussion of methodological antidotes in the context of an information systems study. *Decision Support Systems*, 37 (2), (pp. 265-286). doi: 10.1016/S0167-9236(03)00022-8

Lee, O. (2002). An Action Research report on the Korean national digital library. *Information & Management*, 39, (pp. 255-260). doi: 10.1016/S0378-7206(01)00094-5

López-Martínez, J., Juárez-Ramírez, R., Huertas, C., Jiménez, S. & Guerra-García, C. (2016). Problems in the Adoption of Agile-Scrum Methodologies: A Systematic Literature Review. In 4th International Conference in Software Engineering Research and Innovation, Puebla, pp. 141-148.

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2014). 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* 5 (4), (pp. 71-90), IGI-Global.

Macasaet, R. J. (2017). The Project Start Review Group. In M. Brambilla, T. Hildebrandt (eds.), *Proceedings of the Industry Track of the 15th International Conference on Business Process Management BPM*, (pp. 81-87).

Macasaet, R.J. (2018). Just in Time Demos in the Scrum Framework. *Proceedings of the 3rd International Conference on System Reliability and Safety ICSRS*, (pp.21-24).

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2019). *Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations*. *Journal of Universal Computer Science JUCS* 25 (7), (pp. 764-787).

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

Philippines MSME Statistics. (2011). Philippine Department of Trade and Industry. Last accessed on December 5, 2013 at <http://www.dti.gov.ph/dti/index.php?p=321>

PSRI. (2018). Pentathlon Systems Resources Incorporated. Last accessed on December 15, 2018 at <http://www.pentathlonsystems.com>

PSRI Action Research. (2021). Action Research Material for Micro-businesses. Last accessed on December 31, 2021 at <http://www.pentathlonsystems.com/ar4mb.html>

RE-Tools. (2013). RE-Tools. Last accessed on October 9, 2013 at https://personal.utdallas.edu/~chung/Sam_Supakkul/RE-Tools/index.html

Schreiber, R. & Stern, P. (2001). Using Grounded Theory in Nursing. Springer Publishing Company, New York.

Scrum.org. (2018). The Scrum Guide. Last accessed on August 17, 2018

Spain SME Statistics. (2011). Spanish Ministry of Industry, Energy, and Tourism. Last accessed on December 5, 2013 at http://www.ipyme.org/Publicaciones/ESTADISTICAS_PYME_N10_2011.pdf

Strauss, A. & Corbin, J. (1990). Basics of qualitative research: Grounded Theory procedures and techniques. Sage Publications, Basics of Qualitative Research

Supakkul, S., & Chung, L. (2012). The RE-Tools: A Multi-notational Requirements Modeling Toolkit. In Proceedings of the 20th International Requirements Engineering Conference RE (pp. 333-334). IEEE. doi: 10.1109/RE.2012.6345831

Supakkul, S., Chung, L., Macasaet, R., Noguera, M., Rodriguez, M., & Garrido, J. (2013). 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*, (pp. 128-130). Last accessed on October 9, 2013 at http://ceur-ws.org/Vol-978/paper_23.pdf

United States Census Bureau. (2011). County Business Patterns. Last accessed on December 5, 2013 at <http://censtats.census.gov/cgi-bin/cbpnaic/cbpcomp.pl>

Virus. (2018). Virus Worldwide. Last accessed on December 15, 2018 at <http://www.virusworldwide.com>

Chapter VI

Conclusion

After presenting and evaluating the μ BRPs in the previous chapters, this concluding chapter discusses the observable strengths, weaknesses, and future work that is planned out for μ BRPs.

1. OBSERVABLE STRENGTHS

Based on the experience of the participating software development companies in the Action Research, we enumerate the observable strengths when using μ BRPs.

First, the μ BRP table during requirements elicitation and analysis has been useful. The tables are outright and straightforward which made it comprehensible for micro-business owners without technical backgrounds. In just one step, the micro-business owners simply had to answer questions in business language without compromising technical details for the software developers. There was no need of going back-and-forth explaining requirements with technical jargon. The μ BRP table also contains a lot of domain knowledge which is useful for understanding the context of an implementation. The μ BRP table saves a lot of time while maintaining the quality of the requirements.

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

The models indicate the activities in business processes and the associated software components which are critical and which directly or indirectly relate to NFRs. For example, the use of operationalization target links connects the 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 models 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 μ BRPs. 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, then the length of projects could be shortened. Improved communication and promoting software component reuse by using μ BRPs 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.

Fourth, Action Research was applied in a different context in industry and it had favorable feedback from the collaborators, i.e., the use of Action Research in evaluating Just in Time Demos in the Scrum Framework, as mentioned in the previous chapter. Positive feedback from using Action Research is encouraging for our research teams. This provides us with more motivation to continue Action Research with μ BRPs and to continue doing Action Research in more contexts in industry as well.

2. DISCUSSIONS ON WEAKNESSES AND LIMITATIONS

Based on the experience of the participating software development companies in the Action Research, we also discuss the weaknesses and limitations when using μ RPs.

First, although Action Research lacks the rigorousness of other methods, e.g., controlled experiments, field experiments, surveys, case studies, etc., it makes up for its shortcomings when researchers make positive real-world contributions in the day-to-day operations of practitioners. The participating software development companies improved their software component reuse and reduced man days in the “projects under study,” resulting in tangible savings. The practitioners, having an orientation to results, placed high value on this kind of outcome and were zealous to say that the use of μ RPs were indeed suitable for micro-business software projects.

As researchers, making claims on whether the savings were due to the use of μ RPs is still unascertainable at this moment. Multiple iterations of Action Research in the future could demonstrate observable trends through statistical analysis, eventually paving the way for stronger conclusions. In relation to multiple iterations, the kind of mutual collaboration demonstrated in this Action Research motivates practitioners to continue participating in Action Research, allowing researchers to carry-on collecting and analyzing valuable real-world data.

The second discussion relates to the software development companies that participated and plan to participate in Action Research. PSRI has been involved in developing and improving the μ RPs since 2010. There will be favorable biases when creators are asked about the opinion of their own work. Virus, Everyware, and Desarrollo TIC have strong ties to PSRI, got special attention in terms of training and support, and were enthusiastic when using and applying the μ RPs. The results of the Action Research could have been different if random software development companies with micro-business projects participated. We are interested in finding out if the use of μ RPs would have similar results in other software development companies without strong ties to PSRI. Hence, we have made all Action Research available to the public so that others may conduct independent studies at their own convenience. We are also collecting feedback from anonymous μ RP users by regularly engaging in workshops, tutorials, and conferences.

The third discussion is related to the comprehension of the μ BRPs by micro-business owners. Since the tables of μ BRPs are comprehensible in a straightforward, business-type way, it was the comprehensibility of the models which was being evaluated in the one-on-one interviews. When asked to explain the processes in the models of the μ BRPs in chronological order, all 16 of the interviewed micro-business owners were able to do it correctly. All 16 interviewees also provided relevant responses when interpreting the SIGs in the models. We see this as an indication that the models of the μ BRPs could be comprehensible to micro-business owners, on top of the straightforward comprehensibility of the tables.

However, since 16 one-on-one interviews merely represent a tiny portion of the entire micro-business domain, making any claims on μ BRP model comprehensibility is still inconclusive. We plan to conduct more evaluations on the comprehensibility of the μ BRPs using comprehension testing, which can capture a larger population sample of the micro-business domain. We extend this discussion of future work in the next section.

The fourth and last discussion relates to the unknowns. The main concern at the start of the Action Research was whether using the proposed μ BRPs would be suitable for micro-business software projects. During the course of the Action Research, more unknowns surfaced as shown using Grounded Theory, leaving several open-ended questions as mentioned in the previous chapter. It would have been unlikely for us to ask such detailed research questions beforehand. The full set of relevant questions regarding the suitability of the μ BRPs for micro-business software projects is yet to be uncovered. As researchers, we value the discovery of the right questions because asking the right questions build a good foundation for research work.

3. CONCLUSIONS AND FUTURE WORK

This thesis has proposed μ BRPs and has discussed its application in industrial practice. First, the domain of micro-businesses was discussed in detail, including how requirements differ from large businesses. Second, research work related to patterns and to the evaluation of proposals based on their technical relevance or comprehensibility was reviewed. Third, the practicality of using models in the micro-business domain was discussed, particularly BPMN, UML, and SIGs. Fourth, the conceptual model, the tables, views, modes, instantiations, component model, use in practice, and the supporting tool, RE-tools, for the μ BRPs were presented. Fifth, the suitability of μ BRPs for micro-business software projects was evaluated by four software development companies using Action Research. The four software development companies provided observations related to the technical relevance of μ BRPs. Specifically, in relation to how μ BRPs affect timeliness and affordability in micro-business software projects. The comprehensibility of μ BRPs by micro-business owners was evaluated using one-on-one interviews with the help of the participating software development companies in the Action Research. Finally, the strengths and weaknesses of the μ BRPs and its evaluation were discussed.

Responding to the first main research question, as stated in the introduction – how should requirements be represented so that they would be comprehensible for micro-business owners and technically relevant for software developers? Representing μ BRPs in a comprehensible manner for micro-businesses and in a technical manner for software developers is difficult since both goals are somewhat conflicting. Throughout our research, we have learned that in order to meet the demands of both parties, not one, but multiple languages, notations, models, and point-of-views used in a complementary manner are recommended, providing different views for the users as has been described in this thesis. In addition, these multiple languages, notations, and models, must be fit for use in the micro-business domain. There should not be unnecessary cognitive load for understanding and using such models. Future work related to μ BRP representation involves the continuous improvement of modeling the relationships among BPMN, SIGs, and UML. Proposed lightweight (or even ultra-lightweight) models, harmonizing BPMN, SIGs, and UML, which are apt for the micro-business domain, are currently being developed. RE-Tools is also being continuously improved based on constructive feedback from current users.

Responding to the second main research question – how could the proposed requirements approach be evaluated? Evaluating the μ BRPs using Action Research allowed us to promote the use of μ BRPs, collect real-world data when the μ BRPs are applied, contribute to software process improvement efforts in real-world software development companies, and come up with more relevant research questions based on Grounded Theory for use in our future work. In addition, the evaluation approach was also applied in another context and was shown to be helpful in practice.

We plan to continue conducting more iterations of Action Research, building and evaluating more grounded theories, hopefully in collaboration with other software development companies with micro-business projects, aside from those that have participated in this Action Research. Ideally, the software development companies that will participate in future iterations of Action Research should not have strong ties to PSRI. Since the Action Research material is available to the public, software development companies can conduct their studies at their own convenience. Other μ BRP proposals may also be evaluated using the Action Research presented in this paper. We are constantly collecting feedback from anonymous μ BRP users via workshops, tutorials, and conferences.

We are continuing to conduct one-on-one interviews with micro-business owners in order to further understand the comprehensibility of μ BRPs. As future work, we plan to further understand μ BRP comprehensibility using other means such as comprehensibility tests in order to have a larger sample size, representative of the micro-business domain. As we continue collecting constructive feedback, both from real-world micro-business owners and software developers, we plan to continuously improve the μ BRPs along the way.

Key characteristics for micro-business software are lightweight requirements analysis and software design that are unlikely to go wrong, leading to faster implementation and delivery. It is important for us to bring the complex research of software requirements closer and more applicable to micro-businesses. They are an important driver for economies and they could use all the help they can get nowadays. Given the several physical restrictions from the global pandemic and constantly evolving computing paradigms, now would be the right time to put requirements research into practice and help micro-businesses stay afloat. Our requirements approach could help micro-businesses take advantage of the latest devices, infrastructure, and business trends by providing requirements analysis and software design that is comprehensible, affordable, and technically feasible for them.

The abundance of Software-as-a-Service “SaaS” which could be utilized by several micro-businesses also provides an opportunity for the application of μ BRPs. Just like Commercial-of-the-shelf “COTS” systems, SaaS could also be viewed as stand-alone components for meeting requirements. We plan to include more SaaS based solutions in the modes and instantiations of the μ BRPs as part of our future work.

The reusability of μ BRPs with associated software components in Component Based Software Engineering “CBSE” provide us with a starting point for investigating more non-functional attributes and characteristics of the associated software components such as their maintainability, usability, and efficiency. For instance, as the μ BRPs and their associated software component repositories grow in participating software development companies, more challenges such as the maintainability of the software components in practice have to be taken into consideration. We are also keen in investigating reusability of the components within the context of Model-Driven Development.

In other contexts, given the small-scale nature of micro-businesses, we believe that μ BRPs could be applied in various Agile Software Development Frameworks such as Scrum, particularly aiding in the discovery and writing of Epics, User Stories, and their Acceptance Criteria. μ BRPs could also aid in writing test cases in Test-Driven Development.

Notwithstanding issues, we feel that we have done among the first studies on μ BRPs, obtaining some lessons and observations, and provoking relevant discussions which would be helpful in future work.

Capítulo VI

Conclusión

Después de presentar y evaluar los μ BRPs en los capítulos anteriores, este capítulo final analiza las fortalezas y debilidades observables y el trabajo futuro que se planea para los μ BRPs.

1. FORTALEZAS OBSERVABLES

Según la experiencia de la participación de empresas de desarrollo de software en la investigación de acción, enumeramos las fortalezas observables cuando se usan los μ BRPs.

Primero, la tabla μ RP ha sido útil durante la obtención de requisitos. Las tablas son directas y sencillas. Son comprensibles para los propietarios de microempresas sin conocimientos técnicos. En un solo paso, los propietarios de microempresas respondían preguntas de requisitos en lenguaje natural sin comprometer los detalles técnicos para los desarrolladores de software. No había necesidad de explicar los requisitos con palabras técnicas. También, la tabla μ RP tiene mucha información del dominio que ha sido útil para comprender el contexto de la implementación. La tabla μ RP ahorra mucho tiempo manteniendo la calidad de los requisitos.

En segundo lugar, los modelos de μ RP han sido útiles para usar los componentes de software, específicamente para su reutilización e implementación. Los modelos muestran a los desarrolladores de software que existen oportunidades para reutilizar componentes de software en ciertos contextos de procesos comerciales y que los desarrolladores de software podrían aprovechar estas oportunidades de reutilización si es posible. También, los modelos con componentes de software asociados dan información sobre las relaciones entre los componentes de software. Usando las palabras claves específicas y los nombres de archivo que se encuentran en los modelos, los desarrolladores de software pueden buscar componentes de software en los repositorios con más orientación y velocidad, sabiendo buscar de antemano cuáles son los componentes de software asociados.

Los modelos indican las actividades en los procesos comerciales y los componentes de software asociados que son críticos y que se relacionan directa o indirectamente con los NFRs. Por ejemplo, el uso de enlaces de objetivos de operacionalización conecta los NFRs. Priorizando qué actividades comerciales y qué componentes de software son críticos. Eso permite que los desarrolladores de software concentren sus esfuerzos en tareas más importantes y que finalmente contribuyen en la tasa de éxito de las implementaciones de software. El uso de los modelos proporciona tanto a los desarrolladores de software como a los propietarios de microempresas una visión general más clara para la implementación del software. Eso evita los puntos de vista miopes. Ser consciente de los muchos factores que afectan la implementación del software permite que tanto los desarrolladores de software como los propietarios de microempresas ejerzan esfuerzos conscientes en áreas críticas para el éxito del proyecto de software.

En tercer lugar, según *Grounded Theory*, observamos que la duración total de las implementaciones podría haberse disminuido debido al uso de μ RP. Cuando se mejora la comunicación interna dentro de una implementación, por ejemplo, la comunicación entre los desarrolladores de software y la comunicación externa con el cliente, la duración de los proyectos podría disminuir. La comunicación mejorada y la promoción de la reutilización de componentes de software mediante el uso de μ RP podrían estar relacionadas con la reducción del tiempo de desarrollo y, finalmente, con la reducción del tiempo total de implementación. En consecuencia, los tiempos de implementación más cortos podrían haberse convertido en menos esfuerzo por día y costos de implementación más bajos, resultando que el software sea más asequible para las microempresas conscientes de sus recursos.

Cuarto, la *Action Research* se aplicó en un contexto diferente en la industria y tuvo resultados favorables con los colaboradores - el uso de *Action Research* en la evaluación de Just in Time Demos en Scrum Framework, como se mencionó en el capítulo anterior. Los resultados positivos del uso de *Action Research* han sido de alguna forma positivo para nuestros equipos de investigación. Esto nos motiva para continuar con *Action Research* con μ RPs y también para otros contextos en la industria.

2. DISCUSIÓN SOBRE DEBILIDADES Y LIMITACIONES

Según la experiencia de las empresas de desarrollo de software participantes en *Action Research*, discutimos las debilidades y limitaciones cuando se usan μ RPs.

Primero, aunque *Action Research* falta la rigurosidad de otros métodos, por ejemplo, experimentos controlados, casos de estudio, encuestas, etc., compensa sus deficiencias cuando los investigadores hacen contribuciones positivas al mundo real en las operaciones diarias. Las empresas participantes de desarrollo de software mejoraron la reutilización de sus componentes de software y redujeron los días-hombre en los “proyectos en estudio”, lo que resultó en ahorros tangibles. Los profesionales en el mundo real, que tienen una orientación hacia los resultados, proporcionaron un gran valor a este tipo de resultados y se mostraron entusiastas al decir que el uso de μ RPs era realmente adecuado para proyectos de software de microempresas.

Como investigadores, todavía no se puede determinar si los ahorros se debieron al uso de μ RP en este momento. Múltiples iteraciones en *Action Research* en el futuro podrían demostrar tendencias observables a través del análisis estadístico, lo que eventualmente allanaría el camino para conclusiones más sólidas. En relación con las iteraciones múltiples, el tipo de colaboración mutua demostrada en esta *Action Research* motiva a los profesionales a seguir participando, lo que les permite continuar recopilando y analizando datos valiosos del mundo real.

La segunda discusión se relaciona con las empresas de desarrollo de software que participaron y tienen planes en participar en *Action Research*. PSRI ha estado involucrado en el desarrollo y la mejora de μ RPs desde 2010. Habrá sesgos favorables cuando se pregunte a los creadores sobre la opinión de su propio trabajo. Virus, Everywhere y Desarrollo TIC tienen fuertes colaboraciones con PSRI, recibieron atención especial en términos de capacitación y soporte, y se mostraron entusiastas al usar y aplicar μ RPs. Los resultados de *Action Research* podrían haber sido diferentes si participaran empresas aleatorias de desarrollo de software con proyectos de microempresas. Estamos interesados en averiguar si el uso de μ RPs tendría resultados similares en otras empresas de desarrollo de software sin fuertes vínculos con PSRI. Por lo tanto, hemos puesto a disposición al público todo el material de *Action Research* para que otros puedan hacer estudios independientes a su conveniencia.

También, recopilamos comentarios de usuarios anónimos de μ BRPs que han participado regularmente en talleres, tutoriales y conferencias.

La tercera discusión es la comprensión de μ BRPs por parte de los microempresarios. Dado que las tablas de μ RP son comprensibles de una manera sencilla y de tipo comercial, fue la comprensibilidad de los modelos lo que se evaluó en las entrevistas individuales. Cuando se les pidió que explicaran los procesos en los modelos de los μ RP en orden cronológico, los 16 propietarios de microempresas entrevistados pudieron hacerlo correctamente. Los 16 entrevistados también proporcionaron respuestas relevantes al interpretar SIGs en los modelos. Vemos esto como una indicación de que los modelos de μ BRPs podrían ser comprensibles para los propietarios de microempresas, además de la comprensibilidad directa de las tablas.

Sin embargo, dado que las 16 entrevistas simplemente representan una pequeña porción de todo el dominio de la microempresa, hacer afirmaciones sobre la comprensibilidad del modelo μ RP aún no es concluyente. Planeamos hacer más evaluaciones sobre la comprensibilidad de μ BRPs mediante pruebas de comprensión, que pueden capturar una muestra de población más grande del dominio de microempresas. Ampliamos esta discusión del trabajo futuro en la siguiente subsección.

La cuarta y última discusión se relaciona con las incógnitas. La pregunta inicial al comenzar *Action Research* era si el uso de μ BRPs sería adecuado para proyectos de software de microempresas. Durante el curso de *Action Research*, surgieron más incógnitas con el uso de *Grounded Theory*, dejando varias preguntas abiertas como se mencionó en el capítulo anterior. Habría sido poco probable que hiciéramos preguntas de investigación tan detalladas de antemano. Aún no se ha descubierto el conjunto completo de preguntas relevantes sobre la idoneidad de μ BRPs para proyectos de software de microempresas. Como investigadores, valoramos el descubrimiento de las preguntas adecuadas porque las preguntas adecuadas construyen una buena base para el trabajo de investigación.

3. CONCLUSIONES Y TRABAJO FUTURO

Esta tesis ha propuesto μ BRPs y ha presentado su aplicación en la industria. Primero, presentó en detalle el dominio de las microempresas, incluyendo cómo los procesos de requisitos son diferentes en las grandes empresas. Segundo, revisaron trabajos de investigación relacionados con patrones y con la evaluación de propuestas en función de su pertinencia técnica o comprensibilidad. Tercero, presentó la practicalidad de usar modelos en el dominio de la microempresa, particularmente BPMN, UML y SIGs. Cuarto, presentaron el modelo conceptual, las tablas, las vistas, los modos, las instancias, el modelo de componentes, el uso en la práctica y la herramienta, RE-tools, para μ BRPs. Quinto, cuatro empresas de desarrollo de software evaluaron la idoneidad de μ BRPs para proyectos de software de microempresas mediante *Action Research*. Las cuatro empresas de desarrollo de software proporcionaron observaciones relacionadas con la relevancia técnica de μ BRPs. Específicamente, en relación con la forma en que μ BRPs afectan la puntualidad y la asequibilidad en proyectos de software para microempresas. La comprensibilidad de μ BRPs para los propietarios de microempresas se evaluó mediante entrevistas individuales con la ayuda de las empresas participantes de desarrollo de software en *Action Research*. Finalmente, se discutieron las fortalezas y debilidades de μ BRPs y su evaluación.

En respuesta a la primera pregunta de nuestra tesis, como se indicó en la introducción, *¿cómo se deben representar los requisitos para que sean comprensibles para los propietarios de microempresas y técnicamente relevantes para los desarrolladores de software?* Representar μ BRPs en una manera comprensible para las microempresas y también técnicamente útil para los desarrolladores de software es difícil, ya que ambos objetivos son algo contradictorios. A lo largo de nuestra investigación, hemos aprendido que para satisfacer las demandas de ambas partes, se recomienda el uso de no uno, sino múltiples lenguajes, notaciones, modelos y puntos de vista de manera complementaria, proporcionando diferentes puntos de vista como se ha descrito en esta tesis. Además, estos múltiples lenguajes, notaciones y modelos deben ser aptos para su uso en el dominio de las microempresas. No debería existir una carga cognitiva innecesaria para comprender y utilizar tales modelos. El trabajo futuro relacionado con la modelación de μ BRPs implica la mejora continua de las relaciones entre los modelos de BPMN, SIGs y UML. Actualmente, se están desarrollando propuestas de modelos ligeros (o incluso ultraligeros), que armonizan BPMN, SIG y UML, y que son aptos para el dominio de las microempresas. RE-Tools también se mejora continuamente en función de los comentarios constructivos de los usuarios actuales.

En respuesta a la segunda pregunta de nuestra tesis: *¿cómo se puede evaluar la propuesta de μ RPs?* La evaluación de μ RPs mediante *Action Research* nos permitió promover el uso de μ RPs, recopilar datos del mundo real cuando se aplican μ RPs, contribuir a los esfuerzos de mejora de procesos de software en empresas de desarrollo de software del mundo real y generar preguntas de investigación más relevantes basadas en *Grounded Theory* para su uso en nuestro trabajo futuro. Además, la manera de evaluación se aplicó en otro contexto y demostró ser útil en la práctica.

Planeamos continuar realizando más iteraciones de *Action Research*, construyendo y evaluando más teorías por *Grounded Theory*, con suerte en colaboración con otras empresas de desarrollo de software con proyectos de microempresas, además de aquellas que ya han participado en la *Action Research*. Idealmente, las empresas de desarrollo de software que participarán en futuras iteraciones de *Action Research* no deberían tener fuertes lazos con PSRI. Dado que el material de *Action Research* está disponible para el público, las empresas de desarrollo de software pueden realizar sus estudios a su conveniencia. También, se pueden evaluar otras propuestas de μ RP utilizando la manera de *Action Research* presentada en esta tesis. Constantemente, recopilamos comentarios de usuarios anónimos de μ RPs a través de talleres, tutoriales y conferencias.

Continuamos realizando entrevistas individuales con propietarios de microempresas para comprender mejor la comprensibilidad de μ RPs. Como trabajo futuro, planeamos comprender mejor la comprensibilidad de μ RPs utilizando otros medios, como pruebas de comprensibilidad, para tener un tamaño de muestra más grande y más representativo del dominio de microempresas. Mientras recopilamos comentarios constructivos, tanto de propietarios de microempresas del mundo real como de desarrolladores de software, planeamos mejorar continuamente los μ RPs a lo largo del camino.

Las características clave del software para microempresas son el análisis de requisitos ligeramente y el diseño de software donde hay poca probabilidad de que salga mal. Con estas, se lleva a una implementación y entrega más rápida. Es importante para nosotros llevar y hacer más aplicable la compleja investigación de los requisitos de software a las microempresas. Son un motor importante para las economías y les vendría bien toda la ayuda que puedan obtener hoy en día. Dadas las diversas restricciones físicas por la pandemia global y la tecnología en constante evolución, ahora sería el momento adecuado para poner en práctica la investigación de requisitos y ayudar a las microempresas a mantenerse en

operaciones. Nuestra propuesta de requisitos podría ayudar las microempresas a aprovechar los dispositivos, la infraestructura y las tendencias comerciales más recientes, proporcionando análisis de requisitos y diseño de software comprensible, asequible y técnicamente factible para ellos.

La abundancia de Software-as-a-Service "SaaS" que podría ser utilizado por varias microempresas también da una oportunidad para el uso de μ BRPs. Igual que los sistemas Commercial-of-the-Shelf "COTS," SaaS también podría verse como componentes independientes para cumplir con los requisitos. Planeamos incluir más soluciones basadas en SaaS en los modos e instancias de μ BRPs como parte de nuestro trabajo futuro.

La reutilización de μ BRPs con componentes software asociados en Component Based Software Engineering "CBSE" nos da un punto para empezar a investigar más atributos y características no funcionales de los componentes de software asociados, como su mantenimiento, usabilidad y eficiencia. Por ejemplo, cuando crecen los μ BRPs y sus repositorios de componentes software asociados en las empresas participantes de desarrollo de software, se deben tener en cuenta más desafíos, como el mantenimiento de los componentes software en la práctica. También estamos interesados en investigar la reutilización de los componentes en el contexto de *Model-Driven Development*.

En otros contextos, dada la pequeña escala de las microempresas, creemos que μ BRPs podrían aplicarse en varios *Agile Frameworks* de desarrollo de software como Scrum. Ayudaría particularmente en el descubrimiento y redacción de épicas, historias de usuarios y sus criterios de aceptación. También, los μ BRPs podrían ayudar a escribir casos de prueba en el entorno de *Test-Driven Development*.

A pesar de los retos y dificultades, creemos que hemos realizado uno de los primeros estudios sobre μ BRPs, obteniendo algunas lecciones y observaciones, y provocando discusiones relevantes que serían útiles en el trabajo futuro.

Appendix A

Initial Catalogue of μ BRPs

1. **Name:** Inventory (Macasaet et al., 2014)

Context: This pattern has common micro-business inventory-related requirements. The basic inventory process involves the storage, retrieval, or checking/verification of physical items by an inventory clerk in a storage facility.

Keywords: item, inventory, storage, retrieval, warehouse, stock, stockroom, checking, verification

Problem: The problem and requirements are about managing how inventory is stored, retrieved, and managed in micro-businesses.

Solution: Several solutions could solve these problems such as manual recording, the use of barcodes, and scanners, to name a few. Storing items can be done in one or several places.

Inventory µbRP		
description: this pattern has common micro-business inventory-related requirements. The basic inventory process involves the storage, retrieval, or checking/verification of physical items by an inventory clerk in a storage facility.		
keywords: item, inventory, storage, retrieval, warehouse, stock, stockroom, checking, verification		
Functional Requirements Section (Q&A to be answered by micro-business owner)		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choices
Does the µb need to identify the users of the system?	yes/no, choose one	
How does the clerk record the storage/retrieval of items?	manual (pen and paper) / Automatic (barcode, RFID, etc.), select as many that apply	
How does the clerk verify/check/monitor items in inventory?	manual (visually), automatic (system query), select as many that apply	
Where is inventory information stored?	manually / local database / shared database / cloud database, select as many that apply	
What item information must be stored?	date stored- retrieved / item info / supplier data / customer data / other data, select as many that apply	
Is the inventory system linked to a sales system? (sales µbRP)	yes / no, choose one	
How many storage facilities are there?	single / multiple, choose one	
Non-functional Requirements Section, to be ranked in terms of priorities of the micro-business owner		
The ranking (1st as top priority) makes the developer and the µb owner aware of the soft goals or NFR priorities.		
- Micro-business Side -		
Question - Of what importance is _____ to the µb owner?	Ranking	
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
minimization of spoilage in inventory		
maximization of inventory storage capacity		
exactness (preciseness) of inventory data		
Question - Of what importance is _____ to the customers of the µb owner?		
the quickness (responsiveness) of the software system		
the security of the transactions of the software system		
availability of the product/service during purchase		
- Developer Side -		
Question - Of what importance is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

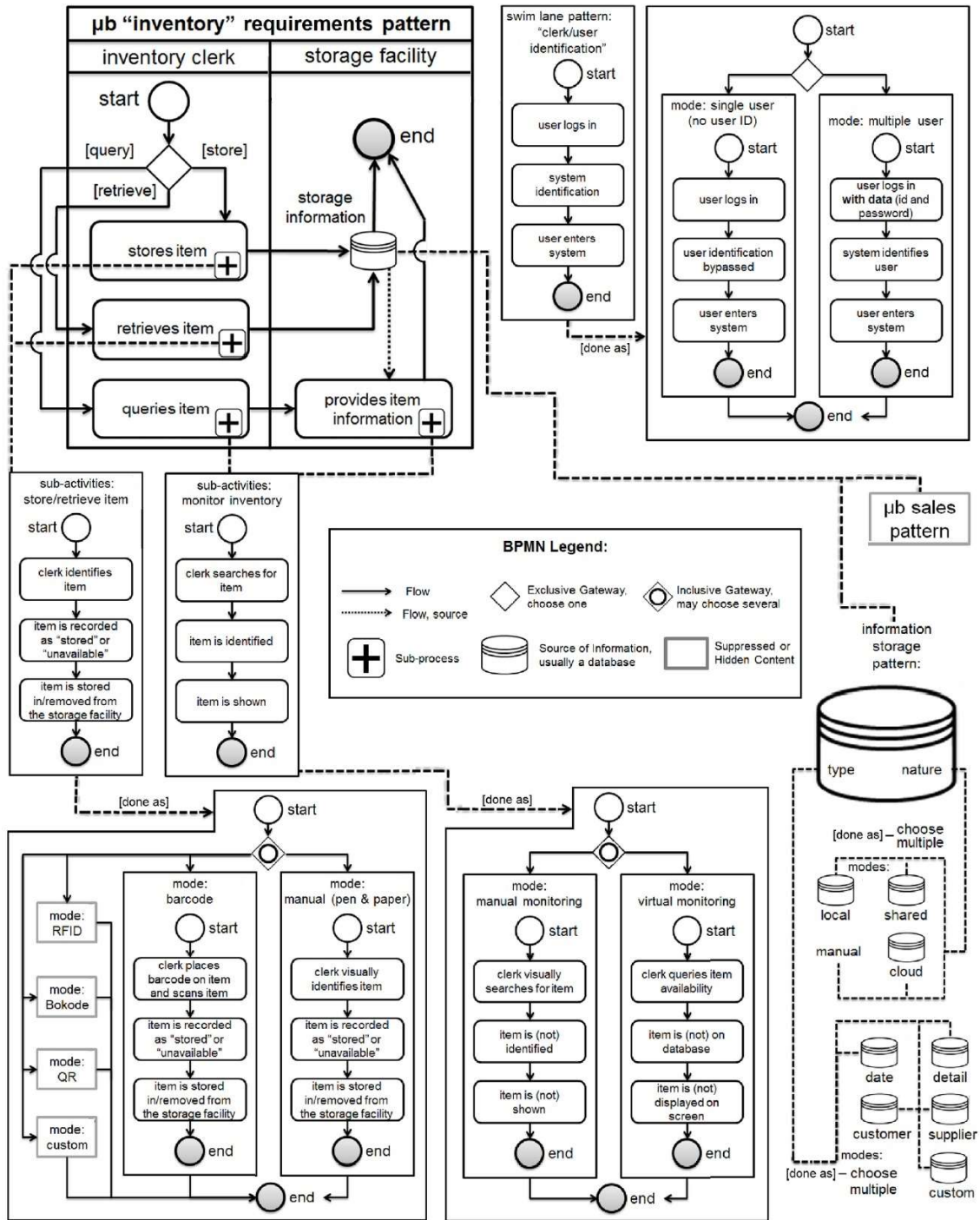


Figure A.1 Inventory BPMN Example

2. **Name:** Sales (Macasaet et al., 2013)

Context: This pattern has common micro-business sales-related requirements. The basic sales process involves a customer who is willing to purchase a product or solicit a service from a micro-business in exchange for a monetary amount or any other means applicable (such as coupons or gift cheques).

Keywords: sales, sale, point-of-sale, POS, online sale, cash, credit card, debit, cheque, gift certificate, coupon

Problem: The problem and requirements are about managing how customers are shopping and paying for items in micro-business establishments. Also, the micro-business has to record sales activity for reporting purposes, internally and/or externally on a case-to-case basis.

Solution: The solutions could come in the form of automated point-of-sale "POS" systems which involve payments in cash, credit card, coupons, to name a few or recording of such sales manually.

Sales µbRP		
description: this pattern has common micro-business sales-related requirements. The basic sales process involves a customer who is willing to purchase a product or solicit a service from a micro-business in exchange for a monetary amount or any other means applicable (such as coupons or gift cheques). This µbRP is also presented in (Macasaet et al., 2013).		
keywords: sales, sale, point-of-sale, POS, online sale, cash, credit card, debit, cheque, gift certificate, coupon		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Questions	Options	Choices
How does the customer shop?	in a physical shop / online, select as many that apply	
Does the µb need to identify the users of the system?	yes / no, choose one	
How does the µb prefer to identify his items on sale?	manual verification / automated, choose one	
Does the µb need to know item details?	yes / no, choose one	
How does the customer pay?	by cash / credit-debit / other, select as many that apply	
How does the µb want to record his sales?	does not want to / automated POS / other, select as many that apply	
How many sales points does the µb have?	single / multiple (specify #), choose one	
Non-functional Requirements Section		
The ranking structure (1 as top priority) allows the developer and the µb owner to understand the soft goals or the non-functional		
- Micro-business Side -		Ranking (1 as top priority)
Question - Of what importance is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
minimization of spoilage in inventory		
availability of products in inventory		
exactness (preciseness) of inventory data		
Question - Of what importance is _____ to the customers of the µb owner?		
the quickness (responsiveness) of the software system		
the security of the transactions of the software system		
availability of the product/service during purchase		
- Developer Side -		
Question - Of what importance is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

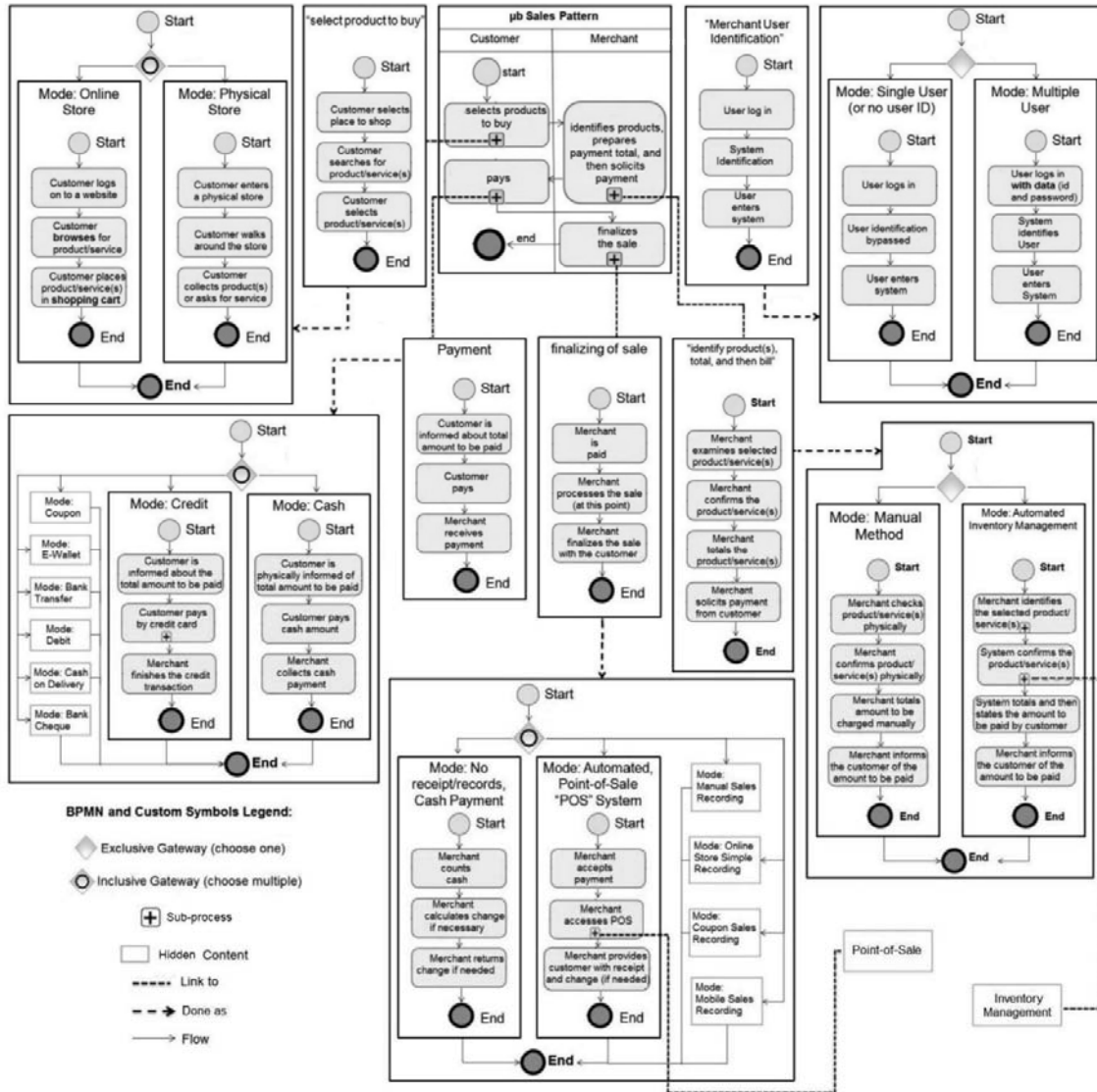


Figure A.2 Sales BPMN Example

3. **Name:** Logistics

Context: This pattern involves recurring requirements related to logistics, supply chain, and distribution in micro-businesses. The basic logistics process involves procurement of physical goods, storage (see inventory µbRP), and distribution of physical goods to retailers or distributors.

Keywords: logistics, supply chain, storage, distribution, physical goods, transportation

Problem: The problem and requirements are about managing the way micro-businesses are sourcing, checking, distributing, and delivering items throughout their distribution network.

Solution: The various solutions for micro-businesses would be purchasing from suppliers, distributing through delivery contractors, and sourcing internally through production and delivery through in-house staff, to name a few.

Logistics µbRP		
description: this pattern involves recurring requirements related to logistics, supply chain, and distribution in micro-businesses. The basic logistics process involves procurement of physical goods, storage (see inventory µbRP), and distribution of physical goods to retailers or distributors.		
keywords: logistics, supply chain, storage, distribution, physical goods, transportation		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
How does the µb owner procure physical goods?	purchase order / bidding / scheduled deliveries / other, select as many that apply	
How does the µb owner transport procured goods?	contracted courier / in-house transport / other, select as many that apply	
How does the µb owner store supplier information?	manually / local server / shared server / cloud / other, select as many that apply	
How does the µb owner check the quality of procured goods?	manually / automatically / other / does not check, select as many that apply	
How does the µb owner store physical goods? (see inventory µbRP)	in-house / third party / other, select as many that apply	
How does the µb owner transport/distribute stored goods?	contracted courier / in-house transport / other, select as many that apply	
How does the µb owner store distributor information?	manually / local server / shared server / cloud / other,select as many that apply	
How does the µb owner track location of physical goods?	per location / real-time (assisted) / other / does not track location, select as many that apply	
How does the µb manage return/replacement of physical goods?	customer mails good(s) / µb picks up good (s) / no return policy, select as many that apply	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
speed of delivery of physical goods		
exactness (preciseness) of logistics data		
How important is _____ to the customers of the µb?		
the quickness (responsiveness) of the system for orders/returns		
the security of the transactions of the software system		
availability of the physical good(s) during purchase		
timeliness of the arrival of the physical good(s)		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

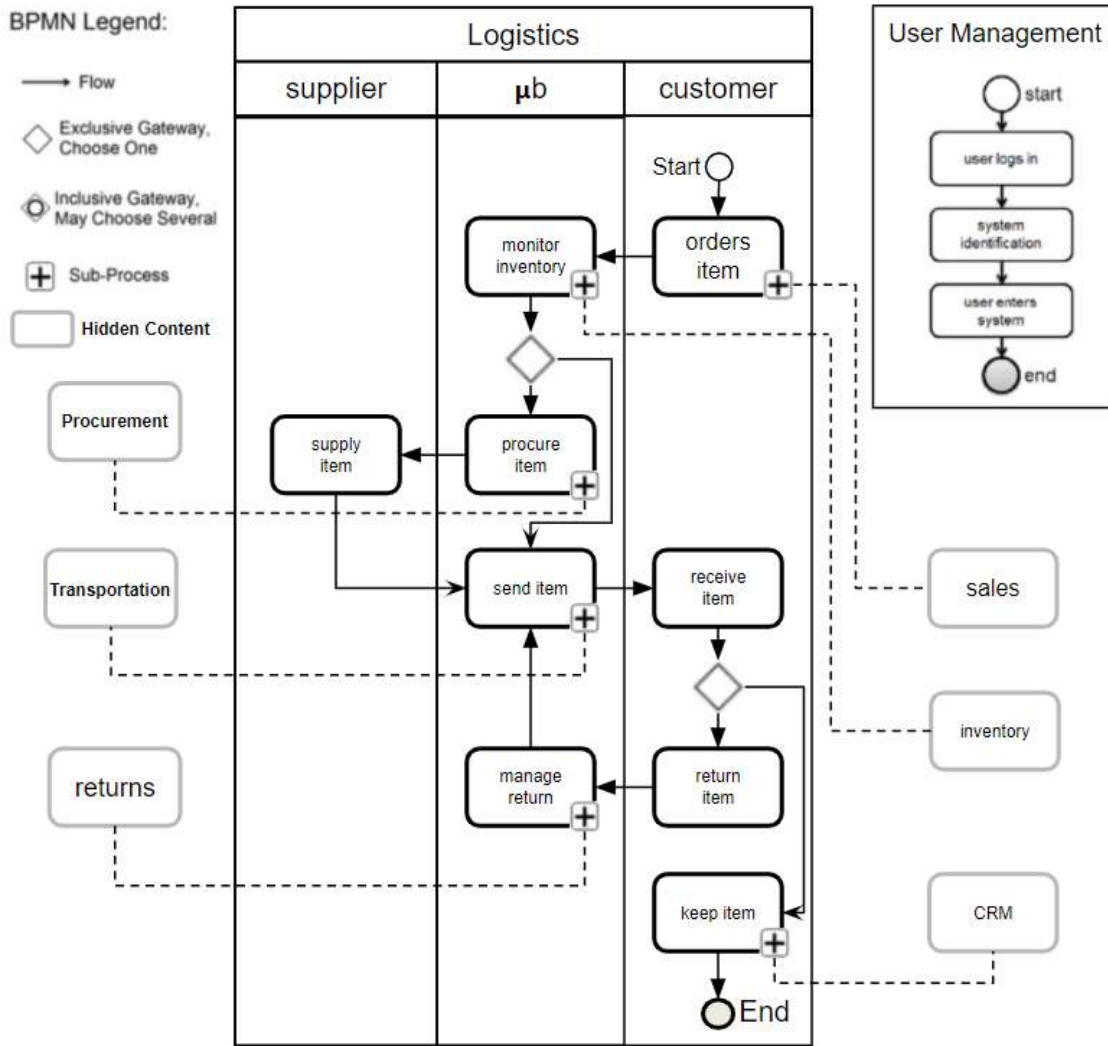


Figure A.3 Logistics BPMN Example

4. **Name:** Production

Context: This pattern involves recurring requirements related to the production of goods in a micro-business context. The basic production process includes use of input goods or material, production of goods by means of machinery or labor, and storage of the finished goods in inventory.

Keywords: production, manufacturing, labor, man days, overhead, machinery, cost of goods sold "COGS", finished goods, work-in-progress

Problem: The problem and requirements are about how the production of goods and items are managed in a micro-business such as doing it in batches, by job, or flow. Other challenges are the way production equipment is managed, dealing with excess, and shortages, to name a few.

Solution: The solutions involve Just-in-time inventory "JIT", Economic Order Quantity "EOQ", Activity Based Costing "ABC", to name a few.

Production µbRP		
description: this pattern involves recurring requirements related to the production of goods in a micro-business context. The basic production process includes use of input goods or material, production of goods by means of machinery or labor, and storage of the finished goods in inventory.		
keywords: production, manufacturing, labor, man days, overhead, machinery, cost of goods sold (COGS), finished goods, work-in-progress		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
How does the µb prefer to record input costs for production (bill of materials "BOM")?	automatically (through logistics µbRP) / manually, select one	
How does the µb prefer to track the availability/shortages of input material for production?	automatically (through inventory µbRP) / manually, select one	
How does the µb ensure availability of input material for production?	just-in-time (JIT) / economic order quantity (EOQ), select one	
How does the µb prefer to produce goods? (calculation of direct costs depends on this as well)	by job / batch / flow, select one	
How does the µb prefer to calculate the indirect costs of producing goods?	activity-based costing (ABC) / manual overhead allocation / other, select many	
How does the µb prefer to depreciate tools and equipment used for producing goods? (linked to accounting µbRP)	straight-line / declining / annuity in units of production / sum of years digits / other, select one	
How does the µb prefer to monitor the storage of produced goods?	automatically (through inventory µbRP) / manually, select one	
How does the µb prefer to deal with wastage or excess input material?	recycling / by-products / other, select many	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
availability of input material for production		
the speed of production of goods		
exactness (preciseness) of production data		
How important is _____ to the customers of the µb?		
availability of the physical good(s) during purchase		
quality of the physical good(s) produced		
responsiveness when product(s) are returned by the customer for repair or under warranty		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

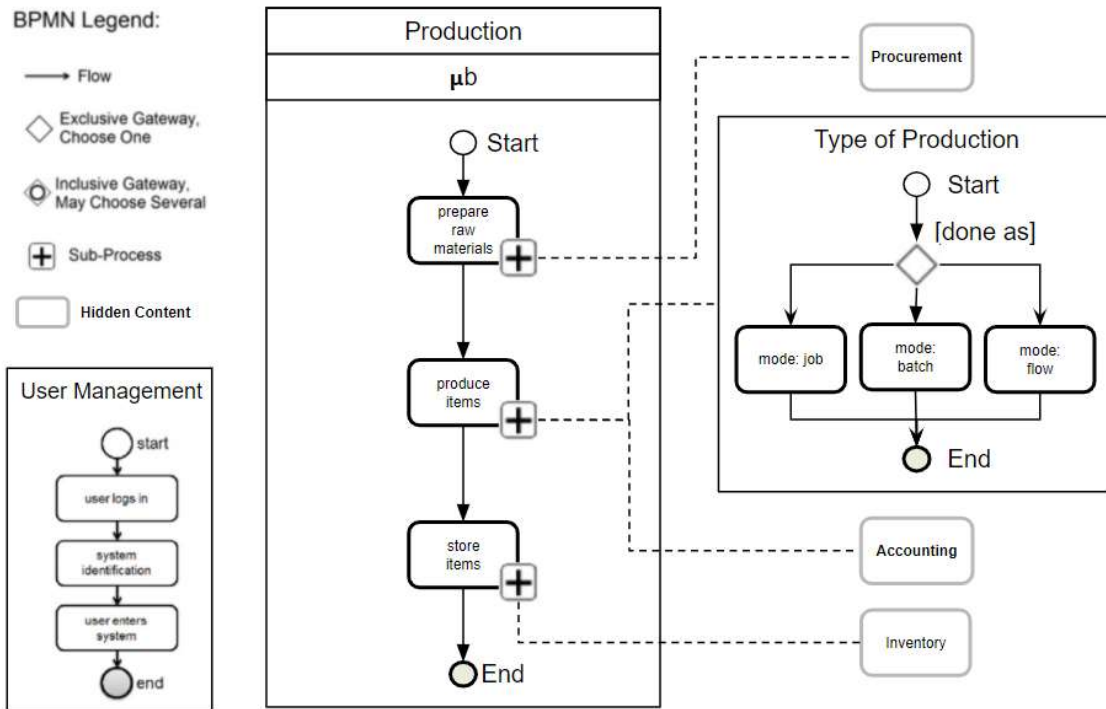


Figure A.4 Production BPMN Example

5. **Name:** Customer Relationship Management CRM

Context: This pattern involves recurring requirements related to the management of customer relationships, mainly acquisition and retention, in a micro-business context. Customer relationship management involves maintaining customer details, managing products bought or services rendered to clients, including technical support and call centers, and marketing of products and services both for existing and new customers.

Keywords: customer relationship management, CRM, customer data, new sales, marketing, technical support, call center

Problem: The problem and requirements are about how to store customer data, how to inform customers of new campaigns, how to manage customer feedback, and how to manage returned items.

Solution: The solutions include in-house servers, cloud servers, traditional ads, social media campaigns, website, mobile messaging, call centers, and ticketing systems.

Customer Relationship Management (CRM) µBRP		
description: this pattern involves recurring requirements related to the management of customer relationships, mainly acquisition and retention, in a micro-business context. Customer relationship management involves maintaining customer details, managing products bought or services rendered to clients, including technical support and call centers, and marketing of products and services both for existing and new customers.		
keywords: customer relationship management, CRM, customer data, new sales, marketing, technical support, call center		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
Where does the µb store customer data?	on a CRM database / linked to sales µBRP / manually / does not store customer data, choose one	
How does the µb store customer data?	in-house server / cloud / other, choose one	
How does the µb track potential customers (in the sales pipeline)?	similar to current customers / separately / other / does not track potential customers, choose one	
How does the µb want to inform customers of promos, new products, etc.?	traditional ads / email / social media / website / mobile / call center / other, select many	
How does the µb want to monitor customer feedback?	traditional feedback forms / email response / through website product ratings / other / no customer feedback, select many	
How does the µb manage returned products?	no return policy / full refund / repair and return / other, choose one	
How does the µb monitor the location of returned products?	linked to logistics µBRP / tracked on location / tracked real-time / not monitored, choose one	
How does the µb monitor costs related to transporting returned products?	linked to accounting µBRP / on a per transit basis (point-to-point charges) / not monitored, choose one	
How does the µb plan to monitor call center/tech support activity?	ticketing / recording / other / no monitoring of activity, select many	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
responsiveness to customer demands		
exactness (preciseness) of CRM data, including costs		
How important is _____ to the customers of the µb?		
the quickness (responsiveness) of the system for orders/returns		
preciseness of information related to the current status of a returned product		
timeliness of the return (of the returned product)		
quality of customer service rendered by the µb		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

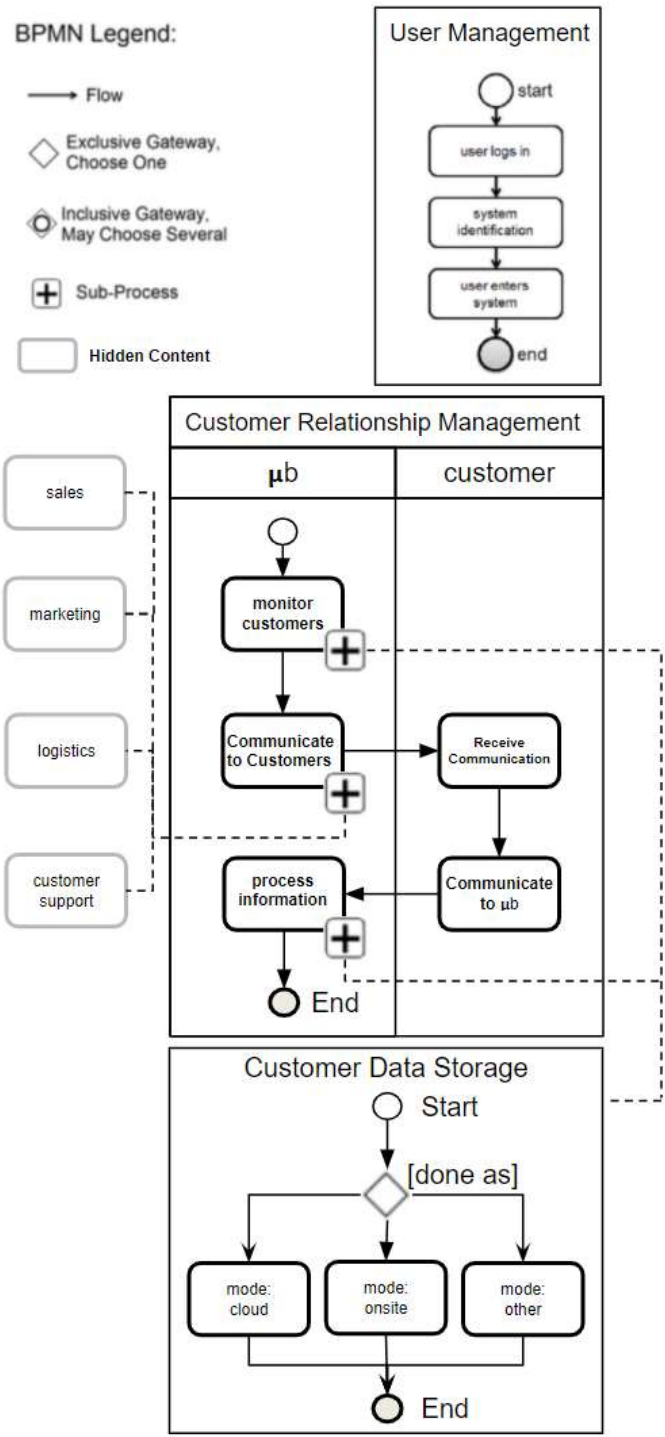


Figure A.5 Customer Relationship Management BPMN Example

6. **Name:** Human Resources

Context: This pattern involves recurring requirements related to the management of human resources in a micro-business context. Human resource management involves hiring ideal candidates for the micro-business, retaining employees through competitive salaries and benefits, monitoring time, attendance, and performance of employees, and the management of labor costs for micro-business.

Keywords: Human Resource Management, Payroll, Benefits, Salaries, Time and Attendance, Hiring, Employee, Labor

Problem: The problems and requirements are about how and what employee information is stored, how to monitor employee performance and attendance, how to manage the hiring pipeline, how to manage payroll and benefits, and how to manage the users of the system.

Solution: The solutions provided are storing employee fields such as name, address, birthday, position, tax number, among others. Employee performance can be managed through success criteria per role or by reaching quotas. Attendance can be checked using biometrics or manually. Some proposed types of users for this system are admins, managers, and employees. There are also proposed solutions on including benefits calculation for the employees.

Human Resources µbRP		
description: this pattern involves recurring requirements related to the management of human resources in a micro-business context. Human resource management involves hiring ideal candidates for the micro-business, retaining employees through competitive salaries and benefits, monitoring time, attendance, and performance of employees, and the management of labor costs for micro-business.		
keywords: Human Resource Management, Payroll, Benefits, Salaries, Time and Attendance, Hiring, Employee, Labor		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
What employee details does the µb want to record and store in the system?	name / position / birthday / tax number / salary / others, select many	
How does the µb plan to monitor employee performance?	criteria per role / quotas / other / no monitoring, select many	
What details for hiring candidates does the µb want to record in the system?	name / position / cv / salary range / others, select many	
How does the µb manage time and attendance for employees?	biometrically / employee logs / manually / no timekeeping, choose one	
What are the employee benefits that the µb wants to administer through the system?	medical / retirement / life insurance / others / none, select many	
What are the types of users on this system?	admin / manager / employee / other, select many	
How does an employee file for leaves?	personally / through the system / no filing needed, choose one	
How does the µb process payslips for employees?	with benefits / taxes / other deductions / no processing, select many	
How does the µb calculate labor costs?	salaries only / with overhead allocation / no calculation, choose one	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
exactness (preciseness) of employee data, including costs		
How important is _____ to the employees of the µb?		
the quickness (responsiveness) of the system for generating payslips and reports		
preciseness of information of an employee, including timekeeping and performance reviews		
relevance of the information available for employee user accounts		
for future employees, relevance of information available for candidates during the hiring process		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

BPMN Legend:

- Flow
- ◇ Exclusive Gateway, Choose One
- ⊕ Inclusive Gateway, May Choose Several
- + Sub-Process
- Hidden Content

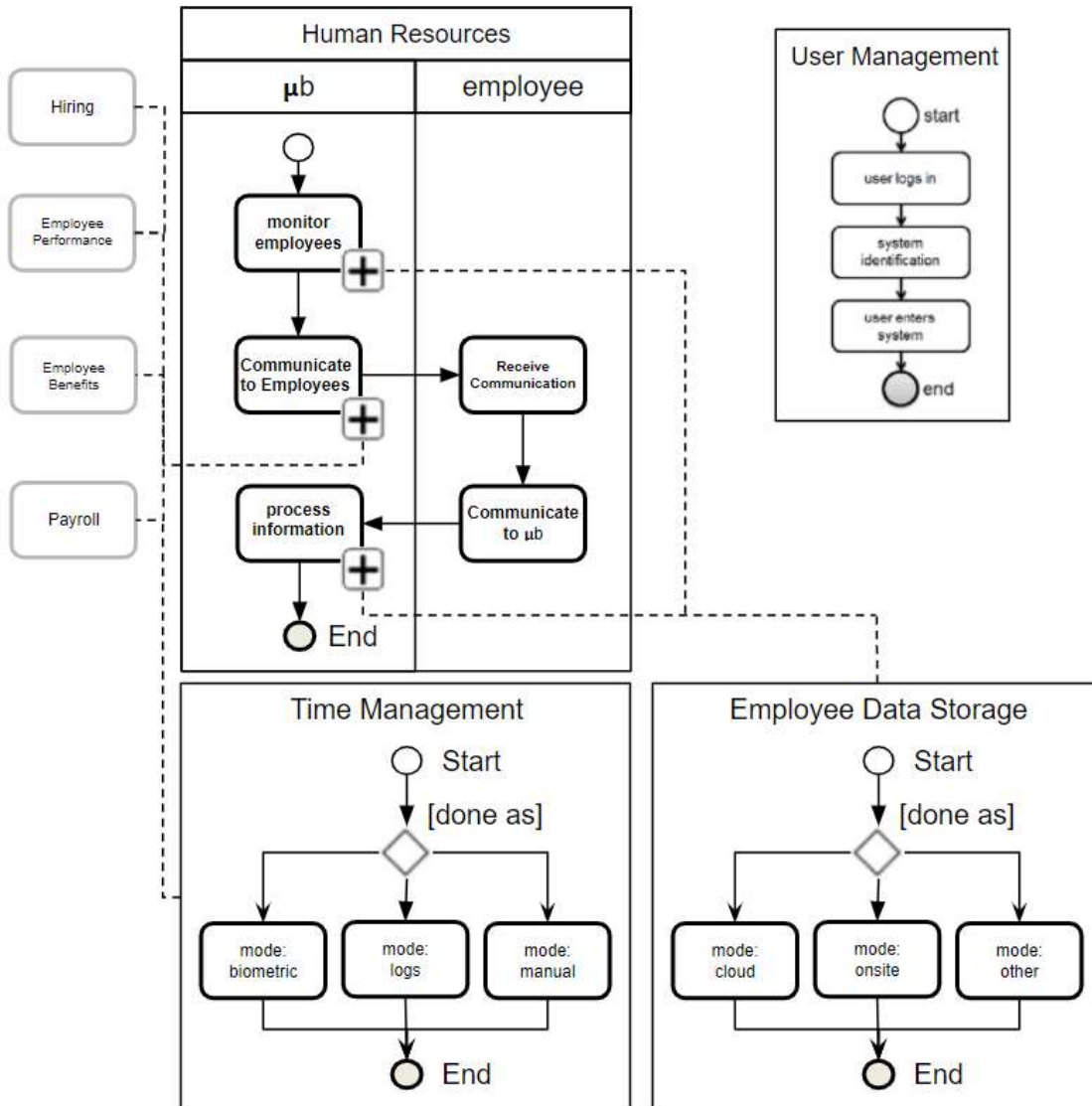


Figure A.6 Human Resources BPMN Example

7. **Name:** Accounting

Context: This pattern involves recurring requirements in the accounting processes of micro-businesses. Accounting involves the input of journal entries onto ledgers which enable the reporting of balance sheets, income statements, statements of cash flow, and other government-required reports. Accounting principles vary from country to country but majority will either follow Generally Agreed Accounting Principles "GAAP" or International Accounting Standards "IAS."

Keywords: keywords: accounting, GAAP, IAS, balance sheet, income statement, cash flow, general ledger, GL, journal entry, compliance

Problem: The problem and requirements are about which accounting standards are being followed by the micro-business, how and when the micro-business files its taxes, the nature of the accounting system and its reports, and the kinds of users of the system.

Solution: The solutions involve a system complying with Generally Agreed Accounting Principles "GAAP" or International Accounting Standards "IAS", maintaining and/or reporting monthly, quarterly, semi-annually, and annually, having a system oriented towards financial and/or managerial accounting including the nature of reporting, and the ability to maintain a single or multiple currencies.

Accounting µBRP		
description: this pattern involves recurring requirements in the accounting processes of micro-businesses. Accounting involves the input of journal entries onto ledgers which enable the reporting of balance sheets, income statements, statements of cash flow, and other government-required reports. Accounting principles vary from country to country but majority will either follow Generally Agreed Accounting Principles "GAAP" or International Accounting Standards "IAS."		
keywords: accounting, GAAP, IAS, balance sheet, income statement, cash flow, general ledger, GL, journal entry, compliance		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
What accounting principle does the µb follow?	GAAP (US) / IAS / other, select one	
How does the µb file its government-related taxes?	external accountant / directly / other, select one	
What are the user types for the accounting system?	admin / manager / auditor / employee / other, select many	
What accounting time period does the µb have to maintain in records?	no need to maintain records / up to 3 mos. / 6 mos. / 1 year / 2 years... / 5 years or more, select one	
Does the accounting system need to support multiple currencies?	yes / no, select one	
What is the nature of the accounting system?	financial / managerial / both, select one	
How many ledgers (a.k.a. "GLs") does the µb need in the system?	one / two / three... ..N, specify a number	
What are the other systems linked to the accounting system?	sales / inventory / online / logistics / production / crm / hr / project management / management, select many	
What are the kinds of reports that the accounting system can generate?	balance sheet / income statement / cash flow / list of journal entries / other (see management µBRP), select many	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
exactness (preciseness) of accounting data		
adaptability of software to several currencies and fluctuating exchange rates		
How important is _____ to the customers and third parties of the µb?		
the quickness (responsiveness) of the system for generating bills, invoices, and reports		
exactness (preciseness) of the accounting information		
relevance of the information available for customers and third parties (including government) of the µb		
for governments, compliance of the system to accounting standards		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

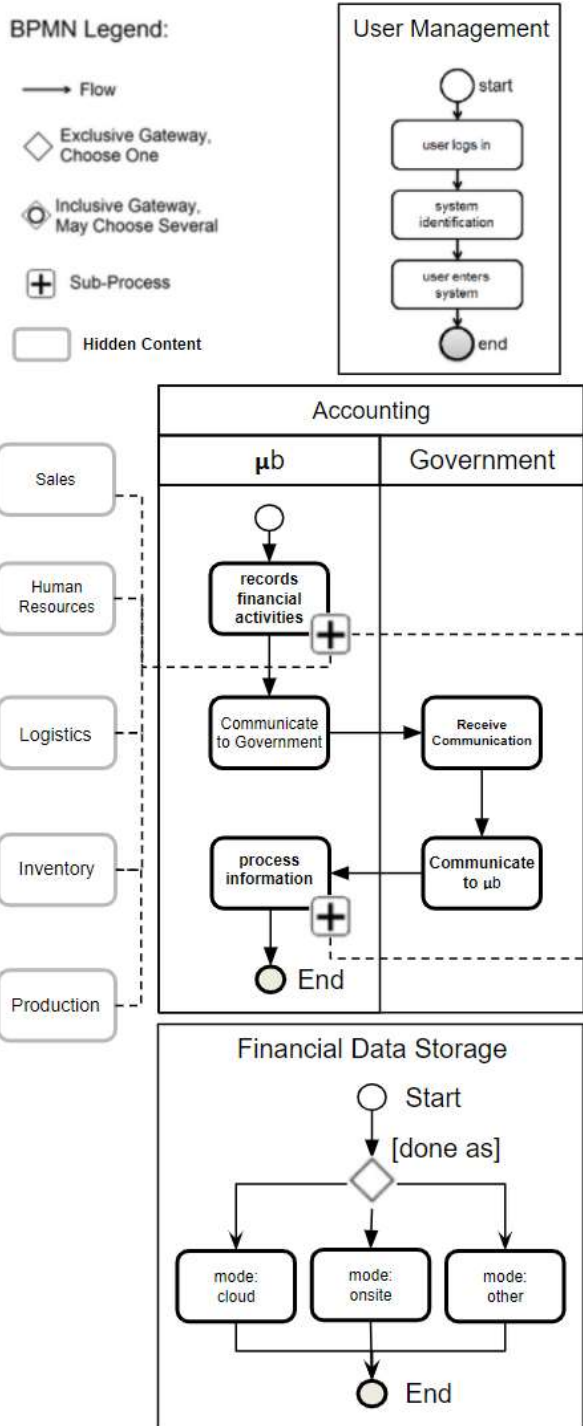


Figure A.7 Accounting BPMN Example

8. **Name:** Management

Context: This pattern involves recurring requirements in management reporting for micro-businesses. Management reporting involves supporting decision makers with relevant information on profitability, sales, costs, forecasts, and employee information.

Keywords: Management, Reporting, Profitability, Sales, Forecast, Costs, Employee Roster

Problem: The problems and requirements are about who makes the decisions and drives the success of the micro-business, who the micro-business has to provide reports to, and where and how these reports are stored and recorded.

Solution: The solution involves identifying stakeholders such as proprietors, partners, investors, executives, management, and staff. The reports could be stored on local servers or on the cloud and can be provided to tax authorities, local government, internal employees, or the public. The cadence of reporting can be done monthly, quarterly, semiannually, or annually.

Management µbRP		
description: this pattern involves recurring requirements in management reporting for micro-businesses. Management reporting involves supporting decision makers with relevant information on profitability, sales, costs, forecasts, and employee information.		
keywords: Management, Reporting, Profitability, Sales, Forecast, Costs, Employee Roster		
Functional Requirements Section		
The question-answer structure allows the developer to clarify the hard goals or FRs of the µb owner		
Question	Options	µb Choices
Who are making management decisions in the µb?	sole proprietor / partners / board members / c-level executives / top management / middle management / staff / other, select many	
To whom does management need to provide reports to?	third parties (B2B) / government agencies (IRS) / employees / aspiring employees / the general public / others / none, select many	
In what format(s) do the reports have to be in?	Microsoft (DOC, XLS, PPT, etc...) / Adobe (PDF) / Google Docs / other, select many	
How long does the µb have to maintain reports on a server?	no need to maintain reports on a server / 3 mos. / 6 mos. / 1 year / 2 years / N years or more, select one	
What are the other linked systems used for generating management reports?	sales / inventory / online / logistics / production / crm / hr / project management / accounting, select many	
What are the reports generated by this system?	reports generated by linked systems (see previous question) / profitability per product or service / cost centers and drivers / forecasts / other management reports, select many	
What are the types of users needed in this system?	admin / manager / third party / employee / stockholder / other, select many	
Non Functional Requirements Section		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
the cost of the software project		
the speed of the delivery of the software project		
the speed (responsiveness) of the software system		
the user-friendliness of the software system		
ease of maintaining the software system		
the security of the software system		
portability of the software		
exactness (preciseness) of management reports		
relevance of the management reports for decision-making		
How important is _____ to the customers and third parties of the µb?		
the quickness (responsiveness) of the system for generating management reports		
exactness (preciseness) of the management reports		
relevance of the information available for customers and third parties (including government) of the µb		
integrity of management reports in comparison with government-related reports		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
initial profitability of the software project		
long-term profitability of the software project		
the speed of the delivery of the software project		
ease of delivering the software project		
ease of maintaining the software after the project		

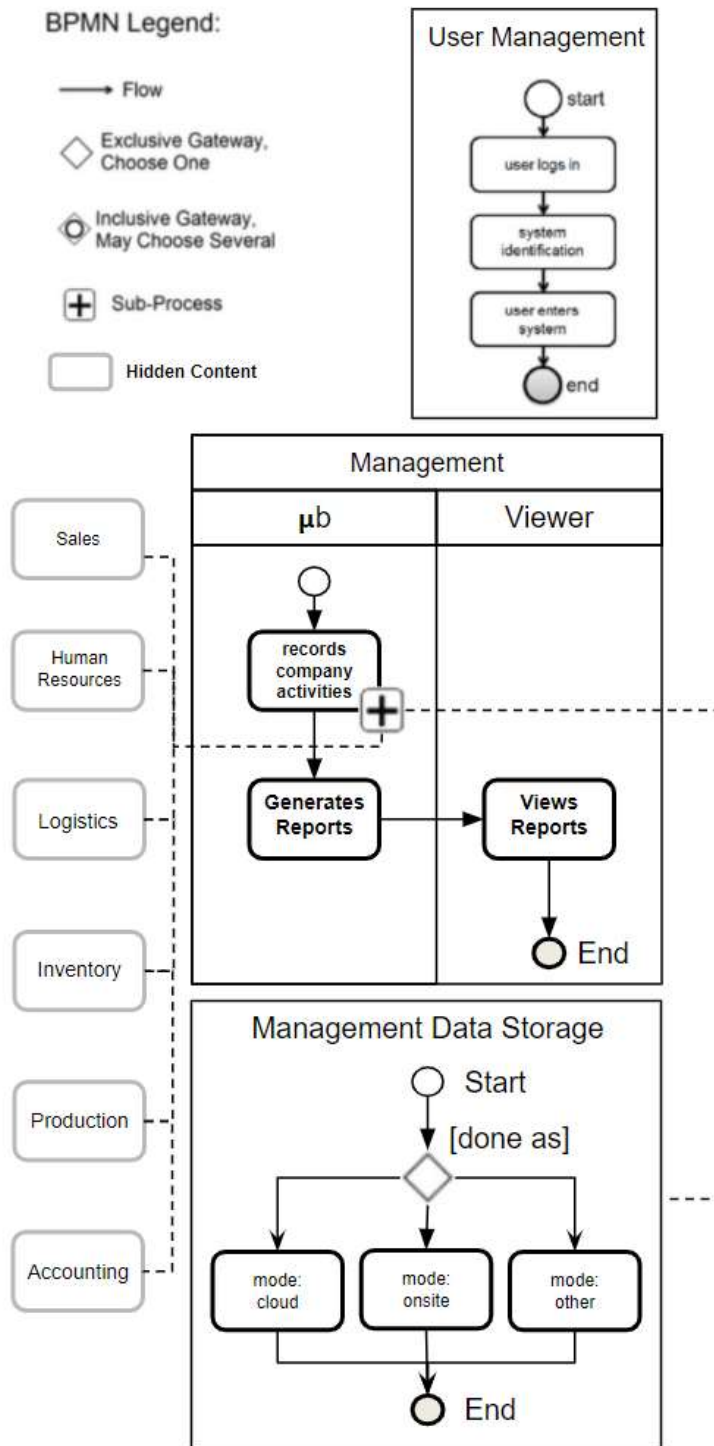


Figure A.8 Management BPMN Example

9. **Name:** Restaurant

Context: This pattern involves a restaurant micro-business sourcing ingredients, making food, and then providing the food for consumption for the customers

Keywords: ingredients, food, restaurant, delivery

Problem: The problem and the requirements are about how the micro-business sources its ingredients, how the customers will place their orders to the restaurant micro-business, and how the micro-business will deliver the food to the customers for their consumption.

Solution: The solutions provided involve: sourcing from the market, having ingredients delivered by a third party, customers placing orders with an in-house menu, phone, website, or mobile/app, having the customers consume the food in-house, have it delivered to a destination, or have the food picked up at the restaurant.

Restaurant µBRP		
description: µb sources ingredients, µb produces food for customers to eat, and then customer pays.		
keywords: ingredients, food, restaurant, delivery		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb gather the ingredients for the food to be served?	buy at market / have ingredients delivered by third party / other select as many that apply	
(b) How can customers order food at the restaurant µb?	in-house menu / phone / website / mobile app / other select as many that apply	
(c) How do customers eat the food from the restaurant?	in-house / take away / delivery select as many that apply	
(d) How can customers pay the restaurant µb?	credit / debit / Paypal / other, select as many that apply	
(e) What roles do the µb-side users have on the system?	admin / manager / user / other select as many that apply	
(f) Is the restaurant µb linked to a customer relationship management system? (µb inventory pattern can be found in the Appendix)	yes / no	
(g) Is the restaurant µb linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	
(h) Is the restaurant µb linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	
(i) Is the restaurant µb linked to a logistics management system?	yes / no	
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
system speed (responsiveness)		
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
quickness (responsiveness) of the food service		
quickness (responsiveness) of customer payment		
quickness (responsiveness) of the software system (online restaurant site/app)		
user-friendliness of the software system, customer side (online restaurant site/app)		
security of the software system (transactions made on online restaurant site/app)		
availability of the food on the menu (availability of ingredients)		
delivery time of the food (fast food delivery)		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		

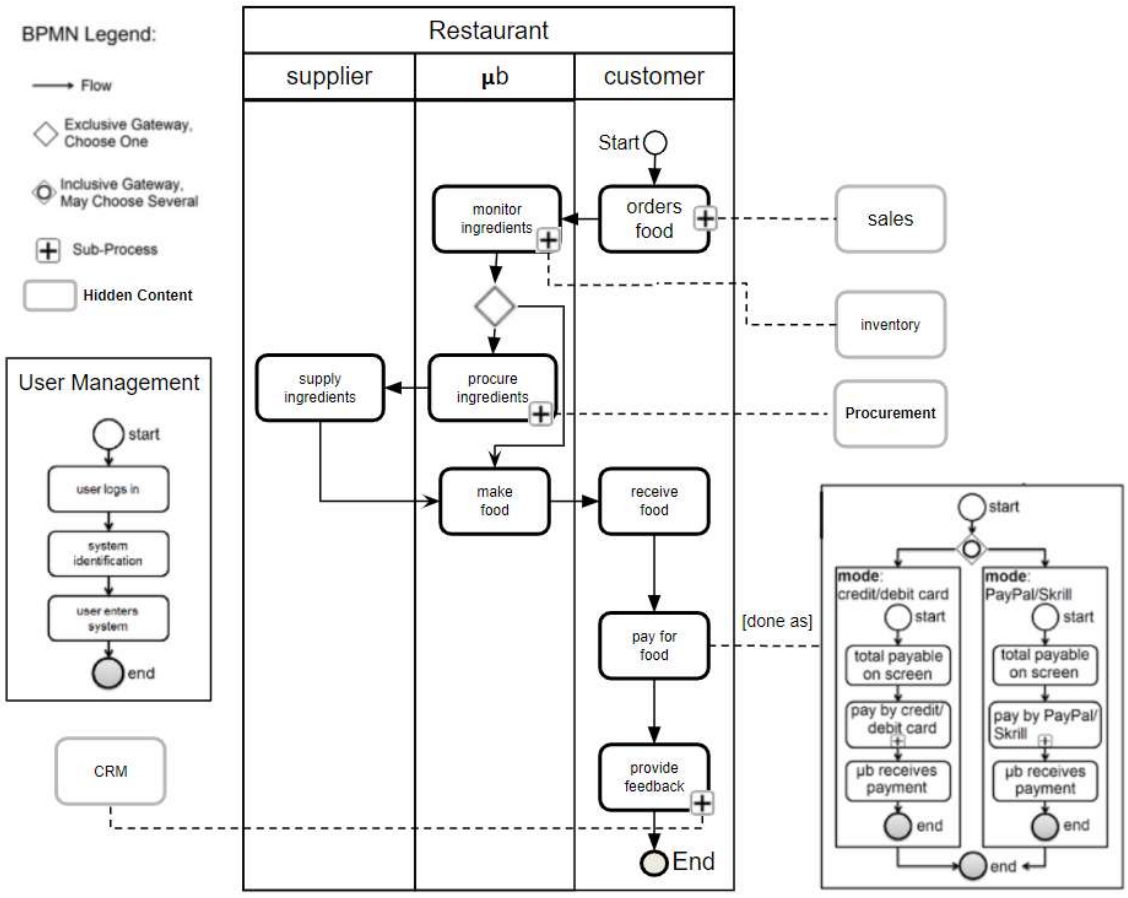


Figure A.9 Restaurant BPMN Example

10. **Name:** Online Retail Shop

Context: This pattern involves the micro-business selling items online, customer pays online, and then micro-business ships item to the customer.

Keywords: online shopping, online payment, online activity monitoring

Problem: The problem and requirements are about how customers are looking for the items they want to buy, how these items are presented online to the customers, how payment is made by the customer, how customer reviews are made, how and where the data from the online activity will be stored, and also how to track the delivery of the items to customers.

Solution: The solutions are search engines, filters, catalogs, presentation with images, video, text, types of customer reviews, payment methods such as credit, debit, bank transfers, e-wallets, saving customer data on local servers or in the cloud, and having a delivery tracking system.

online retail shop µBRP		
description: µb sells items online, customer pays online, and then µb ships item to the customer.		
keywords: online shopping, online payment, online activity monitoring		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb customers find the product they want at the online shop?	search engine / filters / product catalog / offers / other, select as many that apply	
(b) How are the products of the µb presented on the online shop?	image / text / video / review (see (c)) select as many that apply	
(c) Can an µb customer leave product reviews on the online shop?	yes, optional (see account in (f)) / no	
(d) How does the online shop accept payments? (credit card pattern can be found at (Macasaet et al., 2011))	credit / debit / PayPal / Skrill / other select as many that apply	
(e) Will the online shop offer a tracking service for product deliveries?	yes (see (l)) / no	
(f) How will the online shop collect visitor/shopper data?	server-based / client-based / customer account-based / other / none, select as many that apply	
(g) Are admin privileges needed by µb-side users?	yes / no	
(h) Does the online shop have a physical retail store location?	yes (see (l)) / no	
(i) What other µb information is available on the online shop?	company info / physical store locations / FAQs / other, select as many that apply	
(j) Is the online shop linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	
(k) Is the online shop linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	
(l) Is the online shop linked to a logistics management system?	yes / no	
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
system speed (responsiveness)		
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
quickness (reponsiveness) of the software system (online shop)		
user-friendliness of the software system, customer side (online shop)		
security of the software system (transactions made on online shop)		
availability of the product in the inventory (inventory stock levels)		
delivery time of the product (fast product delivery)		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		

Sample Choices for Online Retail Shop

online retail shop µBRP		
description: µb sells items online, customer pays online, and then µb ships item to the customer.		
keywords: online shopping, online payment, online activity monitoring		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb customers find the product they want at the online shop?	search engine / filters / product catalog / offers / other, select as many that apply	search engine, filters, catalog
(b) How are the products of the µb presented on the online shop?	image / text / video / review (see (c)) select as many that apply	image, text, review
(c) Can an µb customer leave product reviews on the online shop?	yes, optional (see account in (f)) / no	yes
(d) How does the online shop accept payments? (credit card pattern can be found at (Macasaet et al., 2011))	credit / debit / PayPal / Skrill / other select as many that apply	credit, debit, PayPal, Skrill
(e) Will the online shop offer a tracking service for product deliveries?	yes (see (l)) / no	yes
(f) How will the online shop collect visitor/shopper data?	server-based / client-based / customer account-based / other / none, select as many that apply	server and customer-based
(g) Are admin privileges needed by µb-side users?	yes / no	yes
(h) Does the online shop have a physical retail store location?	yes (see (i)) / no	no
(i) What other µb information is available on the online shop?	company info / physical store locations / FAQs / other, select as many that apply	company info, FAQ
(j) Is the online shop linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	yes
(k) Is the online shop linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	yes
(l) Is the online shop linked to a logistics management system?	yes / no	yes
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
(m) system speed (responsiveness)		3rd
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		5th
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
(m) quickness (reponsiveness) of the software system (online shop)		2nd
user-friendliness of the software system, customer side (online shop)		
security of the software system (transactions made on online shop)		4th
availability of the product in the inventory (inventory stock levels)		
(n) delivery time of the product (fast product delivery)		1st
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		1st
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		2nd

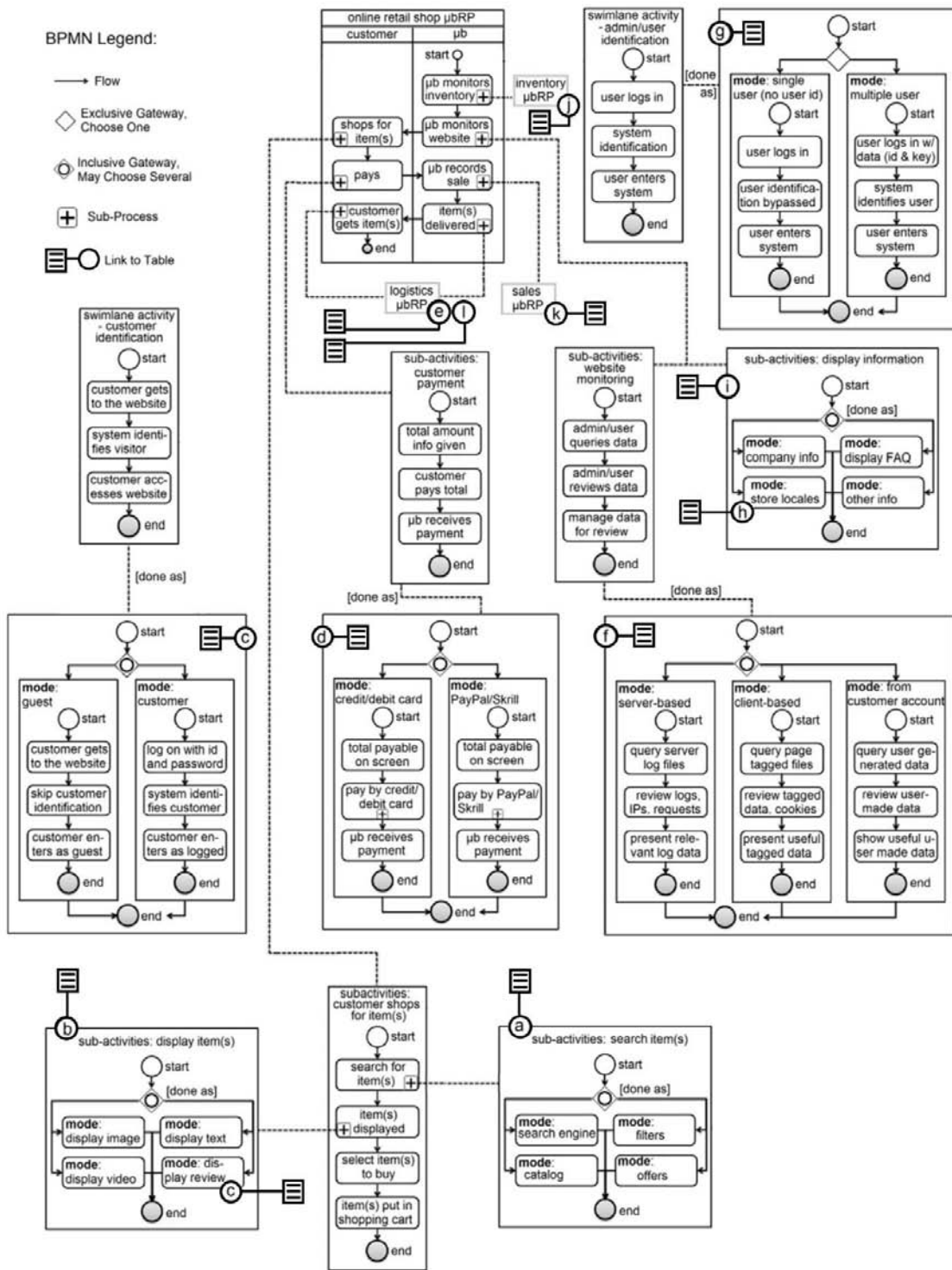


Figure A.10 Model of the modes or “options” of the Online Retail Shop

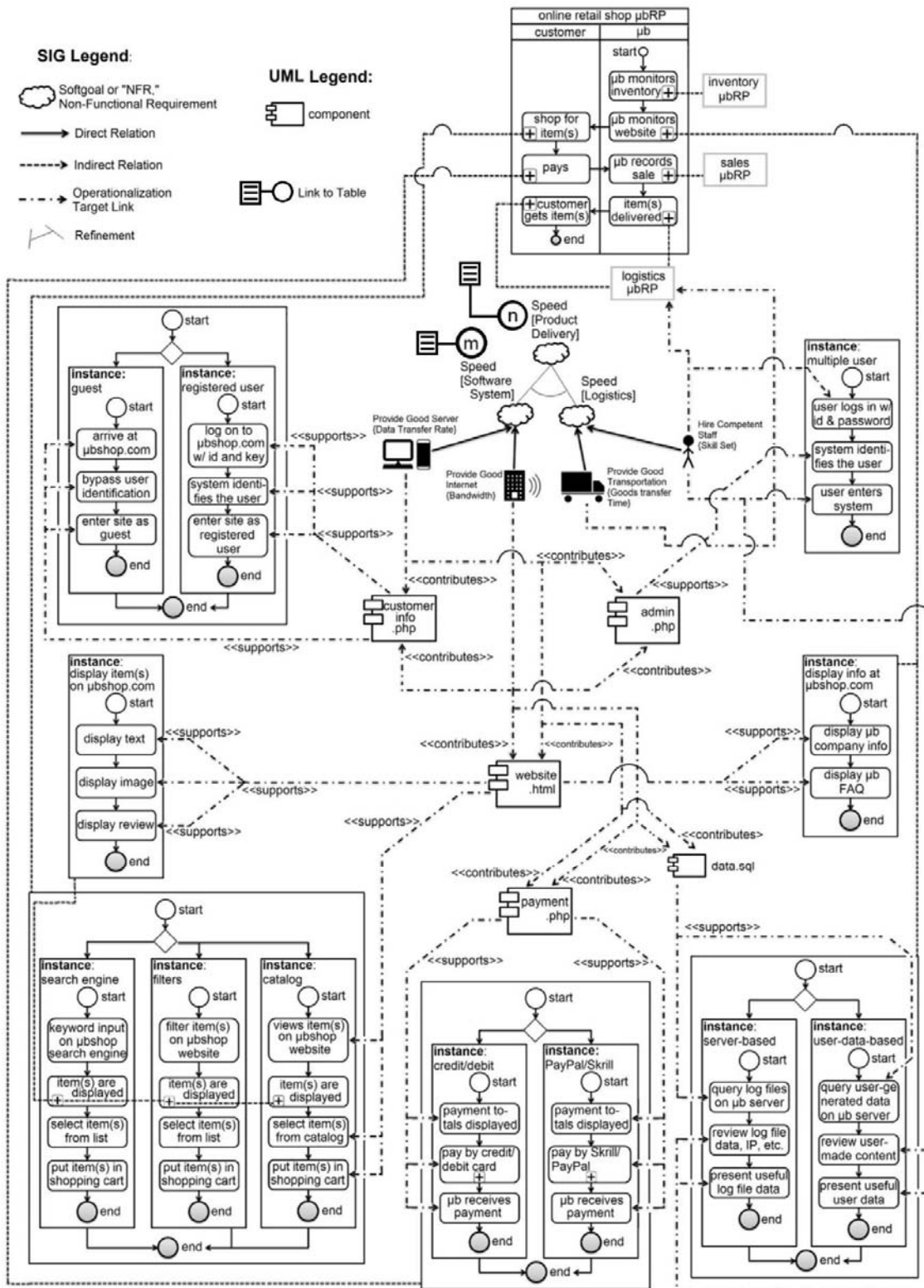


Figure A.11 Model of the Sample Online Store μRP instantiation with “choices” and “priorities”

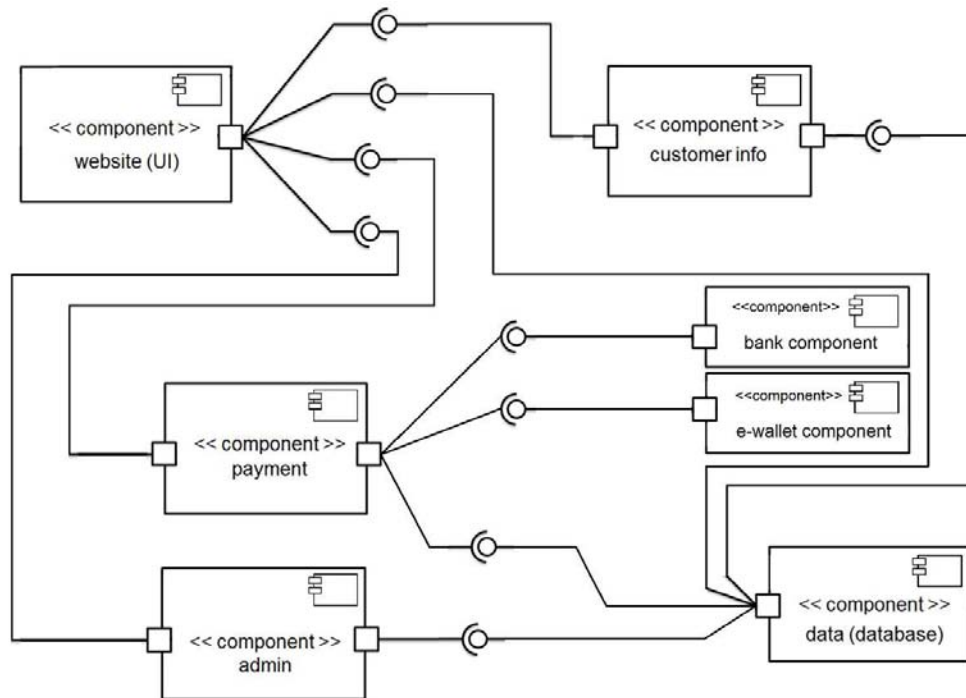


Figure A.12 Model of a Component Diagram for the Online Retail Shop Sample

Appendix B

Languages and Notations in Practice

1. Business Process Modeling Notation “BPMN”

1.1 BPMN Concepts

The sequence flow defines the execution order of the activities in its supposed chronological order. A sequence flow is considered as the default flow if all other conditions for its execution are false. In a micro-business, the default flow from one activity to another could be: a cashier gets the change from the cash register and by default, he or she hands over the cash to the customer.

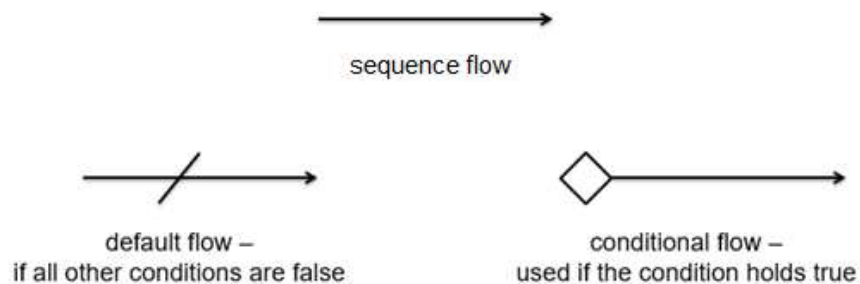


Figure B.1 BPMN Sequence Flows



Figure B.2 BPMN Example of a Default Flow

Another type of flow is a conditional flow. A conditional flow executes if the condition for its execution holds true. In a micro-business, a conditional flow could be: if the customer pays in cash more than what is billed then the customer will receive change and if not, the cashier will simply accept the payment, put it in the cash register, and then simply thank the customer for coming. Hence, the condition to give back change is dependent on the amount of cash paid

by the customer for his or her bill. The flows, both default and sequential, are shown below in BPMN.

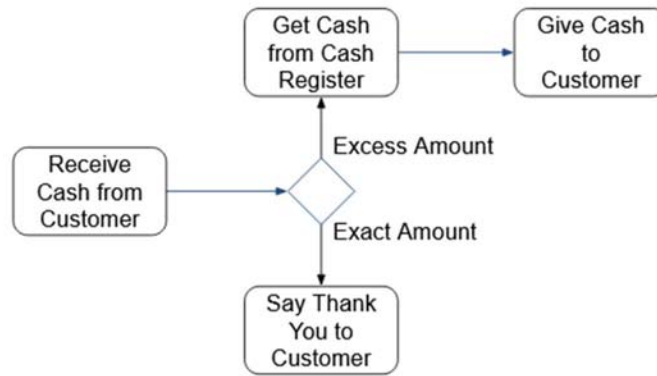


Figure B.3 BPMN Example of a Conditional Flow

The start event and an end event are where flows begin and end.



Figure B.4 BPMN Start and End Events

In a micro-business, the start event could be the moment when a customer walks into a brick-and-mortar store while the end event could be the moment when a customer walks out of the store.



Figure B.5 BPMN Example of Start and End Events

The activity task is a unit of work and is the job to be performed. When an activity task is marked with a + symbol, it indicates the existence of a sub-process, an activity that can be refined into more granular or specific jobs.

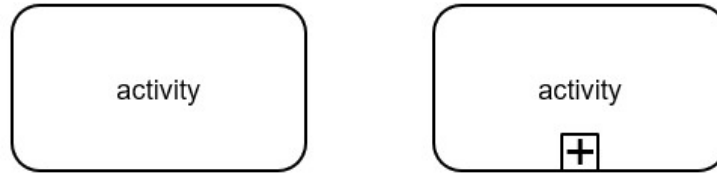


Figure B.6 BPMN Activity Task and Sub-Process

In a micro-business, an activity could be arranging grocery items in the store. A sub-process could be arranging the items in the milk section depending on the requests and contracts with each of the suppliers.

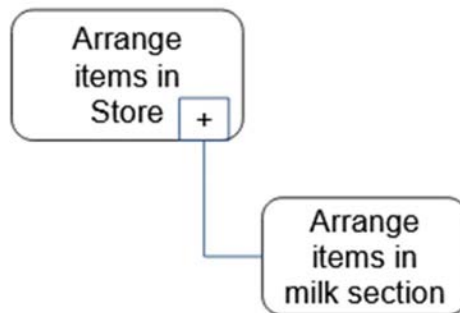


Figure B.7 BPMN Example of Activity Task and Sub-Process

A data object represents information flowing through a process such as a document, email, or letter.



Figure B.8 BPMN Data Object

In a micro-business, this could be the receipt that is provided to a customer after he or she pays for his or her merchandise at the store.



Figure B.9 BPMN Example of a Data Object

A data store is a place where the process can read or write data such as a database or a filing cabinet. It exists even outside the whole process.



Figure B.10 BPMN data store

In a micro-business, when a customer pays by credit card, the credit card machine usually produces a copy for the merchant or the micro-business and an optional copy for the customer if he or she requests for their copy. If a micro-business decides to physically store these merchant copies, they can be placed in a filing cabinet or another physical repository.



Figure B.11 BPMN Example of a data store

Pools and lanes represent responsibilities for activities in a process. A pool/lane may be an organization, a role, or a system. Lanes further subdivide pools or other lanes hierarchically.

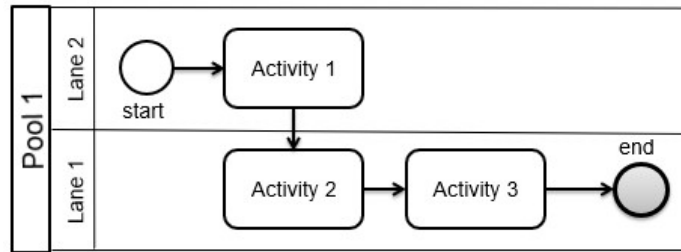


Figure B.12 BPMN Pools and Lanes

In a micro-business, a lane could be a role such as the cashier who is handling all the payments of the customers at the brick-and-mortar store. Two roles such as the cashier and the customer could form the pool that groups all on-site roles inside the brick-and-mortar store.

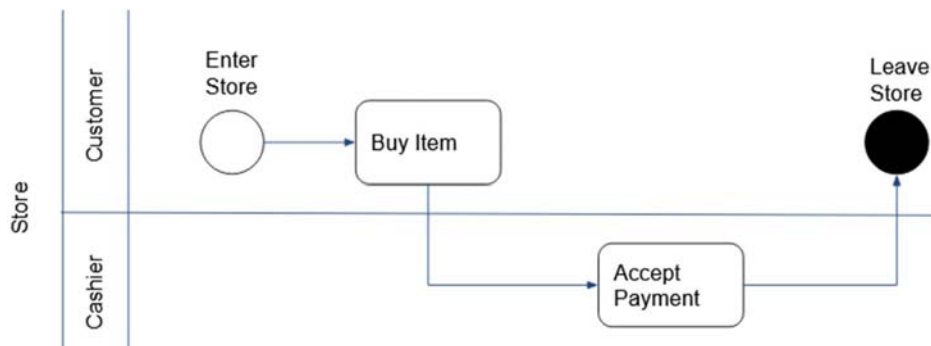


Figure B.13 Example of BPMN Example of Pools and Lanes

The exclusive gateway routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

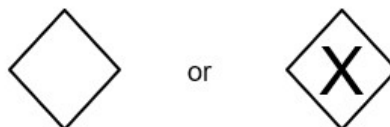


Figure B.14 BPMN Exclusive Gateway

In a micro-business, this could be all the different payment types such as by cash, credit card, or coupon all ending in the same way which is when the cashier provides a receipt to the customer, regardless of their payment method.

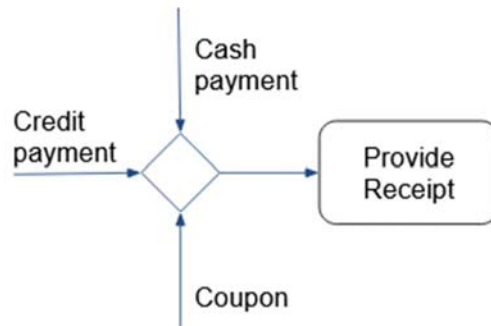


Figure B.15 BPMN Example of an Exclusive Gateway

The inclusive gateway activates one or more outgoing flows. When merging, all active incoming branches must complete before proceeding.



Figure B.16 BPMN Inclusive Gateway

In a micro-business food stall, this could be the moment after the customer pays and then simultaneously, the cashier hands out the receipt to the customer and one of the kitchen staff starts preparing the food for the customer.

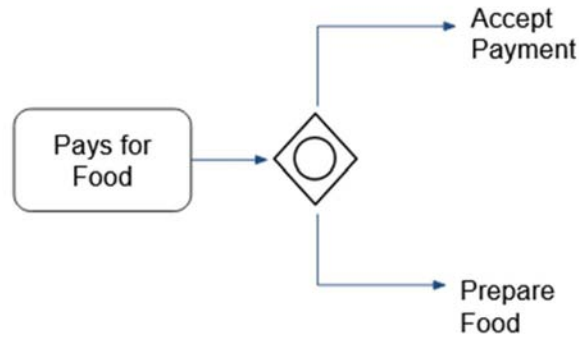


Figure B.17 BPMN Example of an Inclusive Gateway

A parallel gateway splits the sequence flow and activates all outgoing branches simultaneously. When merging, parallel branches wait for all incoming branches to complete before triggering the outgoing flow.



Figure B.18 BPMN Parallel Gateway

In an online retail store micro-business, this could be the moment when a customer makes an online order and prefers to pay by bank transfer. Before any merchandise is sent out, the retailer must wait for the bank transfer to complete. Upon completion of the bank transfer, the retailer can then ship the merchandise to the customer.

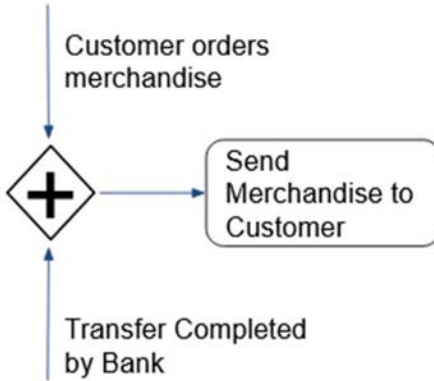


Figure B.19 BPMN Example of a Parallel Gateway

An event-based gateway is always followed by catching events or receiving tasks. Sequence flow is routed to the subsequent event/task which happens first.



Figure B.20 BPMN Event-Based Gateway

In a micro-business, this event could be a customer who is complaining about his or her food in a restaurant. When this happens, the subsequent activity is for a responsible role to resolve the situation or to escalate the incident to his or her manager.



Figure B.21 BPMN Example of an Event-Based Gateway

A complex gateway has branching behavior that is not captured by the other gateways.



Figure B.22 BPMN Complex Gateway

In a micro-business, this could be a very specific case particular to the micro-business. For instance, if a fast food restaurant gets a request from a customer to prepare his or her food in a very specific way then depending on this request, certain activities will or will not take place. There are some customers who do not prefer to have pickles on their burgers, have the burger cooked in between medium rare and well done, and have slightly cooked onions. There are just so many permutations on how you could prepare a burger in a fast food restaurant micro-business.

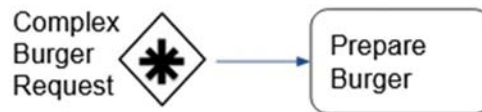


Figure B.23 BPMN Example of a Complex Gateway

1.2 BPMN Micro-business Example

One example that could be used to apply the BPMN concepts in the previous subsection is to illustrate the ordering process by customers at a fast food restaurant. First, the customer enters the fast food restaurant and approaches a touch-screen panel where he or she can place an order. The customer selects the food and drink items they would want and then they are given the option to pay by credit card at the touch screen or to pay by cash at the counter.

If the customer pays by credit card then he or she is given a confirmation slip at the touch-screen panel and can wait for his or her order at the food disbursement counter. If the customer decides to pay by cash then he or she must proceed to the cashier and pay there physically.

When the payment is made, he or she can now wait for his or her order at the food disbursement counter. When the food preparation staff of the fast food restaurant completes the preparation of food, they hand the food out at the food disbursement counter to the customers. At this point, the customers can select their seat at the fast food restaurant and eventually enjoy their meal.

The illustration of the ordering process by customers at a fast food restaurant is presented in the BPMN diagram below, using the concepts discussed in the previous subsection.

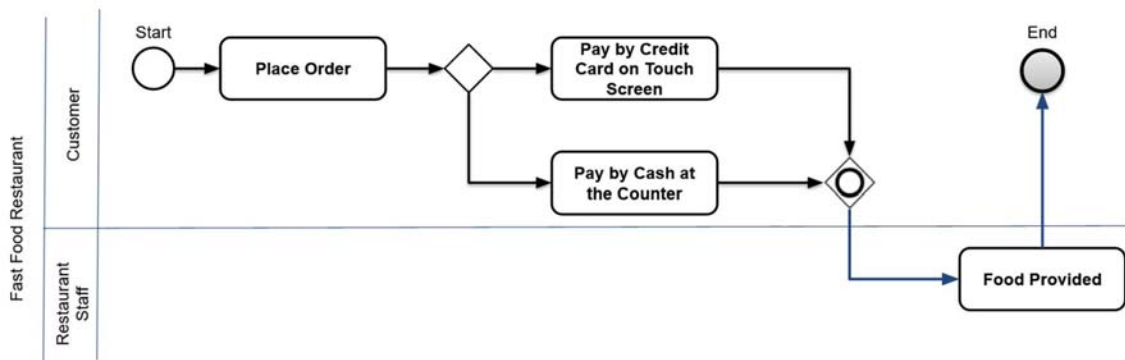


Figure B.24 BPMN Diagram of the ordering process of customers at a fast food restaurant

2. Softgoal Interdependency Graphs “SIGs”

2.1 SIGs Concepts

NFRs (or softgoals) are represented using a cloud, accompanied by a label which indicates the NFR and what it pertains to in brackets “[].” Using SIGs, an NFR or softgoal is considered as a goal which cannot be strictly satisfied but “satisfied,” meaning it is satisfied sufficiently,



Figure B.25 NFR or softgoal representation

In the micro-business domain, an NFR or softgoal could be the timeliness that a cashier completes a transaction with a customer in a brick-and-mortar retail shop. How timely should the cashier be in order to satisfy a customer?



Figure B.26 NFR or softgoal representation example

When decomposing or refining softgoals, “and” and “or” solid line connectors are used.

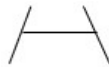


Figure B.27 AND Refining of Softgoals



Figure B.28 OR Refining of Softgoals

An example of refining softgoals in a micro-business could be refining the timeliness of a transaction between the cashier and the customer into the responsiveness of the cash register machine and the agility of the customer to make the payment.

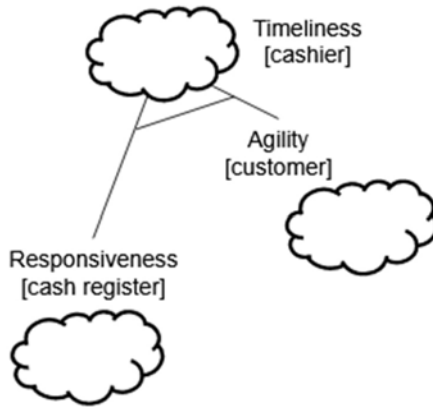


Figure B.29 Refining of Softgoals Example

Interdependency arrows, indicating explicit or implicit relationships are used to connect softgoals. Using SIGs, an explicit or direct relationship of softgoals is represented with a directional arrow.



Figure B.30 SIG Direct or Explicit Softgoal Relationship

In a micro-business, an example of a direct relationship to a softgoal is that employing a qualified cashier directly affects the responsiveness of the cash register machine. Without a qualified cashier, the responsiveness of the cash register can not even be assessed. Employing a qualified cashier is an operationalizing method which will be discussed in more detail later.

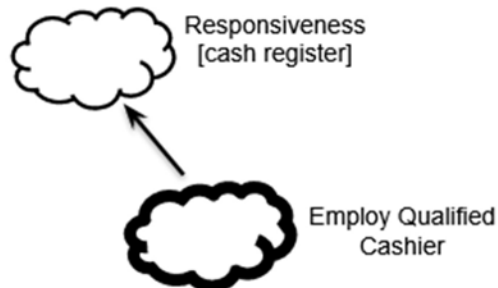


Figure B.31 SIG Direct or Explicit Softgoal Relationship Example

Using SIGS, an indirect or implicit relationship of softgoals is represented with a broken directional arrow.



Figure B.32 SIG Indirect or Implicit Softgoal Relationship

In a micro-business, an indirect relationship to a softgoal could be that the responsiveness of the cash register machine may indirectly depend on various distractions at the store such as noise levels, lighting, presence of children, among other things.



Figure B.33 SIG Indirect or Implicit Softgoal Relationship Example

In SIGs, a + sign above an interdependency arrow is used for a positive contribution that helps satisfy a softgoal but does not satisfy it by itself.

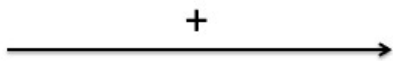


Figure B.34 SIG Positive Direct Dependency Softgoal

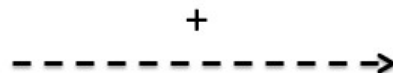


Figure B.35 SIG Positive Indirect Dependency Softgoal

In a micro-business, an example could be that the qualified cashier directly and positively affects the responsiveness of the cash register but does not entirely make the cash register as responsive as possible. There are also other factors that directly affect the responsiveness of the cashier such as the firmware of the cash register.



Figure B.36 SIG Positive Direct Dependency Softgoal Example

In SIGs, a ++ sign above an arrow is used to describe a strong positive contribution. As such, can satisfice a softgoal by itself.

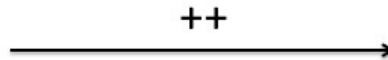


Figure B.37 SIG Strong Positive Direct Dependency Softgoal

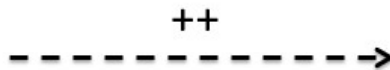


Figure B.38 SIG Strong Positive Indirect Dependency Softgoal

In a micro-business, the security of a transaction at a cash register could be satisfied by a biometric system incorporated into the cash register, only allowing approved cashiers to execute transactions.

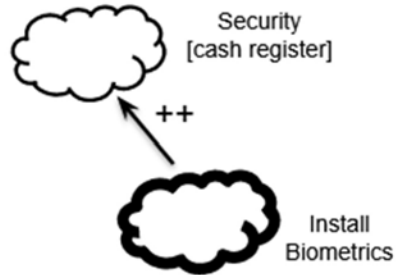


Figure B.39 SIG Strong Positive Direct Dependency Softgoal Example

In SIGs, a - sign above an arrow is used for a negative contribution that hampers the achievement of a softgoal but does not, by itself, prevent satisfying the softgoal.



Figure B.40 SIG Negative Direct Dependency Softgoal



Figure B.41 SIG Negative Indirect Dependency Softgoal

In a micro-business, the noise level of children may contribute to distractions to the cashier when making transactions at the cash register. These distractions could indirectly and negatively affect the responsiveness of the cash register in completing a transaction for the customer.

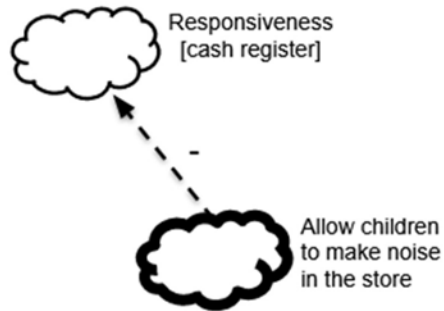


Figure B.42 SIG Negative Indirect Dependency Softgoal Example

In SIGS, a -- sign above an arrow is used for a strong negative contribution and by itself, can hamper the achievement of the softgoal.



Figure B.43 SIG Strong Negative Direct Dependency Softgoal



Figure B.44 SIG Strong Negative Indirect Dependency Softgoal

Some micro-businesses decide to work and integrate with local banks only. Foreigners at rural stores usually have credit or debit cards that some local cash registers can not recognize. This means that an unacceptable payment method from a customer could entirely hamper the responsiveness of the cash register to complete a transaction.

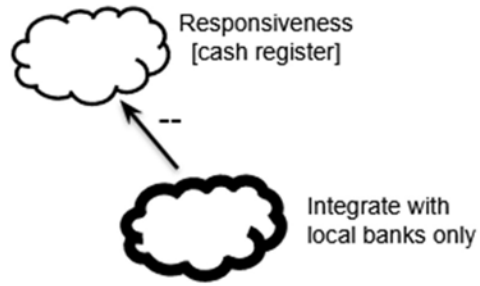


Figure B.45 SIG Strong Negative Direct Dependency Softgoal Example

When softgoals are refined, there comes a point where they can be operationalized. Operationalizing methods are measurable (or estimate-able) elements which satisfy refined softgoals. In SIGs, an operationalizing method is represented as a cloud in **bold**.



Figure B.46 SIG Operationalizing Method

In a micro-business, an operationalizing method could be a router or a network device connected to a cash register which contributes to the responsiveness of the cash register. A network device with a reliable connection allows the cash register to operate responsively.



Figure B.47 SIG Operationalizing Method Example

From an operationalizing method, an operationalization target link is used and directed towards a target system which is a focal point where NFRs and FRs meet. In SIGs, an operationalization target link is represented by a dash-dot-dash arrow.



Figure B.48 SIG Operationalization Target Link

In SIGs, a target system is represented with a rectangle.

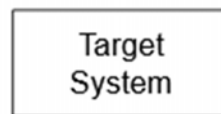


Figure B.49 SIG Target System

In a micro-business, an operationalization target link can come from a router and be directed towards the router's firmware, a target system.



Figure B.50 SIG Operationalization Target Link and Target System Example

From the target system, a design decision link is used and is directed towards one or more FRs which it satisfies (note: satisfice \neq satisfy). In SIGS, a decision link is represented with a directional arrow with a solid line. This is differentiated from the direct dependency arrow by looking at where it originates, a target system, and where it points to, a functional requirement.

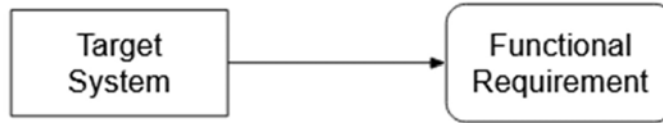


Figure B.51 SIG Design Decision Link

In a micro-business, the firmware on the network device could be able to completely satisfy a functional requirement such as to have a minimum of 10 megabytes per second of both upload and download speed 95% of the time.



Figure B.52 SIG Design Decision Link Example

An overview of SIGs and the NFR Framework, including the flow from an operationalizing method to a Functional Requirement is shown in Figure B.53.

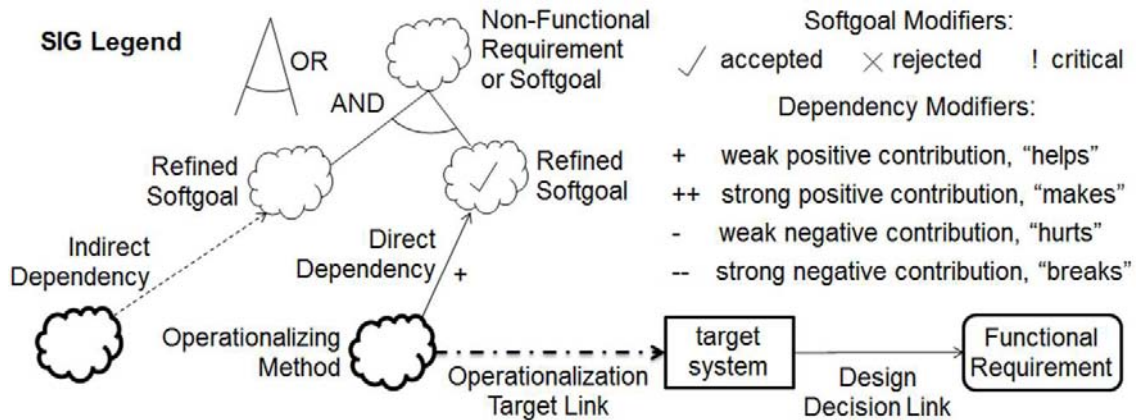


Figure B.53 Softgoal Interdependency Graphs "SIGs"

Other important labels that appear in SIGs are check marks ✓ when a softgoal is fulfilled (or chosen to be implemented), cross marks X when a softgoal can not be realized (or chosen not to be implemented), and exclamations ! when a softgoal is deemed to be critical or important.

2.2 SIGs Micro-business Example

A micro-business example for the SIGs concepts in the previous subsection could be an illustration of how the softgoals affect the touch screen in the fast-food restaurant example in the previous subsection. A non-functional requirement that could be illustrated is reliability – the ability of the entire system to perform what it is supposed to do and to perform it consistently.

The softgoal of reliability could be further refined to reliability of the software and the reliability of the hardware. In the refined softgoal of reliability of the hardware, a direct dependency with a positive contribution can be linked to the operationalizing method which would be a physical touch screen unit. The touch screen unit is linked to the target system via an operationalization target link.

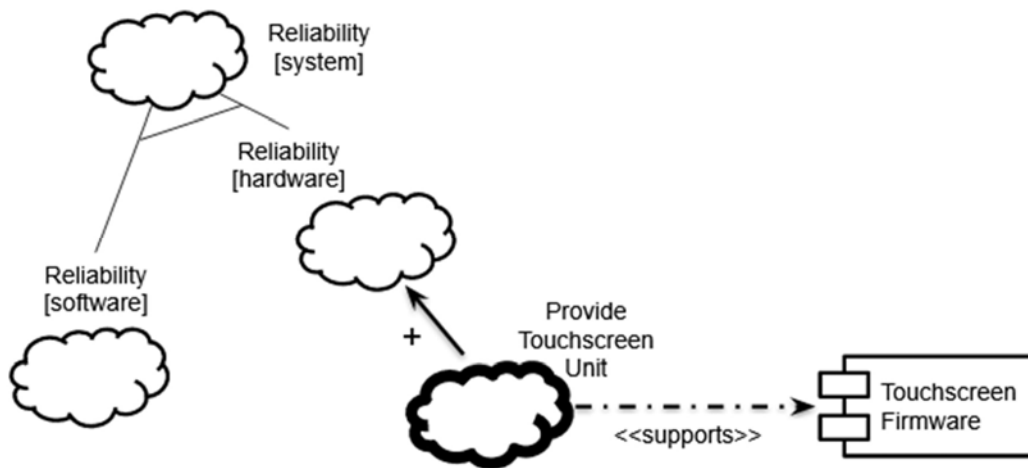


Fig. III.54 SIGs illustrating the softgoal of reliability in the fast food restaurant

3. Unified Modeling Language “UML”

3.1 Class Diagram (Structural)

A class is a template for creating objects in UML. In a micro-business software system, an example of a class could be the register class for the actual cash register at the store.

3.1.1 Class Diagram (Structural) Concepts

A Class Diagram is represented as a rectangle with three sections. The top section is for the name, the middle section is for the attributes, and the bottom section is for the operations. The middle and bottom section in class diagrams are optional.

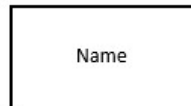


Figure B.55 A Class

In a micro-business software system, “register” could be the name of the class for the actual cash register at the store.



Figure B.56 A Class Example

When a class composes and contains another class, a filled diamond is attached to the class.



Figure B.57 A Class Composed of and Containing another Class

In a micro-business software system, a menu class can contain a product class since a product that a customer can order in a fast food restaurant can be found in the menu.

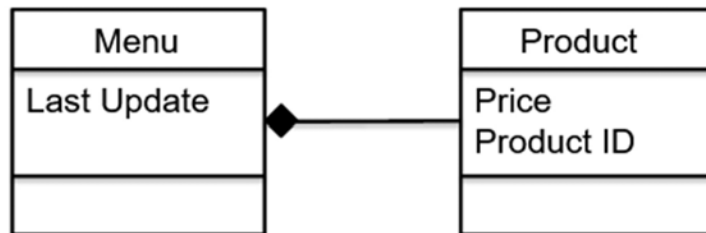


Figure B.58 A Class Composed of and Containing another Class Example

When a class composes but does not contain another class, a hollow diamond is attached to the class.

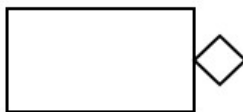


Figure B.59 A Class Composed of but not Contained by another Class

In a micro-business software system, the register class for the cash register could be composed of but not contain the sale class.

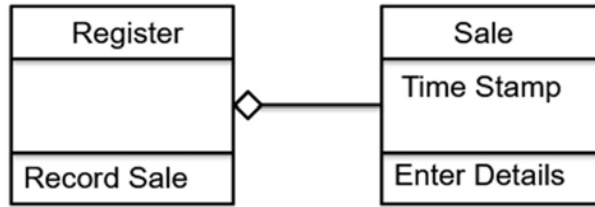


Figure B.60 A Class Composed of but not Contained by another Class Example

Inheritance is indicated by a hollow arrow and refers to the ability of the child class to inherit the functionality of the super class and add new functionality of its own.



Figure B.61 Inheritance among Classes

In a micro-business software system, the savings account class in the payroll system for the employees could inherit all the functionalities of a bank account class in the payroll system for the employees.



Figure B.62 Inheritance among Classes Example

An action of one class to another is indicated by a hollow arrow above the line that connects both classes. The hollow arrow is usually accompanied by descriptive text of such action.

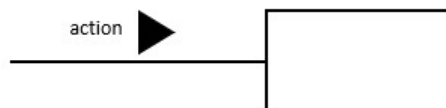


Figure B.63 Actions among Classes

In a micro-business, the kid's set menu class could have an action that gives away the crayons and paper placemat class.

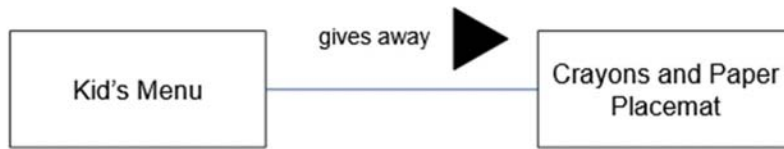


Figure B.64 Actions among Classes Example

One-to-one, many-to-one, one-to-many, and many-to-many relationships are indicated with 1's and *'s on the lines between the classes. The one-to-one relationship between classes is shown in the figure below.

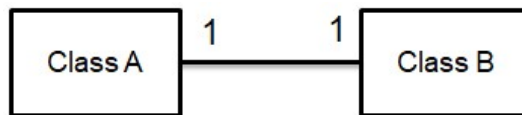


Figure B.65 One-to-one Relationship between Classes

In a micro-business, an example of a one-to-one relationship is the customer payment class and the customer's receipt class. There will always be only one customer's receipt for every unique payment that they have made.



Figure B.66 One-to-one Relationship between Classes Example

The one-to-many relationship between classes is shown in the figure below.

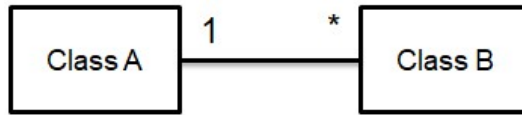


Figure B.67 One-to-many Relationship between Classes

In a micro-business, an example of a one-to-many relationship is the product class to the price class. One specific kind of product or item on the menu could have several different prices depending on market demand or seasonality.



Figure B.68 One-to-many Relationship between Classes Example

The many-to-one relationship between classes is shown in the figure below.

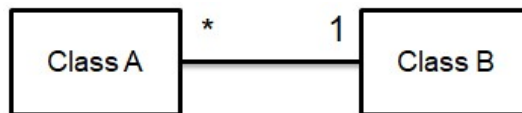


Figure B.69 Many-to-one Relationship between Classes

In a micro-business, an example of a many-to-one relationship is the price class **to** the product class (notice the position of the classes with respect to the word “**to**”). There could be several different prices for a product depending on market demand and seasonality of a product.



Figure B.70 Many-to-one Relationship between Classes Example

The many-to-many relationship between classes is shown in the figure below.

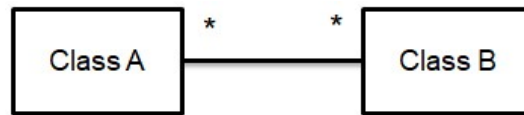


Figure B.71 Many-to-many Relationship between Classes

In a micro-business, an example of a many-to-many relationship is the relationship between waiters and customers. One waiter can serve many customers and many waiters can serve one customer.



Figure B.72 Many-to-many Relationship between Classes Example

3.1.2 Class Diagram (Structural) Example

In order to show how class diagrams (structural) could be applied in a micro-business example, some classes in the system of the fast food restaurant example in the previous sections could be illustrated. In this micro-business software system example, there is a physical, brick-and-mortar fast food restaurant that exists. Each fast food restaurant has a menu which customers can choose from. Each fast food restaurant has one or several touch screens for customers to place their order and pay by credit card and one or several cashiers where customers can pay in cash.

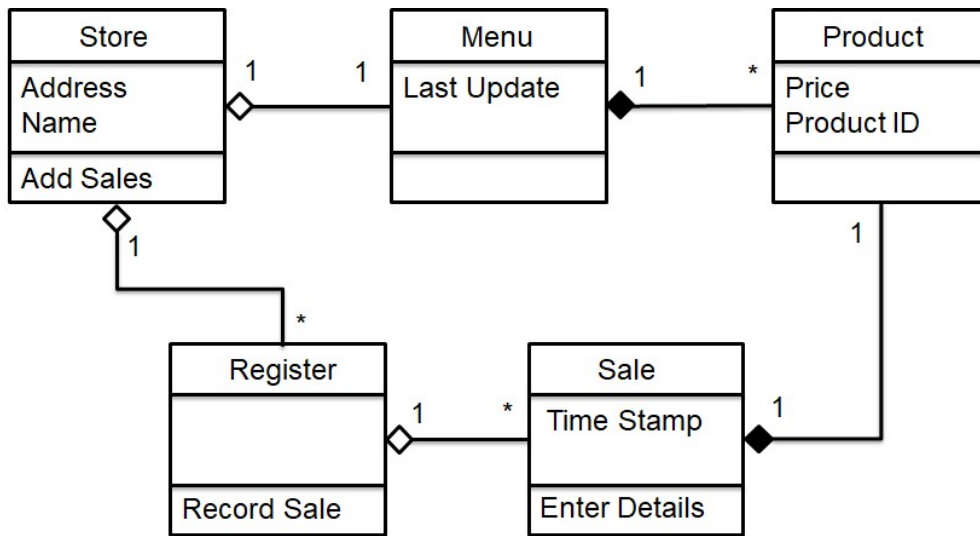


Figure B.73 A Sample Class Diagram (Structural) of the Fastfood Restaurant

3.2 Component Diagram (Structural)

Components are autonomous, encapsulated units in a system or subsystem that provides one or more interfaces. For example, a Point-of-Sale (POS) component forms part of an entire micro-business software system. After discussing concepts of component diagrams (structural), an example of a component diagram as part of a micro-business software system is provided at the end of this subsection.

3.2.1 Component Diagram (Structural) Concepts

Component representation is made with a rectangle and a component symbol on the upper right corner.

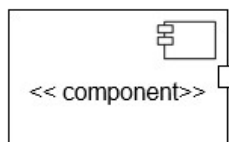


Figure B.74 Representation of a Component

The provided interface of a component is represented by a circle connected to a line to the component.

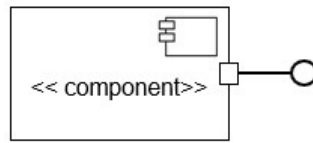


Figure B.75 The Provided Interface of a Component

The required interface of a component is represented by a half circle connected to a line to the component.

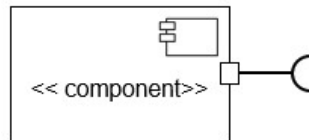


Figure B.76 The Required Interface of a Component

Connecting components with their provided and required interfaces is represented with the circle connecting to the half circle among components. In a micro-business software system, a Point-of-Sale component may require an Employee component if it needs to verify the identity of the user before it records any transactions.

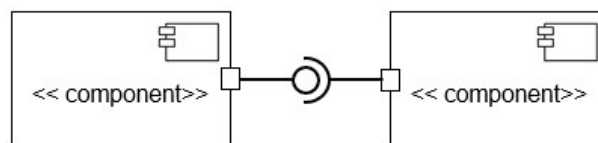


Figure B.77 Connecting the Required and Provided Interface among Components

3.2.2 Component Diagram (Structural) Example

A micro-business example that could be used to show the concepts of the Component Diagram (Structural) discussed in the previous subsection would be to illustrate some of the software components of the fast food restaurant example in the previous section. The software of the fast food restaurant is composed of a point-of-sale (POS) component which takes the orders and collects payments from customers. There is an inventory component which tracks the ingredients used to prepare the food until they are in their final form to be consumed by the customer. There is an employee management component which tracks the time-in and time-out of employees. Only employees that are timed-in are allowed access to all the other software components in the fast food restaurant system.

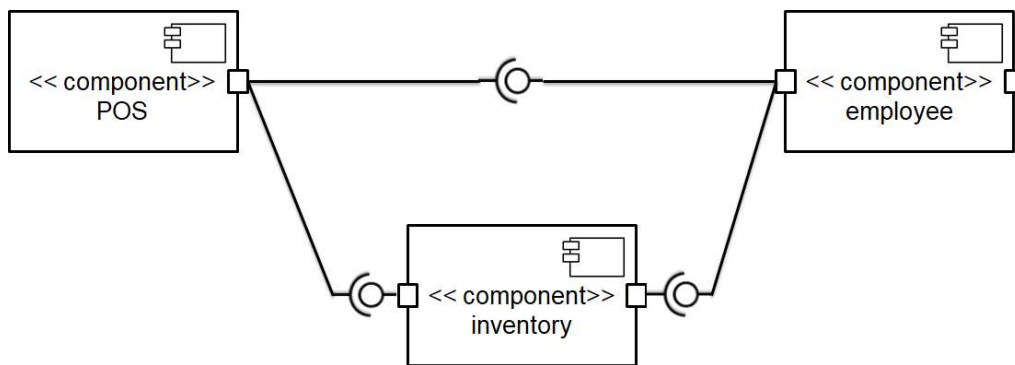


Figure B.78 Component Diagram (Structural) Example of the fast food restaurant software

Appendix C

Action Research Material for Micro-businesses

(Everything in Appendix C is available on <http://www.pentathlonsystems.com/ar4mb.html>)

1. Sample Form used to Evaluate μ RP Diagram Comprehensibility

Evaluation Form for the Representation of Micro-business Processes and Patterns

What is your name? (Optional)

What is your e-mail address? (Optional, leave an e-mail if you want the evaluation results sent to you)

What is your background?

Business student (undergrad/postgrad)

Com Sci student (undergrad/postgrad)

Academic (Researcher/Professor/PhD/PostDoc)

Software developer

Business Owner/Representative

Other

If you selected "Other," what is your background?

How many years have you been in this background (refer to previous questions)?

What kind of diagrams have you used (or are familiar with)?

Unified Modeling Language "UML" (or its extensions such as SysML)

Data Flow Diagrams "DFDs"

Business Process Modeling Notation "BPMN"

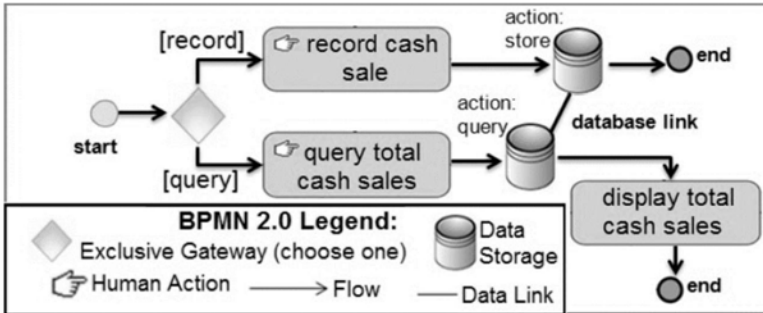
Domain Specific Languages "DSLs"

Others

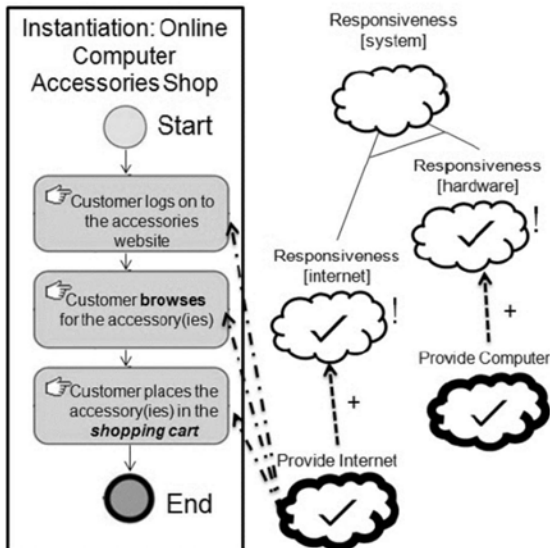
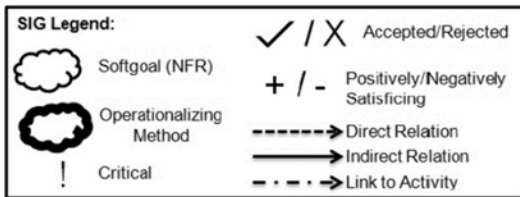
If you chose "DSL" or "Others," please specify them

Are you aware that business process patterns are used (and reused) during the development of software?

- Yes
- No
- Not Sure



Can you explain, in a few sentences, what is going on in this diagram?



Can you explain, in a few sentences, what is going on in this diagram?

Micro-business Requirements Patterns

User Guide v5.2

Micro-business Requirements Patterns

Contents	2
----------------	---

Part 1. Overview of the Approach

1.1 User Pre-requisites.....	3
1.2 Tool Support.....	3
1.3 Source Publication.....	4
1.4 Other Sources.....	4
1.4 Activities in the Approach.....	5
1.4.1 Observation of Micro-businesses.....	6
1.4.2 Identification of Common and Varying Requirements.....	6
1.4.3 Tabulation (Table Sample).....	7 (8)
1.4.4 Iterations.....	9
1.4.5 Complementary Implementation Notes.....	9

Part 2. Representation in the Approach

2.1 Core Concepts.....	10
2.2 Modes/Options and Choices/Answers.....	10
2.3 Using Labels.....	11
2.4 Combining BPMN and SIGs.....	12
2.5 Software Components.....	13
2.6 Evaluation of the Approach.....	14
2.6.1 Evaluation of Suitability.....	14
2.6.2 Evaluation of Comprehensibility.....	15

Part 1. Overview of the Approach

1.1 User Pre-requisites

It is recommended to have an understanding of UML, BPMN, SIGs, and a year of software development experience when using this approach. Refresher tutorials for these notations are available at <http://pentathlonsystems.com/ar4mb.html>

1.2 Tool Support

The following tools are recommended for use with this approach. Exercises are available at <http://pentathlonsystems.com/tutorials/Tutorial%204%20-%20Available%20Tools.pptx>

1.2.1 RE-Tools

RE-tools is capable of diagramming and representing BPMN + SIGs + components + the custom pattern labels discussed in this user guide. Below is the download link:

https://personal.utdallas.edu/~chung/Sam_Supakkul/RE-Tools/index.html

1.2.2. StarUML

StarUML is an open-source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. The goal of the StarUML project is to build a software modeling tool and also platform that is a compelling replacement of commercial UML tools. Below is the download link:

<http://staruml.sourceforge.net/en/download.php>

1.3 Source Publication

The following is the publication citation related to this user guide. It is **highly recommended** to read this publication. The download link provided by the Journal of Universal Computer Science:

https://www.jucs.org/jucs_25_7/micro_business_requirements_patterns/

An evaluator's copy may be requested from the primary author at any time. The citation is as follows:

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2019). ***Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations***. Journal of Universal Computer Science JUCS 25 (7), (pp. 764-787).

1.4 Other Sources

Kotonya, G. & Sommerville, I. (2003). Requirements Engineering: Processes and Techniques. England. John Wiley and Sons Limited.

Kouroshfar, E., Shahir, H. Y. & Ramsin, R. (2009). Process Patterns for Component-Based Software Development. In G. A. Lewis, I. Poernomo & C. Hofmeister (eds.), CBSE (pp. 54-68). Springer. ISBN: 978-3-642-02413-9. doi: 10.1007/978-3-642-02414-6_4

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S. & Chung, L. (2013). 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, (pp.1-12), IEEE. ISBN: 978-1-4673-2912-5. doi: 10.1109/RCIS.2013.6577703

1.5 Activities in the Approach

An overview of the approach is presented below, followed by a step-by-step summary of the activities.

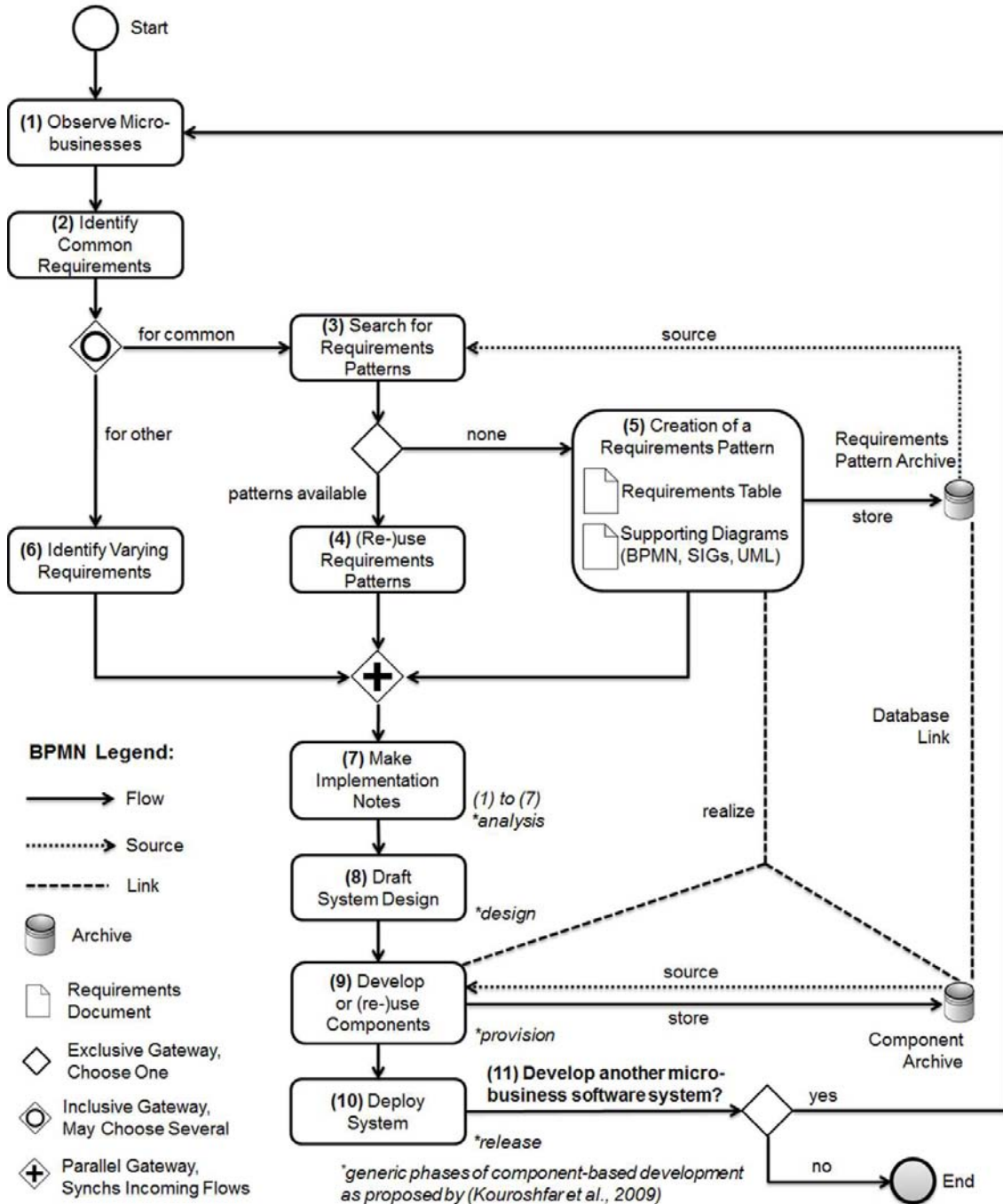


Figure 1. Overview of the Approach, adapted from Macasaet et al., 2019

1.4.1 Observation of Micro-businesses

- 1) List down goals and requirements of two or more micro-businesses.
- 2) Decompose goals into requirements. Below is a sample of how Kotonya and Sommerville perform this activity.

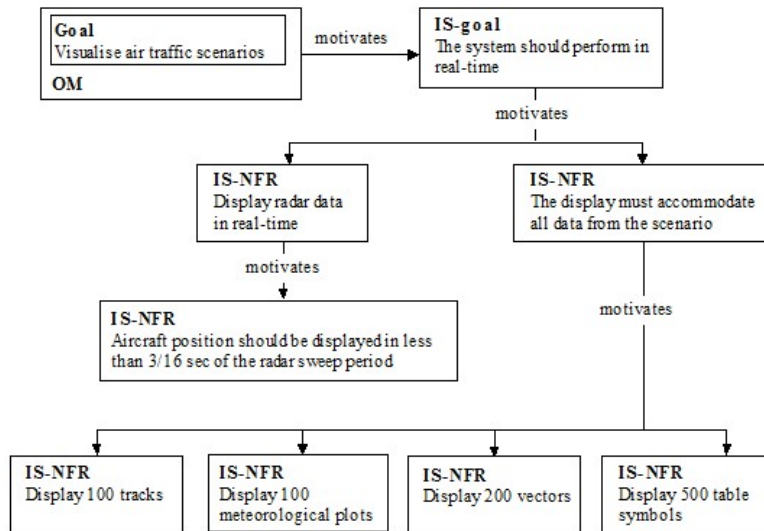


Figure 2. Decomposing goals into requirements by Kotonya and Sommerville, 2003

- 3) Divide requirements into: functional and non-functional.

1.4.2 Identification of Common and Varying Requirements

- 1) Group the common requirements. The minimum number of common requirements to constitute a pattern is 2 (If there was nothing in common, there would be no pattern in the first place). On the next page is a brief example of how requirements are grouped and diagrammed.
- 2) A pattern diagram is a visual representation of the requirements in terms of (business) processes as shown on the next page. BPMN is recommended to illustrate the processes but other languages such as UML may be used.
- 3) Identify uncommon requirements.

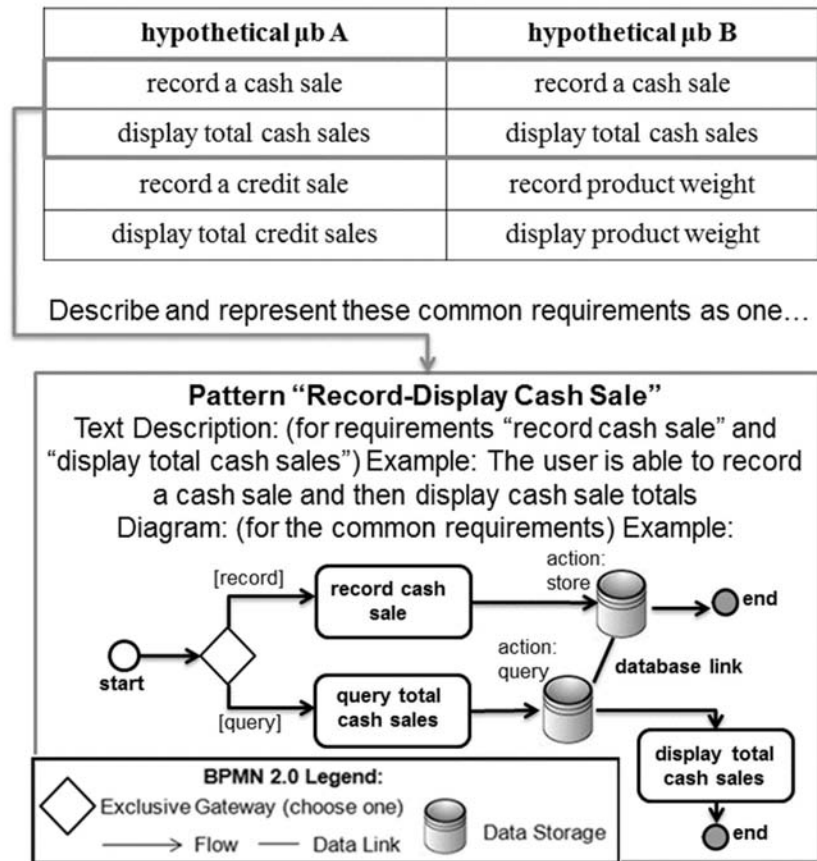


Figure 3. Grouping common requirements, adapted from Macasaet et al., 2013

1.4.3 Pattern Tabulation

Tabulating the requirements data helps organize the requirements (common and uncommon). In order to tabulate a pattern, a requirement(s) is (are) transformed into a non-technical form, such as a business-type question, e.g.

From (Requirement):

display available products online (*μ B side*)

To (Non-technical form):

does the customer shop online?

This is done in order to group the common requirements as shown on the next page.

Table I. Requirements in a Table

Restaurant µBRP		
description: µb sources ingredients, µb produces food for customers to eat, and then customer pays.		
keywords: ingredients, food, restaurant, delivery		
Functional Requirements Section, Q&A to be answered by µb owner		
The question-answer structure allows the developer to clarify the hard goals or functional requirements of the µb owner.		
Question	Options	Choice(s)
(a) How will the µb gather the ingredients for the food to be served?	buy at market / have ingredients delivered by third party / other select as many that apply	
(b) How can customers order food at the restaurant µb?	in-house menu / phone / website / mobile app / other select as many that apply	
(c) How do customers eat the food from the restaurant?	in-house / take away / delivery select as many that apply	
(d) How can customers pay the restaurant µb?	credit / debit / Paypal / other, select as many that apply	
(e) What roles do the µb-side users have on the system?	admin / manager / user / other select as many that apply	
(f) Is the restaurant µb linked to a customer relationship management system? (µb inventory pattern can be found in the Appendix)	yes / no	
(g) Is the restaurant µb linked to an inventory management system? (µb inventory pattern can be found at (Macasaet et al., 2012) and (Macasaet et al., 2014))	yes / no	
(h) Is the restaurant µb linked to a sales management system? (µb sales pattern can be found at (Macasaet et al., 2013))	yes / no	
(i) Is the restaurant µb linked to a logistics management system?	yes / no	
Non-Functional Requirements Section, to be ranked in terms of priority by µb owner		
The priorities are ranked, 1st as top priority, in order to clarify the soft goals or NFRs.		
- Micro-business side -		µb Ranking
How important is _____ to the µb owner?		
system speed (responsiveness)		
affordability of software system (cost)		
deployment time (short implementation period)		
user-friendliness of the software system, admin side (usability)		
ease of maintaining the software system		
security of the software system (data protection)		
exactness (preciseness) of data		
How important is _____ to the customers of the µb?		
quickness (responsiveness) of the food service		
quickness (responsiveness) of customer payment		
quickness (responsiveness) of the software system (online restaurant site/app)		
user-friendliness of the software system, customer side (online restaurant site/app)		
security of the software system (transactions made on online restaurant site/app)		
availability of the food on the menu (availability of ingredients)		
delivery time of the food (fast food delivery)		
- Developer side -		Dev Ranking
How important is _____ to the developer?		
immediate profitability (mobilization, project price)		
long-term profitability (subscriptions)		
deployment time (short implementation period)		
ease of delivering the software project		
ease of maintaining the software system after deployment		

1.4.4 Emergence of patterns in succeeding iterations

For every succeeding micro-business, the software developer does the same activities of observing and listing down requirements. The new list of requirements is compared to all other existing requirements. Similar and varying requirements are identified. Similar requirements are grouped as previously explained. If there are no similar requirements then software components are simply developed to satisfy/satisfice the requirements.

1.4.5 Complementary Implementation Notes

Due to the complexity of NFRs, complementary implementation notes are normally added by the software developer in order to satisfice the NFRs. Examples of complementary implementation notes are shown below.

“discuss reliability of hosting servers and internet providers”

“bill the client on a project-basis or on a subscription-basis”

“hire more staff in order to reduce the time to complete the project”

Part 2. Representations in Micro-business Requirements Patterns

2.1 Core Concepts

Goals and sub-goals are decomposed into requirements. Micro-business processes satisfy/satisfice the goals of the business. Requirements are classified as functional or non-functional. Software components satisfy/satisfice the requirements. Software components realize the micro-business requirements patterns.

2.2 Modes/Option and Choices/Answers

The micro-business requirements pattern has questions which have modes/options and choices/answers. First, a question is [done as] a mode and then the micro-business owner [chooses] the answer(s). This concept is shown below:

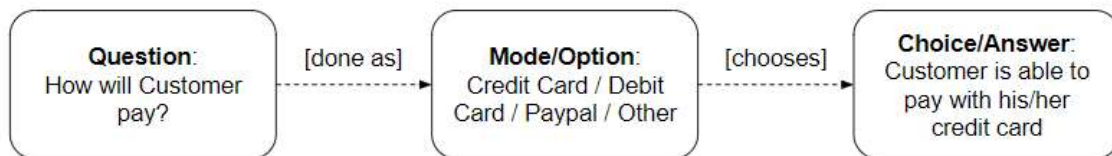


Figure 4. The question has a mode/option and a choice/answer

2.3 Using Labels

When using BPMN diagrams, it is recommended to label business processes. For example, a [done as] label can be placed which links a business process activity to its modes. This could help (other) developers identify possible solutions. An example of a [done as] label which represents modes from Table I is shown below:

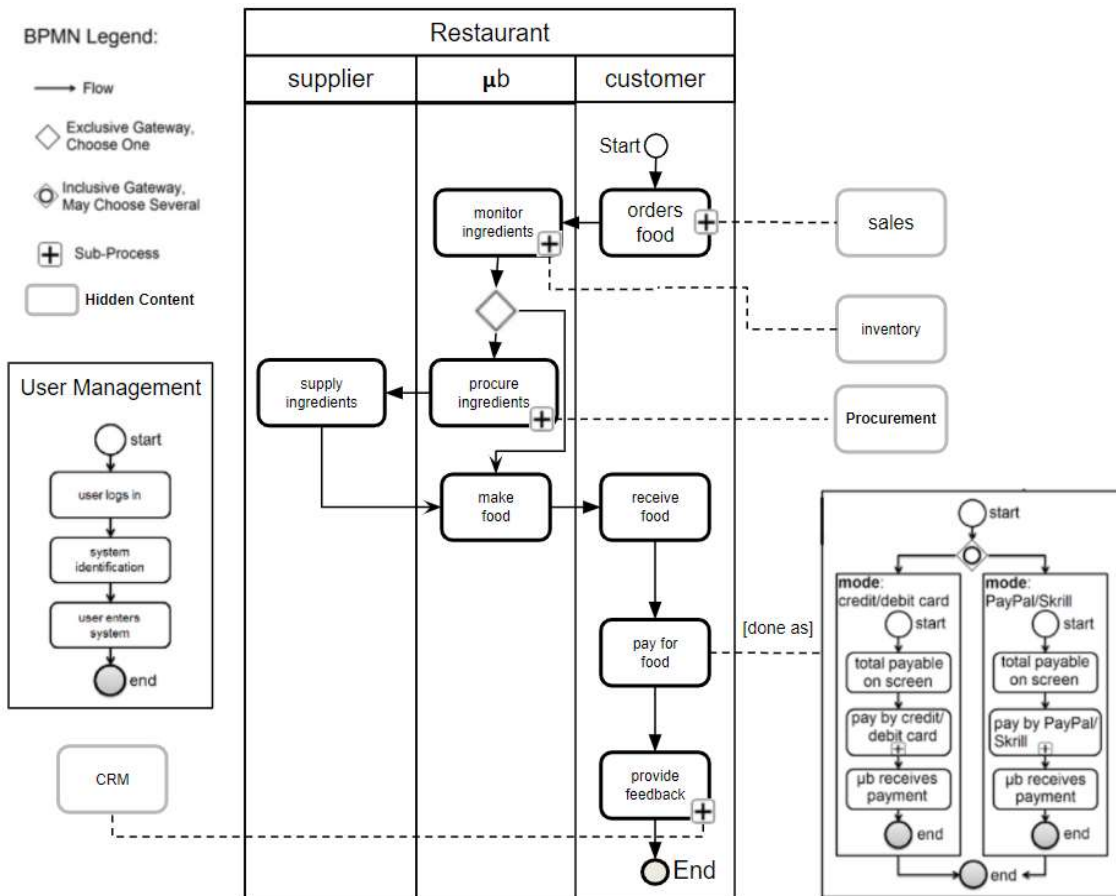


Figure 5. Placing a [done as] label on the modes of micro-business requirements patterns

2.4 Combining BPMN and SIGs

Combining both BPMN and SIGs through an “operationalization target link” allows developers and users to see how the non-functional requirements relate to the activities in a business process. The concept is shown below.

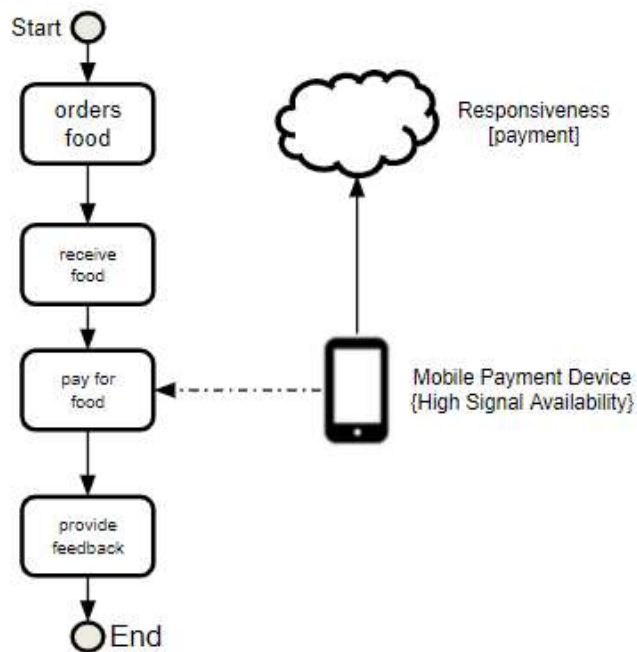


Figure 6. An example of combining BPMN and SIGs in one diagram

2.5 Software Components

2.5.1 Placement of components in the diagrams

Business analysts may already find the aforementioned models sufficient to understand and utilize patterns but software developers may want to represent the placement of components in their diagrams as well. Components may be placed in between the operationalizing method which contributes to the component's function and the business activity which the component supports.

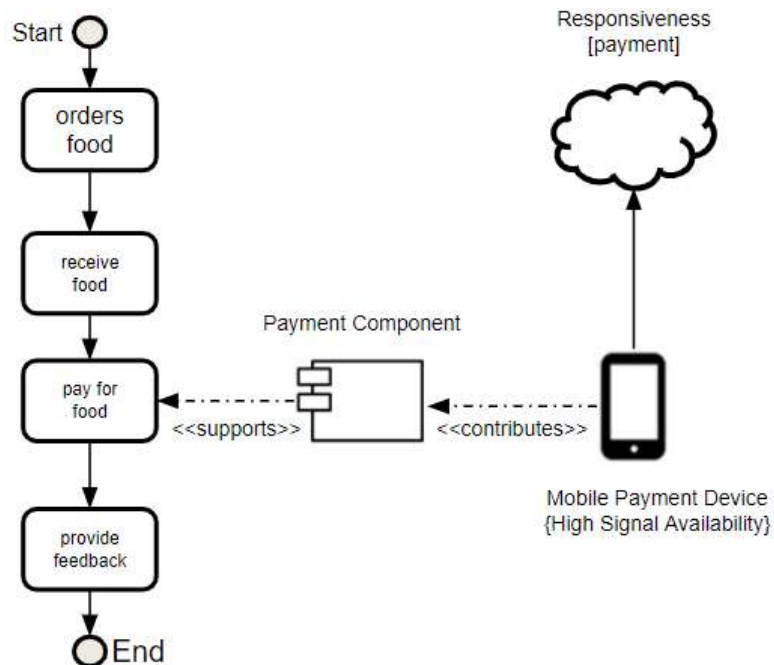


Figure 7. Placement of component representations

2.6 Evaluation of the Approach

2.6.1 Evaluation of Suitability

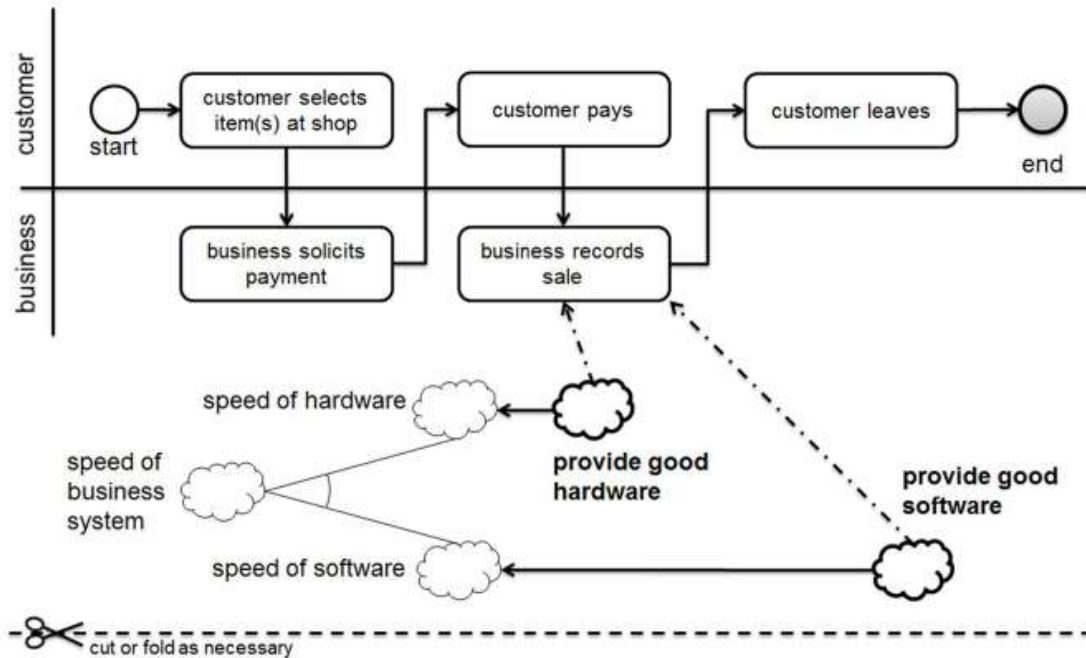
Evaluation is done in the style of “**Action Research.**” **The approach author will participate and try to promote the approach when it is being applied in a software project. Users provide feedback and results when the approach is applied.** The approach author may be contacted at **any time** for questions at rjmacasaet@pentathlonssystems.com

1. Take note of how many times you reused software components **before** applying this approach AND the overall duration of the software project in man days (you may round-up man days, i.e., if 8 hours of development time by one person is 1 man day then 9 hours would be rounded-up to 2-man days.)
2. In a similar software project (as similar as possible), take note of how many times you reused software components **after** applying this approach AND the overall duration of the software project. You may have as many “similar projects” as possible.
3. Did the approach help promote the reuse of software components?
Yes or No? Describe your experience with the approach.
4. Would you **recommend** any improvements to the approach?

2.6.2 Evaluation of Comprehensibility

2.6.2.1 Sample Diagram Comprehensibility Form

The form below may be used to evaluate whether a micro-business owner/stakeholder comprehends the diagrams used in this approach.



Developer Notes:

- (1) Did the micro-business stakeholder explain the business process in correct chronological order?
- (2) What other remarks did the micro-business stakeholder make (about the softgoal interdependency graphs "SIGs")?

Figure 8. Sample Diagram Comprehensibility Form

2.6.2.2 Diagram Comprehensibility Evaluation Test

Visit www.pentathlonsystems.com/eval3.html and complete the evaluation.

----- End of User Manual -----

3. Tutorials

3.1 Tutorial 1 – Basic BPMN and SIGs

<p>Tutorial 1 – Basic BPMN and SIGs</p> <p>RJ Macasaet R&D Dept.</p>	<p>Outline</p> <ul style="list-style-type: none"> I. Basic BPMN symbols II. Sample BPMN diagram III. Basic SIGs symbols IV. Sample SIGs diagram V. Sample BPMN and SIGs diagram together 	<p>Learn basic BPMN symbols</p> <p>BASIC BPMN</p>
<p>I. Basic BPMN constructors</p> <ul style="list-style-type: none"> The sequence flow defines the execution order of the activities <p>default flow – if all other conditions are false</p> <p>conditional flow – used if the condition holds true</p>	<p>I. Basic BPMN constructors</p> <ul style="list-style-type: none"> Start event and end event – where flows begin and end. <p>start end</p>	<p>I. Basic BPMN constructors</p> <ul style="list-style-type: none"> An activity task is a unit of work and is the job to be performed. When marked with a symbol it indicates a Sub-Process, an activity that can be refined. <p>activity activity</p>
<p>I. Basic BPMN constructors</p> <ul style="list-style-type: none"> A data object represents information flowing through a process such as a document, email, or letter A data store is a place where the process can read or write data such as a database or a filing cabinet. It exists even outside the whole process 	<p>I. Basic BPMN constructors</p> <ul style="list-style-type: none"> Pools and lanes represent responsibilities for activities in a process. A pool/lane may be an organization, a role, or a system. Lanes further subdivide pools or other lanes hierarchically. 	<p>I. Basic BPMN constructors - Gateways</p> <ul style="list-style-type: none"> Exclusive Gateway - When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.
<p>I. Basic BPMN constructors - Gateways</p> <ul style="list-style-type: none"> Inclusive Gateway - When splitting, one or more branches are activated. All active incoming branches must complete before merging. 	<p>I. Basic BPMN constructors - Gateways</p> <ul style="list-style-type: none"> Parallel Gateway - When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging, parallel branches wait for all incoming branches to complete before triggering the outgoing flow 	<p>I. Basic BPMN constructors - Gateways</p> <ul style="list-style-type: none"> Event-based gateway - is always followed by catching events or receiving tasks. Sequence flow is routed to the subsequent event/task which happens first

I. Basic BPMN constructors - Gateways

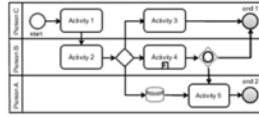
- Complex gateway - Complex merging and branching behavior that is not captured by other gateways



13

II. BPMN Sample Diagram

- Exercise: Can you explain the following diagram?



14

Learn basic SIGs symbols
BASIC SIGS

15

III. Basic SIGs Symbols

- A non-functional requirement (NFR) softgoal



- An operationalizing method (cloud in bold)



16

III. Basic SIGs Symbols - Interdependency

- Direct "explicit" relationship (of softgoals)



- Indirect "implicit" relationship, correlation



17

III. Basic SIGs Symbols - Interdependency

- the following symbols are added to the arrows to further define the interdependencies

++ + - --

and appears, for example, as the arrow below



18

III. Basic SIGs Symbols - Interdependency

- "to satisfy" means to be good enough
- when one positive "+" symbol is added to an arrow, this indicates that there is "help" or some "weak positive contribution" that helps satisfy a softgoal but does not satisfy it just by itself



19

III. Basic SIGs Symbols - Interdependency

- when two positive "++" symbols are added to an arrow, this indicates that there is a "make" or some "strong positive contribution" that can satisfy a softgoal by itself



20

III. Basic SIGs Symbols - Interdependency

- when one negative "-" symbol is added to an arrow, this indicates that there is a "hurt" or some "weak negative contribution" that hampers the achievement of a softgoal but does NOT by itself, prevent satisfying the softgoal



21

III. Basic SIGs Symbols - Interdependency

- when two negative "--" symbols are added to an arrow, this indicates that there is a "break" or some "strong negative contribution" that by itself, prevents the achievement of the softgoal



22

III. Basic SIGs Symbols

- Other important symbols
 - ✓ Accepted – softgoal is fulfilled (or chosen to be implemented)
 - ✗ Rejected/Denied – softgoal can not be realized (or is chosen NOT to be implemented)
 - ! Critical – an important softgoal
 - and/or – used to group (sub) softgoals

23

IV. Custom Operationalizing Methods

- Operationalizing methods (clouds in bold) can be further specified with custom symbols and labels



24

V. Sample SIGs Diagram

- Exercise: Can you explain the following diagram?

25

Learn how to illustrate BPMN and SIGs together

BPMN + SIGS

26

V. Sample BPMN + SIGs together

- The following symbol is used to link an operationalizing method (from SIGs) to a business process activity task (from BPMN) and is referred to as an "operationalization target link"

- For now, we can link BPMN and SIGs with this symbol. In tutorial 3 - component representation, we discuss further refinements to this symbol

27

V. Sample BPMN + SIGs together

- Exercise: Can you explain the following diagram?

28

Questions?
email:
rjmacaesat@pentathlonsystems.com

29

3.2 Tutorial 2 – Pattern Representation

Tutorial 2 – Pattern Representation

RJ Macasaet
R&D Dept.

1

Outline

- I. Overview of a requirements-based approach for representing micro-business patterns
- II. Pattern Essentials
 - A. emergence
 - B. diagrams
 - C. tabulation
 - D. modes and instantiations
 - E. custom symbols

2

Get an idea of the approach from a general perspective

OVERVIEW

3

I. Approach Overview

4

I. Approach Overview - Observation of Micro-businesses

1. List down goals and requirements of two or more micro-businesses.
2. Decompose goals into requirements [sample method: see next slide adapted from Kotonya and Sommerville]
3. Divide requirements into: functional and non-functional.

5

Decompose goals into requirements – from Kotonya and Sommerville

6

I. Approach Overview - Identification of Common and Varying Requirements

1. Group the common requirements. The minimum number of common requirements to constitute a pattern is 2 (if there was nothing in common, there would be no pattern in the first place).
2. Identify uncommon requirements.
3. Tabulating the pattern helps in organizing the requirements (common and uncommon, to be discussed in the next section)

7

I. Approach Overview - Emergence of patterns in succeeding iterations

- For every succeeding micro-business, the software developer does the same activities of observing and listing down requirements. The new list of requirements is compared to all other existing requirements. Similar and varying requirements are identified. Similar requirements are grouped as stated in the previous slide
- If there are no similar requirements then software components are simply developed to satisfy/satisfice the requirements.

8

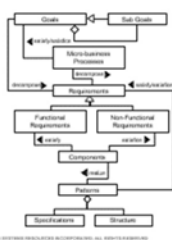
I. Approach Overview - Extra Implementation Notes

- Due to the complexity of NFRs, additional implementation notes are normally added by the software developer in order to satisfy the NFRs. Refer to the source publication for more details.

9

I. Approach Overview – Metamodel

- Here we have the artefacts of the approach represented in a metamodel



10

PATTERN ESSENTIALS

11

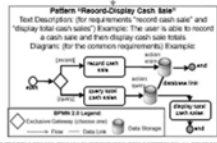
II. Pattern Essentials - emergence

- Common requirements are grouped
- The minimum number of common requirements to constitute a pattern is 2
- There are as many as $n!/(k!(n-k)!)$ sub-patterns, where n is the number of common requirements in a pattern and k is the number of common requirements in a grouping
- The next slide illustrates how common requirements are grouped into patterns (note: μb stands for micro-business)

12

Hypothetical job A	Hypothetical job B
record a cash sale	record a cash sale
display total cash sales	display total cash sales
record a credit sale	record product weight
display total credit sales	display product weight

Describe and represent these common requirements as one...



13

II. Pattern Essentials - diagrams

- The previous slide also shows a pattern diagram which is a visual representation of the requirements in terms of (business) processes.
- We recommend using BPMN to illustrate the processes but other languages and notations such as UML, SysML, or DFDs may be used.

14

II. Pattern Essentials - tabulation

- In order to tabulate a pattern, (a) requirement(s) must be transformed into a non-technical form, such as a question, i.e. Requirement:
display available products online (μb side)
Non-technical form:
does the customer shop online?
- This is done in order to group the requirements as shown in the table on the next slide

15

II. Pattern Essentials - tabulation

- Refer to user guide or source publication for details

16

II. Pattern Essentials – modes and instantiations

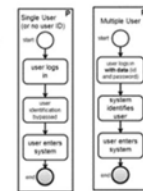
- A pattern is an abstracted concept and in order use it, it must be [done as] a mode and then [applied as] an instantiation.



17

II. Pattern Essentials – labels

- "P" labels are placed in business process activities corresponding to the modes of the requirements pattern



18

II. Pattern Essentials – custom symbols

- The concept also applies when connecting SIGs as well.

Questions?
email:
rjmacasaet@pentathlonsystems.com

19
20

3.3 Tutorial 3 – Component Representation

**Tutorial 3 –
Component Representation**

RJ Macasaet
R&D Dept.

Outline

- I. Basic Concept
- II. Representation

Understand the idea of component representation within the approach

BASIC CONCEPT

I. Basic Concept

- Where is the component represented?

I. Basic Concept

- An operationalizing method
 - must be something “measurable” and “observable”
- contributes to the “ilities” of the component
- The component “supports” the instantiation (of the pattern)

I. Basic Concept

Metamodel reference to the components

Learn how to represent the software components within the approach

REPRESENTATION

II. Representation

- Hence, the basic component representation...

Custom Miniature Component Representation – for easy viewing by the developer

II. Representation

Classic Component Diagram – for more detailed viewing

II. Representation

II. Representation

- Exercise: illustrate the software components that are used in a retail store software system (sales and inventory system). You may use existing BPMN/SIGs process diagrams.

Questions?
Email:
rjmacasaet@pentathlonsystems.com

1
2
3
4
5
6
7
8
9
10
11
12

3.4 Tutorial 4 – Available Tools

Tutorial 4 – Available Tools

RJ Macasaet Sam Supakkul
R&D Principal Architect
PSRI Sabre, Inc.

1 ★

Outline

I. RE-Tools
II. Bonitasoft (BPMN)
III. (UML)

2 ★

I. RE-Tools

- The following tool is capable of diagramming and representing BPMN + SIGs + the custom pattern symbols as discussed in the previous tutorials. It has been presented at the Requirements Engineering Conference in 2012 (RE '12) and in the Conference of Advanced Information Systems Engineering in 2013 (CAISE '13). Below is the download link:

<http://www.csi.cmu.edu/~rjmacasaet/Tools/DownloadInstallation.html>

3 ★

I. RE-Tools

- Video demonstrations
 - Create SIGs using RE-tools
<http://youtu.be/gT5XyEiOMx8>
 - Create BPMN diagram using RE-Tools (with connection to SIGs)
<http://youtu.be/ljVdmdXWV4>
- Exercise: Create a retail store business process (the sale of an item) showing BPMN and SIGs

4 ★

II. Bonitasoft

- The following is an open-source BPM and workflow engine that allows users to define, execute and monitor business processes. Below is the download link:

<http://www.bonitasoft.com/en/ct/download-bpm-software-and-documentation>

- Exercise: Create a retail store business process (storage of inventory) using Bonitasoft

5 ★

III. StarUML

- StarUML is an open source project to develop fast, flexible, extensible, featureful, and freely-available UML/MDA platform running on Win32 platform. The goal of the StarUML project is to build a software modeling tool and also platform that is a compelling replacement of commercial UML tools. Below is the download link:

<http://staruml.sourceforge.net/en/download.php>

6 ★

III. StarUML

- Exercise: Create a software component diagram for a retail store business process (the sale and the storage of items) using StarUML
- Final Exercise: Show your entire retail store business process (including software components) in one diagram using any of the tools mentioned in this tutorial

7 ★

Questions?

Email:
rjmacasaet@pentathlonsystems.com
or
sam.supakkul@sabre.com

8 ★

3.5 Tutorial 5 – UML

Tutorial 5 - UML

RJ MACASAET
R&D Department

1 ▲

Two types of UML Diagrams

- Behavioral
- Structural*

2 ▲

Class Diagrams

- A class is represented as a rectangle with three sections. The top section is for the name, the middle section is for the attributes, and the bottom section is for the operations. The middle and bottom section in class diagrams are optional.

Name

3 ▲

Class Diagrams

- Classes composed and contained by

◆

4 ▲

Class Diagrams

- Classes composed without belonging to

◊

5 ▲

Class Diagrams






- Inheritance is indicated by a hollow arrow and refers to the ability of the child class to inherit the functionality of the super class and add new functionality of its own.

Savings Account

→

Bank Account

6 ▲

<p>Class Diagrams</p> <ul style="list-style-type: none"> Action (e.g. A satisfies B) 	<p>Component Diagrams</p>	<p>Component Diagrams</p> <ul style="list-style-type: none"> Components are considered autonomous, encapsulated units within a system or subsystem that provide one or more interfaces. Component representation is made with a rectangle and a component symbol on the upper right corner. 
<p>7</p>	<p>8</p>	<p>9</p>
<p>Component Diagrams</p> <ul style="list-style-type: none"> Provided Interface 	<p>Component Diagrams</p> <ul style="list-style-type: none"> Required Interface 	<p>Component Diagrams</p> <ul style="list-style-type: none"> Sample Component Diagram 
<p>10</p>	<p>11</p>	<p>12</p>
<p>Questions?</p> <p>email: rjmacasae@pentathionystems.com</p>		
<p>13</p>		

4. BPMN Quick Reference Sheet (care of bpm-b.de)

BPMN 2.0 - Business Process Model and Notation <http://bpm-b.de/poster>

Activities

- Task**: A Task is a unit of work, the job to be performed. When marked with a symbol it indicates a Sub-Process, an activity that can be refactored.
- Transaction**: A Transaction is a set of activities that logically belong together. It might follow a specified transaction protocol.
- Event Sub-Process**: An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can run in parallel (non-terminating) depending on the start event.
- Call Activity**: A Call Activity is a wrapper for a globally defined Task or Process reused in the current Process. A call to a Process is marked with a .

Activity Markers
Markers indicate execution behavior of activities:

- Sub-Process Marker
- Loop Marker
- Parallel ID Marker
- Sequential ID Marker
- Ad Hoc Marker
- Compensation Marker

Task Types
Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

Flow Types

- Sequence Flow**: defines the execution order of activities.
- Default Flow**: is the default branch to be chosen if all other conditions evaluate to false.
- Conditional Flow**: has a condition assigned that defines whether or not the flow is used.

Conversations

- A Conversation defines a set of logically related message exchanges. When marked with a symbol it indicates a Sub-Conversation, a compound conversation element.
- A Call Conversation is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a .
- A Conversation Link connects Conversations and Participants.

Conversation Diagram

Choreographies

- A Choreography Task represents an Interaction (Message Exchange) between two Participants.
- A Sub-Choreography contains a refined choreography with several interactions.
- A Call Choreography is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a .

Choreography Diagram

Events

None: Untriggered events, indicate start point, state changes or final states.

Message: Receiving and sending messages.

Timer: Cyclic timer events (points in time, time spans or intervals).

Exception: Evaluating to an higher level of responsibility.

Conditional: Reacting to changed business conditions or integrating business rules.

Link: Off-page connections. Two corresponding link events signal a sequence flow.

Error: Catching or throwing named errors.

Cancel: Reacting to cancelled transactions or triggering compensation.

Compensation: Handling or triggering compensation.

Signal: Signalling across different processes. A signal event can be caught multiple times.

Multiple: Catching one out of a set of events. Throwing all events defined.

Parallel Multiple: Catching all out of a set of parallel events.

Terminate: Triggering the immediate termination of a process.

	Start	Intermediate	End
None			
Message			
Timer			
Exception			
Conditional			
Link			
Error			
Cancel			
Compensation			
Signal			
Multiple			
Parallel Multiple			
Terminate			

Gateways

- Exclusive Gateway**: When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it ensures one incoming branch to complete before triggering the outgoing flow.
- Event-based Gateway**: Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.
- Parallel Gateway**: When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.
- Inclusive Gateway**: When splitting, one or more branches are activated. All active incoming branches must complete before merging.
- Exclusive Event-based Gateway**: Each occurrence of a subsequent event starts a new process instance.
- Complex Gateway**: Combines merging and branching behavior that is not captured by other gateways.
- Parallel Event-based Gateway**: The occurrence of all subsequent events starts a new process instance.

Collaboration Diagram

Swimlanes

Pools (Participants) and Lanes represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events. The message flow can be decorated with an envelope depicting the content of the message.

Data

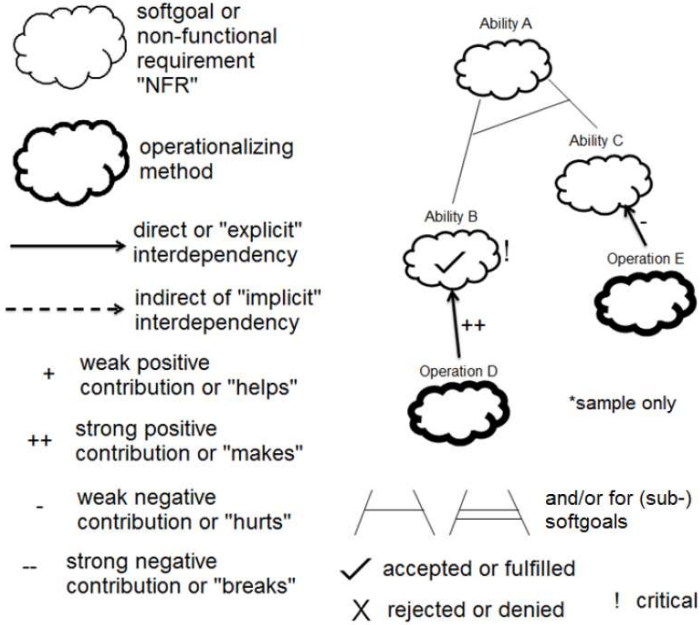
- A Data Object represents information flowing through the process, such as business documents, emails, or letters.
- A Collection Data Object represents a collection of information, e.g., a list of order items.
- A Data Input is an external input for the entire process. A kind of input parameter.
- A Data Output is data result of the entire process. A kind of output parameter.
- A Data Association is used to associate data elements to activities, Processes and Global Tasks.
- A Data Store is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

© 2011

5. SIGs Quick Reference Sheet

Softgoal Interdependency Graphs "SIGs"

adapted from the work of Lawrence Chung



Bibliography

ABB. (2013). ABB Automation Products. Last accessed on October 6, 2013 at <http://www.abb.com/>

Adolph, S., Hall, W. & Kruchten, P. (2011). Using Grounded Theory to study the experience of software development. *Empirical Software Engineering*, 16, (pp. 487-513). doi:10.1007/s10664-010-9152-6

Ahmad, K. & Zabri, S. (2018). The mediating effect of knowledge of inventory management in the relationship between inventory management practices and performance: The case of micro retailing enterprises. *Journal of Business and Retail Management Research*, 12, 2. (pp. 83-93). doi: 10.24052/JBRMR/V12IS02/TMEOKOIMITRBIMPAPTCOMRE

Akkaoui, Z. E. & Zimányi, E. (2009). Defining ETL workflows using BPMN and BPEL. In I.-Y. Song & E. Zimányi (eds.), DOLAP (pp. 41-48), ACM. ISBN: 978-1-60558-801-8

Akkaoui, Z. E., Zimányi, E., Mazón, J.-N. & Trujillo, J. (2011). A model-driven framework for ETL process development. In I.-Y. Song, A. Cuzzocrea & K. C. Davis (eds.), DOLAP (pp. 45-52), ACM. ISBN: 978-1-4503-0963-9

Albert, C. & Brownsword, L. (2002). Evolutionary Process for Integrating COTS-Based Systems (EPIC). CMU/SEI Technical Report CMU/SEI-2002-TR-005, 2002
url: <ftp://ftp.sei.cmu.edu/public/documents/02.reports/pdf/02tr005.pdf>

Aleksy, M. & Stieger, B. (2011). Mobile Service Business Patterns. In *Proceedings of the IEEE 25th International Conference on Advanced Information Networking and Applications AINA* (pp. 62-68). IEEE. doi: 10.1109/AINA.2011.74

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language*. Oxford University Press, New York.

Alonso, A., Kok, S., Sakellarios, N., & O'Brien, S. (2018). Micro-enterprises, self-efficacy and knowledge acquisition: Evidence from Greece and Spain. *Journal of Knowledge Management* 23 (3), (pp.419-438). doi: 10.1108/JKM-02-2018-0118

Alrajeh, D., Kramer, J., Russo, A. & Uchitel, S. (2009). Learning operational requirements from goal models. ICSE (pp. 265-275), IEEE. ISBN: 978-1-4244-3452-7

Alrajeh, D., Kramer, J., Russo, A. & Uchitel, S. (2013). Elaborating Requirements Using Model Checking and Inductive Learning. *IEEE Trans. Software Eng.*, 39, (pp. 361-383).

Alsanoosy, T., Spichkova, M., & Harland, J. (2018). Cultural influences on the requirements engineering process: lessons learned from practice. In: 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS). IEEE, (pp 61–70)

Alsanoosy, T., Spichkova, M. & Harland, J. (2020) Cultural influence on requirements engineering activities: a systematic literature review and analysis. *Requirements Eng* 25, (pp. 339–362). <https://doi.org/10.1007/s00766-019-00326-9>

Álvarez, J. A. T., Nicolás, J., Moros, B. & Garcia, F. (2002). Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach. *Requir. Eng.*, 6, (pp. 205-219).

- Allen, C. (2001). Realizing e-business with components. Addison-Wesley, Harlow, Boston
- Allen, T. (1977). Managing the Flow of Technology. MIT Press
- Alnusair, A., & Zhao, T. (2010). Component Search and Reuse: a ontology-based approach. In proceedings of the IEEE International Conference on Information Reuse and Integration IRI, (pp. 258-261).
- Ambler, S. (2002). Agile modeling. John Wiley and Sons
- Ampatzoglou, A. & Chatzigeorgiou, A. (2007). Evaluation of object-oriented design patterns in game development. *Information and Software Technology*, 49 (May (5)), (pp. 445–454), Elsevier. doi:10.1016/j.infsof.2006.07.003.
- Aranda, J., Easterbrook, S. M. & Wilson, G. (2007). Requirements in the wild: How small companies do it. *Requirements Engineering RE* (pp. 39-48), IEEE. ISBN: 0-7695-2935-6. doi: 10.1109/RE.2007.54
- Aranda, J. (2010). Playing to the Strengths of Small Organizations. In *Proceedings of the 1st Workshop on RE in Small Companies RESC*, (pp. 141-144). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf
- Arnold, H. (1982). Moderator variables: A clarification of conceptual, analytic, and psychometric issues, *Organizational Behavior and Human Performance*, (29) 2, (pp. 143-174), ISSN 0030-5073, doi:10.1016/0030-5073(82)90254-9
- Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., Muthig, D., Paech, B., Wüst, J., & Zettel, J. (2002). Component based product line engineering with UML. Addison-Wesley.
- Atkinson, C., Bayer, J., Laitenberger, O., Zettel, J. (2000). Component-based Software Engineering: The KobrA Approach. In 22nd International Conference on Software Engineering (ICSE 2000), 3rd International Workshop on Component-based Software Engineering, Limerick, Ireland
- Atlassian. (2021). Atlassian. Last accessed on January 30, 2021 on www.atlassian.com
- Azar, J., Smith, R.K., & Cordes, D. (2007). Value-oriented requirements prioritization in a small development organization, *IEEE software*, vol. 24, no. 1, (pp. 32–37)
- Bae, D.-H. (2007). Software Process Improvement for Small Organizations. *COMPSAC* (1). (p. 17). IEEE Computer Society. ISBN: 978-0-7695-2870-0. doi:10.1109/COMPSAC.2007.193
- Bagozzi, R. P. (1980). Causal models in marketing. New York, NY: Wiley. ISBN: 0471015164
- Ballurio, K., Scalzo, B. & Rose, L. (2002). Risk Reduction in COTS Software Selection with BASIS.. In J. C. Dean & A. Gravel (eds.), *ICCBSS* (pp. 31-43), Springer. ISBN: 3-540-43100-4
- Bandara, A. K., Lupu, E. C., Moffett, J., Heslington & Russo, A. (2004). A Goal-based Approach to Policy Refinement. Proceedings of the Fifth IEEE International Workshop on

- Policies for Distributed Systems and Networks, POLICY 2004, (pp. 229-239), doi: 10.1109/POLICY.2004.1309175
- Baron, R. & Kenny, D. (1986). The moderator-mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of personality and social psychology*, 51, (pp. 1173—1182). doi:10.1037/0022-3514.51.6.1173
- Barros, O. (2007) Business process patterns and frameworks: Reusing knowledge in process innovation. *Business Process Management Journal*, 13 (1), (pp. 47-69). doi: 10.1108/14637150710721122
- Beck, K. (2005). Extreme Programming Explained: Embrace Change; 2nd Edition. Addison-Wesley Professional
- Berczuk, S.P., (2003). Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Addison-Wesley Professional, 2003
- Bhuta, J., Mattmann, C., Medvidovic, N. & Boehm, B. W. (2007). A Framework for the Assessment and Selection of Software Components and Connectors in COTS-Based Architectures. Working IEEE/IFIP Conference on Software Architecture WICSA. (p. 6). IEEE Computer Society. ISBN: 978-0-7695-2744-4. doi: 10.1109/WICSA.2007.2
- Bilandzic, M. & Venable, J. (2011). Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics. *J. Community Informatics*, 7. Last accessed on October 9, 2013 at <http://ci-journal.net/index.php/ciej/article/view/786/804>
- Bishop, M. (2003). Computer Security: Art and Science. Addison Wesley
- Bizagi. (2013). Bizagi Business Process Management. Last accessed on October 6, 2013 on <https://www.bizagi.com/>
- Blau, P. & Schoenherr, R. (1971). The Structure of Organizations. Basic Books
- Boehm, B. W. (2000). Requirements that Handle IKIWISI, COTS, and Rapid Change.. IEEE Computer, 33, (pp. 99-102).
- Boehm, B. W., Port, D., Yang, Y., Bhuta, J. & Abts, C. (2003). Composable Process Elements for Developing COTS-Based Applications.. ISESE (pp. 8-17), : IEEE Computer Society. ISBN: 0-7695-2002-2
- Bosch, J. (2000). Design and use of software architectures. Addison-Wesley, England
- Boukheduoma, S., Oussalah, M., Alimazighi, Z., & Tamzalit, D. (2013). Adaptation Patterns for Service-Based Inter-Organizational Workflows. In *Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS*, (pp. 1-10). IEEE. doi: 10.1109/RCIS.2013.6577722
- Brennan, R., Canning, L., & McDowell, R. (2008). Business-to-Business-Marketing, Sage Publications Limited, London
- Brito, F. & Abreu, E. (1995). The MOOD metric set. In proceedings of ECOOP '95 Workshop on Metrics

Budgen, D., Turner, M., Brereton, P., & Kitchenham, B. (2008). Using mapping studies in software engineering. in 20th Annual Psychology of Programming Interest Group Conference, PPIG. Lancaster University, United Kingdom

Buschmann, F., & Meunier, R. (1995). A System of Patterns. Pattern Languages of Program Design, 1, May, 1995, (pp. 325-343).

Bürsner, S. & Merten, T. (2010). RESC 2010: 1st Workshop on Requirements Engineering in Small Companies', in Workshop Proceedings of Requirements Engineering for Software Quality REFSQ 2010, ICB-Research Report no. 40, October 2010, (pp. 128-130). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Cardoso, E., Almeida, J., Guizzardi, R., & Guizzardi, G. (2011). A Method for Eliciting Goals for Business Process Models Based on Non-Functional Requirements Catalogues. *International Journal of Information System Modeling and Design*, 2 (2), (pp. 1-18) doi: 10.4018/jismd.2011040101

Carver, J. (2006). 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* (pp. 42). Springer. ISBN: 978-3-540-71300-5. doi: 10.1007/978-3-540-71301-2_15

Chandler, P., & Sweller, J. (1991). Cognitive Load Theory and the Format of Instruction. *Cognition and Instruction* (8:4), (pp. 293-332)

Charmaz, K. (2006). *Constructing Grounded Theory: a practical guide through qualitative analysis*. London; Thousand Oaks, Calif.: Sage Publications.

Cheesman, J., Daniels, J. & Szyperski, C. (ed.) (2001). UML Components - A Simple Process for Specifying Component-Based Software. Addison-Wesley.

Chung, L., Nixon, B., Yu, E. & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engineering*. Boston, Dordrecht, London. Kluwer Academic Publishers.

Chung, L., Supakkul, S., Subramanian, N., Garrido, J. L., Noguera, M., Hurtado, M. V., Rodríguez, M. L. & Akhlaki, K. B. (2011). Goal-Oriented Software Architecting.. In P. Avgeriou, J. Grundy, J. G. Hall, P. Lago & I. Mistrík (ed.), *Relating Software Requirements and Architectures*, (pp. 91-109). Springer. ISBN: 978-3-642-21000-6.

Chung, L., Hill, T., Legunsen, O., Sun, Z., Dsouza, A. & Supakkul, S. (2013). A goal-oriented simulation approach for obtaining good private cloud-based system architectures. *Journal of Systems and Software*, 86, (pp. 2242-2262). doi: 10.1016/j.jss.2012.10.028

Clements, P., Kazman, R., & Klein, M. (2002). *Evaluating software architectures: methods and case studies*. Addison-Wesley, Boston

Comella-Dorda, S., Dean, J. C., Morris, E. & Oberndorf, P. (2002). A Process for COTS Software Product Evaluation. In J. C. Dean & A. Gravel (eds.), *COTS-Based Software Systems, First International Conference, ICCBSS <p> 2002, Orlando, FL, USA, February 4-6, 2002, Proceedings* (pp. 86--96), Berlin, u.a.: Springer Verlag.

Coplien, J.O. (1995a). A development process generative pattern language, AT&T Bell Laboratories. url: <http://www.bell-labs.com/people/cope/Patterns/Process/index.html> last accessed on October 6, 2013

- Coplien, J.O. (1995b). A generative development-process pattern language. In: Coplien, J.O., Schmidt, D.O. (Eds.), *Pattern Languages of Program Design*. Addison Wesley, Reading, MA, (pp. 183-237)
- Crabtree, C. A., Seaman, C. B. & Norcio, A. F. (2009). Exploring language in software process elicitation: A Grounded Theory approach. *ESEM* (pp. 324-335), ISBN: 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5315984
- Creswell, J. W. (ed.) (1994). *A Qualitative Procedure in Research Design. Qualitative and Quantitative Approaches*. London and New Delhi. Sage.
- Crnkovic, I., Hnich, B., Johnson, T., Kiziltan, Z., (2002). Specification, implementation, and deployment of components. *Communications, Association of Computing Machinery* 45 (October (10)), (pp. 35–40). doi: 10.1145/570907.570928
- Crnkovic, I. & Larsson, M. (2000). A Case Study: Demands on Component-based Development. ICSE'2000 -- *International Conference on Software Engineering* (pp. 23-31), Limerick, Ireland. doi: 10.1109/ICSE.2000.870393
- Crnkovic, I. & Larsson, M. (2002). Challenges of component-based development. *Journal of Systems and Software*, 61, (pp. 201-212). doi: 10.1016/S0164-1212(01)00148-0
- Cugola, G. & Ghezzi, C. (1998). Software processes: a retrospective and a path to the future. *Software Process: Improvement and Practice*, 4, (pp. 101-123). doi: 10.1002/(SICI)1099-1670(199809)4:3<\$101::AID-SPIP103>\$3.0.CO;2-K
- Davis, J. A. (1985). *The Logic of Causal Order* (Vol. 07-055). Beverly Hills, London, New Delhi: Sage.
- Davis, C.J., Fuller, R.M., Tremblay, M.C., & Berndt, D.J. (2006). Communication Challenges in Requirements Elicitation and the Use of the Repertory Grid Technique. In *Journal of Computer Information Systems*, 46, (5), 78. url: <http://www.uta.edu/faculty/richarme/MARK%205338/Davis%20repertory%20grid.pdf>
- Desarrollo TIC. (2018). Desarrollo TIC. Last accessed on December 15, 2018 at <http://www.desarrollotic.com>
- DPDL. (2013). Design Pattern Definition Language DPDL. Last accessed on October 6, 2013 on <http://alshayeb.com/DPDL/>
- Dobing, B., & Parsons, J. (2006). How UML is Used. *Communications of the ACM* 49:5, (pp. 109-113)
- dos Santos, P. S. M. & Travassos, G. H. (2009). Action Research use in software engineering: An initial survey. *ESEM* (pp. 414-417). ISBN: 978-1-4244-4842-5. doi: 10.1109/ESEM.2009.5316013
- dos Santos, P. S. M. & Travassos, G. H. (2011). Action Research Can Swing the Balance in Experimental Software Engineering. *Advances in Computers*, 83, (pp. 205-276). doi: 10.1016/B978-0-12-385510-7.00005-9
- Drew, C.J. & Hardman, M.L. (1985). *Designing and Conducting Behavioral Research*. Pergamon, New York, NY.

- Dwyer, M. B., Avrunin, G. S. & Corbett, J. C. (1999). Patterns in Property Specifications for Finite-State Verification. In B. W. Boehm, D. Garlan & J. Kramer (eds.), ICSE (pp. 411-420). ACM. ISBN: 1-58113-074-0. DOI: 10.1145/302405.302672.
- Dybå, T. (2003). Factors of software process improvement success in small and large organizations: an empirical study in the Scandinavian context. ESEC / SIGSOFT FSE (pp. 148-157), ACM. doi: 10.1145/940071.940092
- Eden, A. (1999). Precise Specification of Design Patterns and Tool Support in Their Application. PhD Thesis, University of Tel Aviv, Israel, 1999.
- El-Boussaidi, G. & Mili, H. (2012). Understanding design patterns - what is the problem? *Software: Practice and Experience*, 42, (pp. 1495-1529). doi: 10.1002/spe.1145
- Elizondo, P. V. & Lau, K.-K. (2010). A catalogue of component connectors to support development with reuse. *Journal of Systems and Software*, 83, (pp. 1165-1178). doi: 10.1016/j.jss.2010.01.008
- Emam, K. E. & Madhavji, N. H. (1995). A field study of requirements engineering practices in information systems development. *Requirements Engineering*, pp. 68-80, IEEE Computer Society.
- European Commission. (2013). User Guide to the SME Definition. Last accessed on April 3, 2021 at https://ec.europa.eu/regional_policy/sources/conferences/state-aid/sme/smedefinitionguide_en.pdf
- Everyware. (2018). Everyware Technologies. Last accessed on December 15, 2018 at <http://www.everywaretech.es>
- Fayad, M., Laitinen, M., & Ward, R. (2000). Thinking objectively: software engineering in the small, *Communications of the ACM* 43, (pp. 115-118). doi: 10.1145/330534.330555
- Firesmith, D. (2004). Specifying Reusable Security Requirements.. *Journal of Object Technology*, 3, 61-75.
- Fowler, M. (1997). *Analysis patterns: Reusable Object Models*. Addison Wesley Longman, Inc.
- Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional
- France, R., Kim, D.-K., Ghosh, S. & Song, E. (2004). A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering*, 30, (pp. 193-206). doi: 10.1109/TSE.2004.1271174
- France, R., Kim, D., Song, E., Ghosh, S. (2002). *Role-Based Modeling Language (RBML) Specification v1.0*. Technical Report 02-106, Computer Science Department, Colorado State University, Fort Collins, Colorado, June, 2002
- Franch, X., Palomares, C., Quer, C., Renault, S. & Lazzer, F. D. (2010). A Metamodel for Software Requirement Patterns. In R. Wieringa & A. Persson (eds.), *Requirements*

Engineering for Software Quality REFSQ (pp. 85-90), Springer. ISBN: 978-3-642-14191-1. doi: 10.1007/978-3-642-14192-8_10.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley.

Garlan, D., Allen, R. & Ockerbloom, J. (1995). Architectural Mismatch or Why It's Hard to Build Systems Out Of Existing Parts. In D. E. Perry, R. Jeffrey & D. Notkin (eds.), *ICSE* (pp. 179-185), ACM. ISBN: 0-89791-708-1

Gemino, A., & Wand, Y. (2005). Complexity and Clarity in Conceptual Modeling: Comparison of Mandatory and Optional Properties. *Data & Knowledge Engineering* 55:3, (pp. 301-326)

Georgakopoulos, D. & Jayaraman, P.P. (2016) Internet of Things: from internet scale sensing to smart services. *Computing* 98(10), pp. 1041–1058

Ghobadian, A. & Gallea, D. (1997) TQM and organization size. *International Journal of Operations & Production Management*, 17, (pp. 121-163)

Glaser, B. G., Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New York, NY: Aldine de Gruyter.

Glushko, R.J. & McGrath, T. (2002). Document engineering for e-business. *ACM Symposium on Document Engineering*, pp. 42-48, ACM

Glushko, R., & McGrath, T. (2008). *Document Engineering – Analyzing and Designing Documents for Business Informatics and Web Services*. Cambridge, MA, USA. MIT Press.

Goldkuhl, G. (2008). Practical Inquiry as Action Research and Beyond. In W. Golden, T. Acton, K. Conboy, H. van der Heijden & V. K. Tuunainen (eds.), *ECIS* (pp. 267-278). Last accessed on October 9, 2013 at <http://aisel.aisnet.org/ecis2008/118>

Goldkuhl G. (2012). From Action Research to practice research. *Australasian Journal of Information Systems*, 17, 2, (pp. 57-78). url: <http://dl.acs.org.au/index.php/ajis/article/view/688>.

Grant, D. & Ngwenyama, O. K. (2003). A report on the use of Action Research to evaluate a manufacturing information systems development methodology in a company. *Inf. Syst. J.*, 13, (pp. 21-36). doi: 10.1046/j.1365-2575.2003.00137.x

Gregory, D. & Ward, H. (1974). *Statistics for Business Studies*. McGraw-Hill, London, England.

Greyfinch. (2019). Greyfinch.com. Last accessed on August 17, 2018

Hadar, I., Reinhartz-Berger, I., Kuflik, T., Perini, A., Ricca, F. & Susi, A. (2013). Comparing the comprehensibility of requirements models expressed in Use Case and Tropos: Results from a family of experiments. *Information & Software Technology*, 55, (pp. 1823-1843). doi: 10.1016/j.infsof.2013.05.003.

Harel, D. & Pnueli, A. (1985). On the Development of Reactive Systems. *Logics and models of concurrent systems*, (pp. 477-498)

Happel, H.J., Maalej, W., & Seedorf, S. (2010). Applications of ontologies in collaborative software development. In I. Mistrík, J. Grundy, A. Hoek, & J. Whitehead (Eds.), *Collaborative*

Software Engineering (pp. 109-129). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-10294-3_6

Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G.K., & Svendsen, A. (2008). Adding Standardized Variability to Domain Specific Languages. In Proceedings of the 12th International Software Product Line Conference SPLC, (pp. 139–148). doi: 10.1109/SPLC.2008.25v

Hoffmann, A., Söllner, M. & Hoffmann, H. (2012a). Twenty Software Requirement Patterns to Specify Recommender Systems that Users will Trust. ECIS. (p. 185) Last accessed on October 9, 2013 at <http://aisel.aisnet.org/ecis2012/185>

Hoffmann, A., Söllner, M., Hoffmann, H. & Leimeister, J. M. (2012b). Towards trust-based software requirement patterns. RePa, (pp. 7-11), IEEE. ISBN: 978-1-4673-4374-9

Hruby, P. (2006). Model-Driven Design Using Business Patterns. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Hsueh, N.-L., Chu, P.-H. & Chu, W. C. (2008). A quantitative approach for evaluating the quality of design patterns. *Journal of Systems and Software*, 81, (pp. 1430-1439). doi: 10.1016/j.jss.2007.11.724

Huston, B. (2001). The effects of design pattern application on metric scores. *Journal of Systems and Software*, 58, (pp. 261-269). doi: 10.1016/S0164-1212(01)00043-7

IBM WSC. (2012). IBM WebSphere Commerce. Last accessed on October 6, 2013 at <http://pic.dhe.ibm.com/infocenter/wchelp/v6r0m0/index.jsp>

IEEE Computer Society. (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard

Immes, S. (1993). Wahrgenommenes Risiko bei der industriellen Kaufentscheidung, Trier

International Organization for Standardization (ISO). (2011). ISO/IEC DTR 29110-1:2011 Software Engineering – Lifecycle Profiles for Very Small Entities (VSEs) – Part 1: Overview. ISO, Switzerland, 2011. Retrieved October 9, 2013, from <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>

Jantunen, S. (2010). 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*, (pp. 131-140). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Kalenborn, A. (2010). Modelling by Example: Requirements engineering during the bidding stage of dialog-oriented software projects. In Proceedings of the 1st Workshop on RE in Small Companies RESC, (pp. 158-166). url: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReportNo40.pdf

Kamsties, E., Hormann, K., & Schlich, M. (1998). Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer. In *Conference on European Industrial Requirements Engineering CEIRE*, London, United Kingdom. url: <http://prof.kamsties.com/download/ceire98.pdf>

Kang, K., Cohen, S., Hess, J., Novak, W. & Peterson, S. (1990). Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-21). Software Engineering Institute, Carnegie Mellon University

Kang, K. C., Lee, J. & Donohoe, P. (2002). Feature-Oriented Project Line Engineering. *IEEE Software*, 19 (4), (pp. 58-65), doi: 10.1109/MS.2002.1020288

Kassab, M. (2021). How Requirements Engineering is Performed in Small Businesses? 29th International Requirements Engineering Conference Workshops (REW), 2021, IEEE, (pp. 220-223), doi: 10.1109/REW53955.2021.00041.

Kauppinen, M., Kujala, S., Aaltio, T. & Lehtola, L. (2002). Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice. *RE* (pp. 43-51). *IEEE Computer Society*. ISBN: 0-7695-1465-0. doi: 10.1109/ICRE.2002.1048504

Kauppinen, M., Vartiainen, M., Kontio, J., Kujala, S. & Sulonen, R. (2004). Implementing requirements engineering processes throughout organizations: success factors and challenges. *Information & Software Technology*, 46, (pp. 937-953). doi: 10.1016/j.infsof.2004.04.002

Kelliher, F. & Reinl, L. (2009). A resource-based view of micro-firm management practice. *Journal of Small Business and Enterprise Development*, 16 (3), (pp. 521-532).

Khwaja S. & Alshayeb M. (2013). Towards design pattern definition language. *Software: Practice and Experience*, 43, (pp. 747-757). doi: 10.1002/spe.1122.

Kilov, H. & Sack, I. (2009). Mechanisms for communication between business and IT experts. *Computer Standards & Interfaces*, 31(1), (pp. 98-109). doi: 10.1016/j.csi.2007.11.001

Kim, D., France, R. & Ghosh, S. (2004). A UML based language for specifying domain-specific patterns. *Journal of Visual Languages and Computing*, 15(3-4): (pp.265-289) doi: 10.1016/j.jvlc.2004.01.004

Kock, N. (2004). The three threats of Action Research: a discussion of methodological antidotes in the context of an information systems study. *Decision Support Systems*, 37 (2), (pp. 265-286). doi: 10.1016/S0167-9236(03)00022-8

Kotonya, G. & Sommerville, I. (2003). *Requirements Engineering: Processes and Techniques*. England. John Wiley and Sons Limited.

Kouroshfar, E., Shahir, H. Y. & Ramsin, R. (2009). Process Patterns for Component-Based Software Development. In G. A. Lewis, I. Poernomo & C. Hofmeister (eds.), CBSE (pp. 54-68). Springer. ISBN: 978-3-642-02413-9. doi: 10.1007/978-3-642-02414-6_4

Kouskouras, K., Chatzigeorgiou, A., & Stephanides, G. (2008). Facilitating software extension with design patterns and Aspect-Oriented Programming. *Journal of Systems and Software* 81 (October (10)), (pp. 1725–1737), Elsevier. doi: 10.1016/j.jss.2007.12.807

Krogstie, J. & Sølvsberg, A. (2003). Information Systems Engineering - Conceptual Modeling in a Quality Perspective. Kompendiumforlaget, NTNU, Trondheim, Norway.

Kruchten, P. (2003). *The Rational Unified Process: An Introduction*. Boston, MA: Addison-Wesley. ISBN: 0201707101

- Lakhal, F., Dubois, H., & Rieu, D. (2013). Pattern-based Methodology for UML profiles evolution management. In *Proceedings of the IEEE 7th International Conference on Research Challenges in Information Science RCIS*. IEEE. (pp. 1-12) doi: 10.1109/RCIS.2013.6577681
- Lano., K., Bicarregui, J., & Goldsack, S. (1996). Formalising Design Patterns. In *Processing of the 1st BCS-FACS Northern Formal Methods Workshop, Electronic Workshops in Computer Science*, 1996.
- Lee, O. (2002). An Action Research report on the Korean national digital library. *Information & Management*, 39, (pp. 255-260). doi: 10.1016/S0378-7206(01)00094-5
- Lethbridge, T. C., Singer, J. & Forward, A. (2003). How software engineers use documentation: the state of the practice. *IEEE Software*, 20, (pp. 35-39). doi: <http://dx.doi.org/10.1109/MS.2003.1241364>
- López-Martínez, J., Juárez-Ramírez, R., Huertas, C., Jiménez, S. & Guerra-García, C. (2016). Problems in the Adoption of Agile-Scrum Methodologies: A Systematic Literature Review. In *4th International Conference in Software Engineering Research and Innovation, Puebla*, (pp. 141-148).
- Lyytinen, K. & Robey, D. (1999). Learning failure in information system development. *Information Systems Journal*, 9, (pp. 85–101).
- Macasaet, R., Chung, L., Garrido, J., Rodriguez, M., & Noguera, M. (2011). 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*, (pp. 50-56). ACM New York, NY, USA. doi: 10.1145/2181101.2181114
- Macasaet, R., Noguera, M., Rodriguez, M., Garrido, J., Supakkul, S., & Chung, L. (2012). 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*, Article 7, (pp. 1-6). ACM New York, NY, USA. doi: 10.1145/2424563.2424573
- Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S. & Chung, L. (2013). 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*, (pp.1-12), IEEE. ISBN: 978-1-4673-2912-5. doi: 10.1109/RCIS.2013.6577703
- Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2014). 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 5 (4)*, (pp. 71-90), IGI-Global.
- Macasaet, R. J. (2017). The Project Start Review Group. In M. Brambilla, T. Hildebrandt (eds.), *Proceedings of the Industry Track of the 15th International Conference on Business Process Management BPM*, (pp. 81-87).
- Macasaet, R.J. (2018). Just in Time Demos in the Scrum Framework. *Proceedings of the 3rd International Conference on System Reliability and Safety ICSRS*, (pp.21-24).

Macasaet, R. J., Noguera, M., Rodriguez, M. L., Garrido, J. L., Supakkul, S., & Chung, L. (2019). *Micro-business Requirements Patterns in Practice: Remote Communities in Developing Nations*. *Journal of Universal Computer Science JUCS* 25 (7), (pp. 764-787).

Mairiza, D., Zowghi, D. & Nurmuliani, N. (2010). An investigation into the notion of non-functional requirements. *Proceedings of the 2010 ACM Symposium on Applied Computing* (p./pp. 311--317), New York, NY, USA: ACM. ISBN: 978-1-60558-639-7

Medvidovic, N. & Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26, (pp. 70-93).

Mendez-Bonilla, O., Franch, X., & Quer, C. (2008) Requirements Patterns for COTS Systems. In *Proceedings of the 7th International Conference on Composition-Based Software Systems ICCBSS* (pp. 232-234). IEEE. doi: 10.1109/ICCBSS.2008.34

Mendling, J., Recker, J., & Reijers, H. (2010). On the usage of labels and icons in business process modeling. *International Journal of Information System Modeling and Design*, 1 (2), (pp. 40-58). doi: 10.4018/jismd.2010040103

Merten, T., Lauenroth, K., & Bürsner S. (2011). 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* (pp. 60-65). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-19858-8_7

Meszaros, G. & Doble, J. (1998). A pattern language for pattern writing. In Martin, Riehle and Buschmann (eds.), *Pattern Languages of Program Design 3*, (pp. 529-574). Reading, MA, Addison-Wesley

Millman, C. & El-Gohary, H. (2011). New Digital Media Marketing and Micro Business: A UK Perspective. *IJOM*, 1, (pp. 41-62). doi: 10.4018/978-1-4666-1598-4.ch076

Mishra, D. & Mishra, A. (2007). Efficient software review process for small and medium enterprises. *IET Software*, 1, (pp. 132-142).

Nikula, U., Sajeniemi, J., & Kalvianen, H. (2000). A state-of-the-practice survey on requirements engineering in small-and-medium-sized enterprises. In *Telecom Business Research Center Lappeenranta Research Report 1*, Lappeenranta University of Technology. url: <https://www.uop.edu.jo/Homeworks/13544442010.pdf>

Object Management Group, Inc. (2008). *Business Process Modeling Notation Version 1.1*. Last accessed on March 10, 2011 at <http://www.omg.org/spec/BPMN/1.1/PDF>

Object Management Group, Inc. (2009). *Unified Modeling Language Version 2.2*. Last accessed on March 10, 2011 at <http://www.omg.org/spec/UML/2.2/Superstructure/PDF/changebar>

Object Management Group, Inc. (2010). *MDA Foundation model*. Needham, MA, USA

Oliveira, B. & Belo, O. (2012). BPMN Patterns for ETL Conceptual Modelling and Validation. In L. Chen, A. Felfernig, J. Liu & Z. W. Ras (eds.), *ISMIS* (pp. 445-454), Springer. ISBN: 978-3-642-34623-1. doi: 10.1007/978-3-642-34624-8_50

Opentaps. (2013). *Opentaps*. Last accessed on October 6, 2013 at <http://www.opentaps.org/>

- Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., Lopez, L., & Gorschek, T. (2021). The state-of-practice in requirements elicitation: an extended interview study at 12 companies. *Requirements Eng* 26, (pp. 273–299). doi: 10.1007/s00766-020-00345-x
- Paludo, M., Reinehr, S. S., Malucelli, A., Bruzon, L. & Pinho, P. (2011). Applying pattern structures to document and reuse components in component-based software engineering environments. IRI (pp. 378-383), *IEEE Systems, Man, and Cybernetics Society*. ISBN: 978-1-4577-0964-7. doi: 10.1109/IRI.2011.6009577
- Peixoto, M. & Silva, C. (2018). Specifying privacy requirements with goal-oriented modeling languages. In Proceedings of the 32nd Brazilian Symposium on Software Engineering (SBES '18). ACM, New York, NY, USA, (pp. 112-121). doi: 10.1145/3266237.3266270
- Pervan, G.P. & Klass, D.J. (1992). The use and misuse of statistical methods in information systems research, in: R. Galliers (Ed.), *Information Systems Research: Issues, Methods and Practical Guidelines*, (pp. 208–229), Blackwell, Boston, MA.
- Pino, F. J., García, F. & Piattini, M. (2008). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal*, 16, (pp. 237-261). doi: 10.1007/s11219-007-9038-z
- Philippines MSME Statistics. (2011). Philippine Department of Trade and Industry. Last accessed on December 5, 2013 at <http://www.dti.gov.ph/dti/index.php?p=321>
- Pokozy-Korenblat, K., Priami, C. & Quaglia, P. (2004). Performance Analysis of a UML Micro-business Case Study. In C. Priami & P. Quaglia (eds.), *Global Computing* (pp. 107-126), Springer. ISBN: 3-540-24101-9. doi: 10.1007/978-3-540-31794-4_7
- PSRI. (2018). Pentathlon Systems Resources Incorporated. Last accessed on December 15, 2018 at <http://www.pentathlonsystems.com>
- PSRI Action Research. (2021). Action Research Material for Micro-businesses. Last accessed on December 31, 2021 at <http://www.pentathlonsystems.com/ar4mb.html>
- Quspe, A., Marques, M., Silvestre, L., Ochoa, S. F. & Robbes, R. (2010). Requirements Engineering Practices in Very Small Software Enterprises: A Diagnostic Study. In S. F. Ochoa, F. Meza, D. Mery & C. Cubillos (eds.), *SCCC*, (pp. 81-87), IEEE Computer Society.
- Ramsin, R. & Paige, R. F. (2008). Process-centered review of object oriented software development methodologies. *ACM Comput. Surv.*, 40 (1), Article 3, (pp. 1- 89)
- Real Academia Española. (2003). *Diccionario de la Lengua Española*. 22nd Edition. Espasa Calpe
- Recker, J., Indulska, M., Rosemann, M. & Green, P. (2006). How good is BPMN really? Insights from theory and practice. *ECIS 2006 Proceedings*. 135.
- Recker, J. (2008). *Understanding Process Modelling Grammar Continuance: A Study of the Consequences of Representational Capabilities*. Faculty of Information Technology, Queensland University of Technology, Brisbane.

Recker, J., zur Muehlen, M., Siau, K., Erickson, J., & Indulska, M. (2009). Measuring method complexity : UML versus BPMN. In: 15th Americas Conference on Information Systems, 6-9 August, 2009, San Francisco, California.

Recker, J. (2010). Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, Vol. 16 No. 1, (pp. 181-201), <https://doi.org/10.1108/14637151011018001>

Riehle, D. & Züllighoven, H. (1996). Understanding and Using Patterns in Software Development. *Theory and Practice of Software Systems*, 2 (1), (pp.3-13)

Reinhartz-Berger, I., Sturm, A. & Tsoury, A. (2011). Evaluating Comprehension and Utilization of Variability Aspects in UML-Based Models. In S. Nurcan (ed.), *CAiSE Forum (Selected Papers)* (pp. 156-171), Springer. ISBN: 978-3-642-29748-9. doi: 10.1007/978-3-642-29749-6_11

RE-Tools. (2013). RE-Tools. Last accessed on October 9, 2013 at https://personal.utdallas.edu/~chung/Sam_Supakkul/RE-Tools/index.html

Reijers, I., Mendling, J., & Dijkman, R.M. (2011) Human and automatic modularizations of process models to enhance their comprehension. *Information Systems*, 36, (pp. 881-897). doi: 10.1016/j.is.2011.03.003

Respect-IT. (2007). KAOS Tutorial Version 1.0. Retrieved on January 15, 2013, from <http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf>

Riaz, M. & Williams, L. (2012). Security requirements patterns: understanding the science behind the art of pattern writing. *RePa* (pp. 29-34): IEEE. ISBN: 978-1-4673-4374-9. doi: 10.1109/RePa.2012.6359977

Roost, M., Taveter, K., Rava, K., Tepandi, J., Piho, G., Kuusik, R., & Õunapuu, E. (2013). Towards Self-development of Evolutionary Information Systems: An Action Research of Business Architecture Development by Students in Socially Networked Groups. *Proceedings of the International Workshop on Approaches for Enterprise Engineering Research AppEER 2013 at the 25th International Conference on Advanced Information Systems Engineering (CAiSE)*. Vol 148. doi: 10.1007/978-3-642-38490-5_1

Ruhe, G., Eberlein, A., & Pfahl, D. (2003). Trade-off Analysis for Requirements Selection. *International Journal of Software Engineering and Knowledge Engineering*, 13 (4), (pp. 345-366). doi: 10.1142/S0218194003001378

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. (1991). *Object-Oriented Modeling and Design*.

Sabre. (2018). Sabre Incorporated. Last accessed on December 15, 2018 at <http://www.sabre.com>

Schmidt, D.C., Fayad, M., & Johnson, R.E. (1996). *Software Patterns*. Communications of the ACM, October, 1996

Schreiber, R. & Stern, P. (2001). *Using Grounded Theory in Nursing*. Springer Publishing Company, New York.

Schumacher, M., Fernandez-Buglioni, E., Hyberston, D., Buschmann, F., & Sommerlad, P. (2006). *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Limited

Scrum.org. (2018). *The Scrum Guide*. Last accessed on August 17, 2018

Segura, S., Durán, A., Troya, J., and Cortés, A.R. (2017). A Template-Based Approach to Describing Metamorphic Relations. *IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*, 2017, pp. 3-9, doi: 10.1109/MET.2017.3

Serrato-Barrera, R., Rodríguez-Gómez, G., Pérez-Sansalvador, J.C., Pomares-Hernández, S., Flores-Pulido, L., and Muñoz, A. (2020). Software system design based on patterns for Newton-type methods. *Computing* 102, pp. 1005–1030

Seruca, I. & Loucopoulos, P. (2003). Towards a systematic approach to the capture of patterns within a business domain. *Journal of Systems and Software*, 67 (1). (pp. 1-18) doi: 10.1016/S0164-1212(02)00083-3

Shaw, M., Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall

Siau, K., & Cao, Q. (2001). Unified Modeling Language: A Complexity Analysis. In *Journal of Database Management*, 12:1, (pp. 26-34)

Siau, K. & Cao, Q. (2002). How Complex Is the Unified Modeling Language?. In *Advanced Topics in Database Research*, Vol. 1 (pp. 294-306)

Siau, K. & Tan, X. (2005). Improving the Quality of Conceptual Modeling Using Cognitive Mapping Techniques. In *Data & Knowledge Engineering* 55:3, (pp. 343-365)

Siau, K. & Tian, Y. (2009). A Semiotics Analysis of UML Graphical Notations. *Requirements Engineering* 14:1, (pp. 15-26)

Smith, B. H. & Williams, L. (2012). On the Effective Use of Security Test Patterns. *SERE* (pp. 108-117), IEEE. ISBN: 978-0-7695-4742-8. doi: 10.1109/SERE.2012.23

Sochos, P., Philippow, I. & Riebisch, M. (2004). Feature-Oriented Development of Software Product Lines: Mapping Feature Models to the Architecture.. In M. Weske & P. Liggesmeyer (eds.), *Net.ObjectDays* (pp. 138-152), : Springer. ISBN: 3-540-23201-X

Solemon, B., Sahibuddin, S., & Ghani, A.A.A. (2009). Requirements engineering problems and practices in software companies: An industrial survey. *Advances in Software Engineering*, Springer, (pp. 70-77)

Sommerville I. (2004). *Software Engineering*. Addison-Wesley: Harlow, England.

Spain SME Statistics. (2011). Spanish Ministry of Industry, Energy, and Tourism. Last accessed on December 5, 2013 at http://www.ipyme.org/Publicaciones/ESTADISTICAS_PYME_N10_2011.pdf

Spain SME Statistics. (2021). Spanish Ministry of Commerce, Industry, and Tourism. Last accessed on June 30, 2021 at <http://www.ipyme.org/Publicaciones/CifrasPYME-enero2021.pdf>

- Stepan, P. & Lau, K.-K. (2012). Controller patterns for component-based reactive control software systems. In V. Grassi, R. Mirandola, N. Medvidovic & M. Larsson (eds.), CBSE (pp. 71-76). ACM. ISBN: 978-1-4503-1345-2. doi: 10.1145/2304736.2304749
- Strauss, A. & Corbin, J. (1990). *Basics of qualitative research: Grounded Theory procedures and techniques*. Sage Publications, Basics of Qualitative Research
- Supakkul, S. & Chung, L. (2009). Extending Problem Frames to deal with stakeholder problems: An Agent- and Goal-Oriented Approach.. In S. Y. Shin & S. Ossowski (eds.), SAC (pp. 389-394), ACM. ISBN: 978-1-60558-166-8
- Supakkul, S., Hill, T., Chung, L., Tun, T., & Sampaio do Prado Leite, J.C. (2010). An NFR Pattern Approach to Dealing with NFRs. In *Proceedings of the 18th IEEE International Requirements Engineering Conference RE* (pp. 179-188). IEEE. doi: 10.1109/RE.2010.31
- Supakkul, S., & Chung, L. (2012). The RE-Tools: A Multi-notational Requirements Modeling Toolkit. In *Proceedings of the 20th International Requirements Engineering Conference RE* (pp. 333-334). IEEE. doi: 10.1109/RE.2012.6345831
- Supakkul, S., Chung, L., Macasaet, R., Noguera, M., Rodriguez, M., & Garrido, J. (2013). 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*, (pp. 128-130). Last accessed on October 9, 2013 at http://ceur-ws.org/Vol-978/paper_23.pdf
- Tan, L., & Wang, N. (2010). Future internet: the internet of things. In: 2010 3rd international conference on advanced computer theory and engineering (ICACTE), vol 5, (pp. 376-380). IEEE doi: 10.1109/ICACTE.2010.5579543
- Thom, L. H., Lazarte, I. M., & Iochpe, C. (2009a). Activity patterns in process-aware information systems: basic concepts and empirical evidence. *IJBPM* 2009 4 (2), (pp. 93-110)
- Thom, L. H., Lazarte, I. M., & Iochpe, C. (2009b). On the Support of Workflow Activity Patterns in Process Modeling Tools: Purpose and Requirements. In 3rd WBPM, 2009, Brazil
- Thom, L. H., Lazarte, I. M., Iochpe, C., Priego-Roche, L.-M., Verdier, C., Chiotti, O. & Villarreal, P. D. (2011). On the Capabilities of BPMN for Workflow Activity Patterns Representation. In R. M. Dijkman, J. Hofstetter & J. Koehler (eds.), *BPMN* (pp. 172-177). Springer. ISBN: 978-3-642-25159-7. doi: 10.1007/978-3-642-25160-3_18
- United States Census Bureau. (2011). County Business Patterns. Last accessed on December 5, 2013 at <http://censtats.census.gov/cgi-bin/cbpnaic/cbpcomp.pl>
- van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. & Barros, A.P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14 (1), (pp. 5-51)
- van Gorp, J., Prehofer, C. & Bosch, J. (2010). Comparing practices for reuse in integration-oriented software product lines and large open source software projects. *Software: Practice and Experience*, 40, (pp. 285-312). doi: 10.1002/spe.955
- van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. *Fifth IEEE International Symposium on Requirements Engineering*. (pp. 249-262). doi: 10.1109/ISRE.2001.948567

- van Lamsweerde, A. (2009). *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley. ISBN: 978-0-470-01270-3
- Veerappa, V., & Harrison, R. (2013). Assessing the maturity of requirements through argumentation: A good enough approach. *Automated Software Engineering ASE*, (pp. 670-675). IEEE. doi: 10.1109/ASE.2013.6693131
- Villalón, J. A. C.-M., Agustín, G. C., Gilabert, T. S. F., de Amescua Seco, A., Sánchez, L. G. & Cota, M. P. (2002). Experiences in the Application of Software Process Improvement in SMES. *Software Quality Journal*, 10, (pp. 261-273).
- Virus. (2018). *Virus Worldwide*. Last accessed on December 15, 2018 at <http://www.virusworldwide.com>
- Wahl, T. & Sindre, G. (2005). An Analytical Evaluation of BPMN Using a Semiotic Quality Framework. In *Proceedings of the CAiSE'05 Workshops*. Volume 1, Castro, J. and E. Teniente, Eds., (pp. 533-544), FEUP, Porto, Portugal.
- Warmer, J. & Kleppe., A. (1999). *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley
- Weitlaner, D., Guettiner, A., & Kohlbacher, M. (2013). Intuitive Comprehensibility of Process Models. *S-BPM ONE 2013*, (pp. 52-71). doi: 10.1007/978-3-642-36754-0_4
- Wen, Y., Zhao, H. & 0001, L. L. (2011). Analysing security requirements patterns based on problems decomposition and composition. *RePa* (pp. 11-20), IEEE. ISBN: 978-1-4577-1020-9
- Withall, S. (2007a). *Software Requirement Patterns*. O'Reilly
- Withall, S. (2007b). *Software Requirement Patterns*. Microsoft Press
- Wohed, P., van der Aalst, W.M.P., Dumas, M. & ter Hofstede, A.H.M. (2005). Pattern-based Analysis of BPMN - An extensive evaluation of the Control-flow, the Data and the Resource Perspectives. In *BPM Center Report No. BPM-05-26*. BPMcenter.org.
- Wong, K. Y. & Aspinwall, E. (2004). Characterizing knowledge management in the small business environment. *Journal of Knowledge Management*, 8, (pp. 44-61)
- Yang, Y., Bhuta, J., Boehm, B. & Port, D. N. (2005). Value-Based Processes for COTS-Based Applications. *IEEE Software*, 22, (pp. 54-62).
- Yoshioka, N., Washizaki, H., & Maruyama, K. (2008). A survey on security patterns. *Progress in Informatics, Special Issue: The future of software engineering for security and privacy*, (pp. 13).
- Young, R. (2004). *The requirements engineering handbook*. Artech House. ISBN: 978-1-58053-266-2
- Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (2011). Modeling Strategic Relationships for Process Reengineering. In *Social Modeling for Requirements Engineering* (pp. 11-152). The MIT Press. ISBN: 9780262240550

Zhao, L., Letsholo, K., Chioasca, E., Sampaio, S., & Sampaio, P. (2012). Can business modeling bridge the gap between business and information systems? In *Proceedings of the 27th annual ACM symposium on applied computing SAC* (pp. 1723-1724). ACM New York, NY, USA. doi: 10.1145/2245276.2232054

Zhao, X., & Zou, Y. (2011). A business process-driven approach for generating software modules. *Software: Practice and Experience*, 41, (pp. 1049–1071). doi: 10.1002/spe.1068

zur Muehlen, M., & Recker, J. (2008). How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Léonard, M., and Bellahsène, Z. (Eds.) *Advanced Information Systems Engineering - CAiSE 2008*, Montpellier, France. (pp. 465-479). Springer