Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

v. 15, n. 2

mai-ago 2016

doi:10.21529/RESI.2016.1502

Sumário

Este trabalho está licenciado sob uma Licença Creative Commons Attribution 3.0 .

ISSN: 1677-3071

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente, mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.

This journal is (and has always been) electronic in order to be more environmentally friendly. Now, it is desktop edited in a single column to be easier to read on the screen. However, if you wish to print this paper, be aware that it uses Eco Sans, a printing font that reduces the amount of required ink.

PROBLEM-BASED SOFTWARE REQUIREMENTS SPECIFICATION

ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE BASEADA EM PROBLEMAS

(artigo submetido em abril de 2016)

Rafael Gorski M. Souza
Mestre pelo Programa de Pós-graduação em Computação Aplicada (PPGCA) da Universidade Tecnológica Federal do Paraná (UTFPR)
rafael@alunos.utfpr.edu.br

Paulo Cézar Stadzisz
Professor do Programa de Pós-graduação em Computação Aplicada (PPGCA) da Universidade Tecnológica Federal do Paraná (UTFPR)
stadzisz@utfpr.edu.br

ABSTRACT

Requirements specification has long been recognized as a critical activity in software development processes because of its impact on project risks when poorly performed. A large amount of studies address theoretical aspects, propositions of techniques, and recommended practices for Requirements Engineering (RE). To be successful, RE has to ensure that the specified requirements are complete and correct, what means that all intents of the stakeholders in a given business context are covered by the requirements and that no unnecessary requirements are introduced. However accurate capturing business intents of the stakeholders remains a challenge and it is a major factor of software project failures. This paper presents a novel approach referred to as "Problem-Based Software Requirements Specification" aiming at improving the quality of the software requirements specification in the sense that the stated requirements provide suitable answers to real customers' business issues. In this approach, knowledge about the software requirements is constructed from the knowledge about the customer´s problems. Problem-Based Software Requirements Specification consists in an organization of activities and outcome objects through a process that contains five main steps. It aims at supporting the software requirements engineering team to systematically analyze the business context and specify the software requirements, taking also into account a first glance and vision of the software. In addition to examples, a CRM software case study is presented and discussed.

Key-words: requirements engineering; software requirement specification; customer problem.

RESUMO

A especificação de requisitos já é reconhecida há muito tempo como uma atividade crítica para processos de desenvolvimento de software devido ao seu impacto sobre os riscos de um projeto quando é mal-feita. Há inúmeros trabalhos que tratam de aspectos teóricos, proposição te técnicas e recomendação de práticas para a engenharia de requisitos (ER). Para que se obtenha sucesso, a ER precisa garantir que os requisitos especificados para um projeto estejam completos e corretos, o que significa que todas as intenções dos *stakeholders* em um determinado contexto de negócios sejam cobertas pelos requisitos e que nenhum requisito desnecessário seja inadvertidamente introduzido. Contudo, a captura precisa das intenções de negócio dos *stakeholders* continua desafiadora e representa o principal fator de fracasso de projetos de desenvolvimento de software. Este artigo apresenta uma nova abordagem chamada "Especificação de Requisitos de Software Baseada no Problema", a qual se propõe a melhorar a qualidade da especificação dos requisitos de software garantindo que os requisitos estabelecidos representem respostas adequadas a questões de negócios reais dos clientes. Nessa abordagem, o conhecimento dos requisitos de software é construído a partir do conhecimento sobre os problemas dos clientes. A Especificação de Requisitos de Software Baseada no Problema consiste na organização das atividades e objetos resultantes por meio de um processo composto de cinco passos. Ela se propõe a ajudar a equipe de engenharia de requisitos de software a analisar sistematicamente o contexto de negócios para especificar os requisitos de software, também levando em conta uma primeira impressão e visão do software. Além de proporcionar exemplos dessa abordagem, o artigo apresenta e discute um caso de estudo envolvendo um software de CRM.

Key-words: requirements engineering; software requirement specification; customer problem.

1 INTRODUCTION

Software is an important part of modern systems and is employed widely in almost all economic areas. Countries use software to control critical resources, including energy, defense, security, and water. Industries make use of computers and software to model and supervising their processes. The telecommunications industry often builds new services based on software. The entertainment industry (*e.g.*, games and movies) makes intense use of software and can reach millions of users. Equally, many other areas are also benefiting from new software systems pushing this market in a continuous growth.

The growing complexity to produce software to answer the increasing market demands requires professional approaches to specify and to construct software systems. Software engineering is the knowledge field to address this demand as it provides systematic, disciplined, quantifiable methods and tools for the development, operation, and maintenance of software (BOURQUE AND FAIRLEY, 2014). Software engineering helps developers to apply scientific and technological knowledge to encompass all aspects of the software production. The Guide of Software Engineering Body of Knowledge (SWEBOK) from the Institute of Electrical and Electronics Engineers (IEEE) provides a consistent view of software engineering and it is an important guide to consider in any phase of the software life cycle (BOURQUE AND FAIRLEY, 2014).

Requirements Engineering (RE) is the first knowledge area within the SWEBOK. It initiates the software life cycle and defines software intents and constraints. RE explores sources (*e.g.*, stakeholders, documents, regulations) and applies elicitation techniques, analysis, and documentation. There are numerous sources for software requirements that may include business goals, domain knowledge, stakeholder's viewpoints, business processes, and environmental aspects. A software analyst needs to ensure that the most important business issues, from different sources, are addressed. The elicitation activity allows the software analyst to understand the problem the software is required to solve. There is a range of elicitation techniques that vary from traditional interviews for requirements gathering to observation or scenario/user story approaches. Independently of the used elicitation approach, an analysis is needed to ensure the consistency of the information. The requirement documentation (often referred to as Software Requirements Specification Document - SRS) plays an important role establishing the basis for the agreement between the parties on what the software is intended to do and what if is not expected to do (BOURQUE AND FAIRLEY, 2014).

Software requirements engineering is a recognized critical area in software engineering because of its impact on project risks when poorly performed. Very often, there is a gap between the software delivered and what was expected by the stakeholders. "Accurately capturing system requirements is the major factor in the failure of 90% of large software projects" (DAVIS *et al.*, 2006).

A significant part of the difficulties for software requirements specification is due to the intricacy to write precise sentences that specify the requirements ensuring the characteristics of a good SRS (IEEE, 1998; ISO, 2011). However, the main difficulties to specify software requirements come from a much more complex matter. The requirements have to accurately capture the business intents of the stakeholders. This way, software analysts should understand concepts, dependencies, business goals, and processes from the stakeholders' domain in order to be able to write suitable requirements. This understanding is often not easy to reach because of the existing distance between the technical domain where software is specified and the business or customer domain where problems arise (BOURQUE AND FAIRLEY, 2014).

Davey and Parker (2015) summarize the requirements elicitation problems in nine categories. One of them is "Clients will sometimes ask for requirements that the organization does not need". This category of problem clearly represents a misunderstanding about the dependency of requirements on the real needs of the customer. Another problem category is "The client cannot say what the business needs". Similarly, this category of problem represents a misunderstanding about the dependency of needs on the customer´s problems. Lack of success in recognizing the Customer Problems (CPs) and Customer Needs (CNs) as different elements in software requirements' specification suggests an opportunity of extension in the current studies on the subject. This paper proposes a method for software requirements' specification referred to as Problem-Based SRS. This method emphasizes the dependency of the requirements specification on customer needs and also the dependency of these needs on customer problems. In addition, this paper presents a novel definition of customer problems and customer needs to address the business aspects that motivate the development of a software solution.

This paper is organized as follows. Section 2 briefly presents related works including RE approaches and business aspects of requirements specification. Section 3 presents key concepts and processes of the Problem-Based SRS approach. Section 4 demonstrates a case study applying the proposed approach. Finally, Section 5 discusses the results and Section 6 presents the conclusions and future work possibilities.

2 RELATED WORK

Requirements specification has long been recognized as critical activity in software development processes. A large amount of studies address theoretical aspects and propositions of techniques and recommended practices for RE (DANEVA *et al.*, 2014). Studies like that conducted by Hofmann and Lehner (2001) identified RE practices that clearly contribute to software project success and concluded that successful projects allocate a significantly higher amount of resources to

RE. However, most studies have focused on requirements elicitation, modeling, and processes/methods (VALASKI *et al.*, 2014).

A number of studies reported in the literature focus on the elicitation of software requirements. According to Zowghi and Coulin (2005), this activity is the process of seeking, uncovering, acquiring, and elaborating requirements for software systems. It concerns learning and understanding the needs of customer with the aim to communicate these needs to software developers (ZOWGUI AND COULIN, 2005). Requirements elicitation remains essentially a human-centered activity in which effective communication among the various stakeholders is a primary principle (BOURQUE AND FAIRLEY, 2014). Additionally, according to Davey and Parker (2015), there is a general agreement that fixing the results of poor requirements elicitation is more expensive than fixing other mistakes.

It is common sense in most studies that software requirements have to satisfy the customer´s (*i.e.*, stakeholders) intents. IEEE 24765 standard uses the expression “in a way that is [the requirement] acceptable to the customer” (ISO, 2010) to emphasize that the requirements must achieve the customer´s intentions. Thus, some authors propose validating specified requirements by determining their conformity with stakeholders´ needs (HOFMANN AND LEHNER, 2001). The term “need” is often employed to refer to the cause or reason that justifies the specified software requirements (ZOWGUI AND COULIN, 2005). The needs would be the sources of the requirements. Moreover, some authors observe that the stakeholders’ needs placed on a software product contribute to the solution of some real-world problem (BOURQUE AND FAIRLEY, 2014). Indeed, a software solution developed according to a set of specified requirements should solve the related customer´s problems (LEFFINGWELL AND WIDRIG, 2003).

Among the requirement specification techniques, the agent- or goal-oriented approaches (*e.g.*, GORE - Goal Oriented Requirement Engineering) have been used in software engineering to model early requirements and non-functional requirements (GIORGINI *et al.*, 2002) by means of goals. Goals are prescriptive statements of intent to be satisfied by a software system. Goal analysis involves decomposing goals into sub-goals by means of an AND- or OR-structure (VAN LAMSWEERDE, 2004). Common methods of GORE are KAOS, I*, and NFR (YU *et al.*, 2011). The KAOS (Knowledge Acquisition in Automated Specification or Keep All Objects Satisfied) method is the richest in formal analysis techniques and enables the system analyst to build the goal models for further derivation of the SRS. In a KAOS model, the stakeholder’s needs represent the goals, responsibilities, objects, and operations of the intended software system (DARDENNE *et al.*, 1993).

With respect to the understanding of real-world problems, the Problem Frames (PF) approach presents a new view that relates problems, specification, and software requirements. The “problem” in PF is defined as a software related problem viewed as a requirement in a real-world context which a software can help to solve. PF has the purpose of meeting a recognized need (i.e., requirement) by transforming the physical world. The part of the world to be transformed, in which requirements are located, is called the problem world. The problem-progression is the main activity in PF and it creates the specifications based on requirements (JACKSON, 1999). Problem frames can be regarded as defining problem classes in software engineering (JACKSON, 2005).

In the field of business analysis, the Guide to the Business Analysis Body of Knowledge (BABOK) refers to problems and needs using the follow terms: (i) Problem statement describes the problems in the current state and clarify what a successful solution will look like, (ii) Business need is a statement of a business objective or an impact the solution should have on its environment. According to IIBA (2009) the process "Define Business Needs" considers the widest possible range of alternative solutions by evaluating the encountered problems (IIBA, 2009).

In the context of system engineering, the deep understanding of system intents and how they map to technical systems requirements is as important as for software engineering, and the underlying concepts are also analogous. The International Council of System Engineering – INCOSE - employs the terms "Problem or Opportunity" to refer to the issues underlying the gaps in the organization strategy with respect to the desired organization goals or objectives (INCOSE, 2015). System engineering also uses the term "stakeholder needs" to mean the "needs determined from the communication with external and internal stakeholders in understanding their expectations, needs, requirements, values, problems, issues, perceived risks and opportunities". The term "stakeholder requirements" refers to "requirements from various stakeholders that will govern the project including required system capability functions and/or services; quality standards; systems constraints; and cost and schedule constraints". Additionally, the term "system requirements" means "what the system needs to do, how well and under what conditions, as required to meet the project and design constraints" (INCOSE, 2015).

Specifically, in the corporate area, studies on business modeling intend to understand and model how businesses function and how a company creates value. When the business model is transmitted to a company´s information systems layer, a certain abstraction is required to accommodate its strategic orientations. A unified view to simplify the business model and to better map the business values and customers is important to express the needs of the stakeholders. Osterwalder and Pigneur (2003) proposed an approach for business modeling referred to as "e-business model" and presented a more recent version called Value Proposition Design (VPD) (OSTERWALDER *et al.*, 2014). Similarly, Blank and Dorf (2012) described the study of customer problems as the understanding of the problems that impact the organization and the "intensity of pain" they cause to the customer. In other words, how customers experience the problems and why the latter matter to the former. This information brings compelling arguments to build solutions to solve problems. Blank and Dorf (2012) suggest that customer problems should be expressed as “latent, passive, active or vision” depending on the customer acknowledgement and motivation to solve them. According to VPD, Osterwalder *et al*. (2014) divided the customer profile in three sections: pains, gains, and tasks. Pains are the weak results, risks, and obstacles related to the customer tasks. Gains are related to the results expected by the customer and the concrete intended benefits. Tasks detail the initiatives and work done to achieve the results in daily businesses.

Despite the large number of studies and approaches proposed to specify software requirements, it is generally accepted that specifying requirements remains an unclear and challenging task. Terms used in the scientific literature and even in the industry (as pointed out in this section) are not convergent, creating lack of understanding on the subject. Software analysts may feel confused when trying gathering information from the stakeholders and other sources in the business or application domain. Therefore, the specification of software requirements may become incomplete or incorrect because of an inadequate understanding of the project intents. However, the related works indicated in this section (among other contributions) bring fundamental concepts that, in addition to their contribution, will be useful to elaborate a new proposition referred here to as Problem-Based SRS, as discussed in the next section.

3 PROPOSAL OF PROBLEM-BASED SRS

This section introduces the Problem-Based SRS approach. It presents an overview and the key concepts of the approach, followed by a general view of its activities and produced outcomes. Also, each activity is depicted describing the goal and format of the related specifications.

3.1 OVERVIEW OF THE PROPOSED APPROACH

The Problem-Based SRS is a novel approach proposed here aiming at improving the quality of software requirements specification in the sense that the stated requirements provide suitable answers to real customer´s business issues. In this approach, knowledge about software requirements (*i.e*., requirements specification) is constructed based on the knowledge about the customer´s problems. However, in this paper, the concept of customer´s problem has a new definition that extends its usual understanding. Instead of expressing problems as purposes (INCOSE, 2015), customer’s needs (BOURQUE AND FAIRLEY, 2014) or goals (YU *et al*., 2011), authors propose a more accurate technique to understand and state the customer´s problems. As discussed in section 3.2, problems are considered as “obligations or expectations” with respect to matters that may affect the customer´s business.

Although the concept of *problem* in the proposed approach is somehow similar to that in the related works, authors argue that the concept of problem in the domain of software systems has a subtle meaning, not unveiled previously. Problems are different from “purposes”, since “purposes” represent objectives or intents, not the original difficulties that constitute the problems. Similarly, problems are not “needs”, because “needs” represent the necessities directed to the software solution and not the problem (*i.e.*, the difficulty) that calls for a solution. Also, problems are not equivalent to “goals”, as “goals” describe early software requirements.

By analyzing the customer´s business context, problems can be defined and their severity can be specified. According to the perceived intensity of the problems, the customer becomes motivated to solve all or a subset of them (BLANK AND DORF, 2012). At this point, authors understand that the customer envisions possible types of solutions for the problems. Solutions may be technical, as software systems, or they may take any other form, as for example, hiring a new employee.

In this paper, the customer´s early view of a software system solution is referred to as “software glance”. The software glance does not contain much details about the software. However, it may describe major characteristics and interfaces perceived by the software analysts considering a set of identified customer problems. Software glance represents (in an early stage) the solution that will provide what is needed to solve the customer problems. Therefore, based on the customer´s problems, and taking into account the software glance, customer´s needs for that software could be defined. As stated in the next section, customer´s needs are defined as the outcomes that a software shall provide to solve (or help solving) the customer´s problems. Because, the customer´s needs detail what is expected as an outcome from the envisioned software solution, they allow the specification of the software glance to evolve and turn to a more detailed specification, called “software vision” in this paper. Then, the software vision, together with the customer´s needs, represent the input for the software requirements specification. This means that software requirements derive from the needs that in turn derive from the customer´s problems, as illustrated in Figure 1.

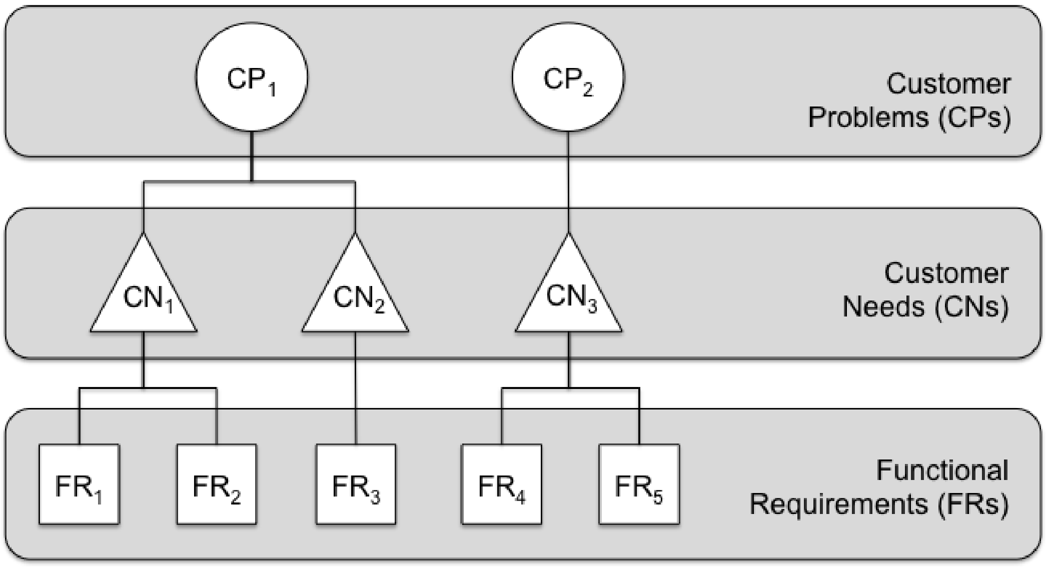

Figure 1. Causal dependencies between problems, needs, and requirements.

Source: the authors

3.2 KEY CONCEPTS OF THE PROBLEM-BASED SRS

The four key concepts of Problem-Based SRS are “customer problems”, “software glance”, “customer needs”, and “software vision”. Although these are not entirely new concepts, they have particular meanings in the proposed approach. They are used along with other common concepts such as “business context” and “software requirements specification”.

(i) Customer Problems - CPs

According to the dictionary, a “problem” is a question that involves doubts, uncertainty or difficulty (MCKEAN, 2005). In a math sense, a problem is a proposition requiring a solution. Similarly, in engineering, engineers seek for solutions for the problems of a company or society. In management, problems represent pains or risks with respect to negative results of carrying out business tasks (OSTERWALDER *et al.*, 2014). Through this paper, authors consider that a company becomes motivated to buy or develop a solution when a problem that affects its business is present. Thus, problems represent the reasons why companies attempt to find new solutions including software systems.

Problem is defined herein as a matter that may cause or is causing an (negative) effect on the company´s businesses. One can perceive problems by means of indicators, measures, and metrics that show how healthy a company or business is. Because many problems are not explicit, one can also perceive problems as (negative) feelings. In the business domain, these perceptions of problems may take the form of business risks, business losses, and business improvements, for instance. In general, problems are considered bad things for a company. However, in many ways, problems may also arise from the company´s intent to improve its businesses by changing its goals, following news strategies or adopting innovative technologies, for instance. These are considered problems because they require efforts (*i.e.* bring difficulties) to be achieved (*i.e.* to be solved).

In this paper, from the software analyst´s point of view, the problems of the business stakeholders are referred to as Customer Problems (CPs).

The CPs impact business in different forms, according to the severity of the problem. To illustrate CPs, consider the business context (*i.e.*, a given business scope) and further list of perceived CPs, presented in Table 1. CPs represent examples of risks and losses perceived in a business context in which a company intends to adapt its automobile model. The underlined verbs in each CP highlight the intensity of the problems to the company.

Table 1. Example of Customer Problems

Business Context Definition: *The company produces and commercializes an automobile model worldwide. This automaker intents to adapt its automobile model to the new emission regulations. Such regulations concern the business because they may cause severe financial penalties and public image degradation if they are not met by the automobile model. The company shall construct technical solutions to ensure compliance with emission regulations without compromising other technical aspects of the automobile.*
Customer Problems Definition: CP.1 - The company´s current automobile model is 80% compliant with the recent emission regulations. The company must adapt it to fully comply with the recent emission regulations; otherwise it will suffer financial penalties. CP.2. The company must deliver the automobile model adapted to the emissions regulation before the competition, otherwise it will lose an important marketing opportunity of this innovative upgrade. CP.3. The company must ensure that the changes in the automobile model comply with the emissions regulations do not affect other technical aspects of the automobile model. CP.4. Customers of the company´s automobile model expect that the model is one of the best in terms of reducing emission. If that does not happen, the attractiveness of the automobile model will decrease. CP.5. Customers of the company´s automobile model aim at contributing to reduce the greenhouse effect and they hope the automobile model will encourage other people to embrace the same cause, otherwise they will be somehow disappointed.

Source: the authors

(ii) Software Glance - SG

Once the customer problems are established, the software analyst elaborates a rough idea of a software to solve or help to solve those problems. This early idea provides a solution direction in wide terms and allows to identify the main pieces of the envisaged software. For instance, the idea might state that “an information system including a database, interface to collect data from Internet sources, and being operated by a professional" could be a rough solution to solve the set of specified problems. Here authors refer to this "rough idea" as **software glance**.

The software glace will be useful for further analysis and synthesis (*e.g.*, software vision definition and customer´s need specification) of the software solution. Although the software glace does not provide details about the software architecture, functionalities, and behavior, it represents a high level and early definition of the software solution.

(iii) Customer Needs - CNs

Because, by definition, customer problems cause some kind of discomfort, pain, possibility of losses, or expectation, the customer (*i.e.*, who suffers the problem) is motivated to seek a solution. At this early stage, the company envisages a solution (a software glance) and starts defining what is necessary for that solution to solve or to minimize the perceived problem. These necessities are referred to as Customer Needs (CNs).

According to Leffingwell and Widrig (2003), the customer needs (also referred to as stakeholder needs) are part of the problem domain and are defined by real users. These real users talk in non-technical terms and express their needs to solve their problems using statements such as: I need a new logistics system to optimize my costs and smash the competition prices. These are simple descriptions, in natural language, that tell what a software should provide to address the problem.

It is important to note that customer needs do not describes functionalities of the software to be developed. Instead, it describes what the software shall provide to the customer in order to solve his problems. They are focused on the outcomes of the software, as, for instance, the information it can provide.

As an example, let us consider that a manager has to pay invoices that are due on a given day, otherwise he will have a payment penalty. A software solution could help to solve this problem by providing information that warns the manager about the invoices due dates. Consequently, the customer need according to the stated problem is "the manager needs to be aware about the invoices due dates by means of information provided by the software solution".

Table 2 presents some examples of customer needs related to the customer problem CP.4 introduced in Table 1.

Table 2. Example of Customer Needs

Customer Needs Definition:
CN.1. The company needs a software system (a web site) to allow its clients to know about (to be aware of) the emission specifications related to the company´s automobile model.
CN.2. The company wishes a software system (a web site) to allow its clients to know (to be aware of) how substantial the emissions related to the company´s automobile model are with respect to other concurrent products.
CN.3. The company aims a software system (a web site) to allow its clients to know about (to be convinced of) the opinions of other clients with respect to the emissions related to the company´s automobile model.

Source: the authors

(iv) Software Vision – SV

A vision document for a software project is a common document used to describe a high-level view of the software to be developed. It provides an overview of the aimed software, its major features, its scope and limitations, positioning, environments, and involved stakeholders, among other general information. As an example, the IBM Rational Unified Process (RUP) describes such information in a vision document (JACOBSON *et al.*, 1999; IBM, 2015) and presents a template to write this document. Similarly, other authors proposed to use analogous vision documents as part of the software requirements engineering process (LEFFINGWELL AND WIDRIG, 2003; WIEGERS AND BEATTY, 2013).

The vision document is a technical view of the software that is useful to support the software requirements specification. It provides boundaries for the software that helps to keep the requirements inside that software scope. The vision document is also a more detailed view of the aimed software outlined in the software glance. It includes some very high-level decisions with respect to a draft of the software organization and connections with the environment. Even presenting some aspects of architecture, this document is not intended to describe the complete software architecture because the architecture will be developed later in the software development process during the design phase.

3.3 GENERAL VIEW OF THE PROBLEM-BASED SRS

The proposed Problem-Based SRS approach involves the organization of activities and outcome objects. It aims at supporting the software requirements engineering team to systematically analyze the business context and specify the software requirements. Figure 2 shows an Object-Process Diagram (OPD) that outlines the Problem-Based SRS. The OPD notation is defined in the Object-Process Methodology (OPM) (DORI, 2011).

Problem-Based SRS consists in a process that contains five steps also referred to as processes or sub-processes, as illustrated in Figure 2. The first step is the process to specify customer problems based on the analysis of the Business Context and producing the list of Customer Problems as outcome. The second step it the process to design a Software Glance suitable for the defined customer problems. Next, in the third step, customer needs are specified according to the Customer Problems and the Software Glance. Fourth, the software vision is produced enhancing the software view expressed in the Software Glance and considering the specified Customer Needs. Finally, step 5 involves the process of specifying the software requirements based on the Customer Needs and according to the defined vision of the software system.

Problem-Based SRS approach may be conducted sequentially as well as following iterative and incremental work flows. Although the OPD in Figure 2 suggests a sequential flow of processes from the first to last step, one should consider that these processes do not need to be sequentially synchronized. This means, for instance, that the second step can start even if the first step is not entirely concluded. This way Problem-Based SRS may clearly be employed along with agile approaches.

The following subsections explain the processes that encompass the Problem-Based SRS. The relationship between the processes and their inputs and generated outcomes are detailed for each specific process.

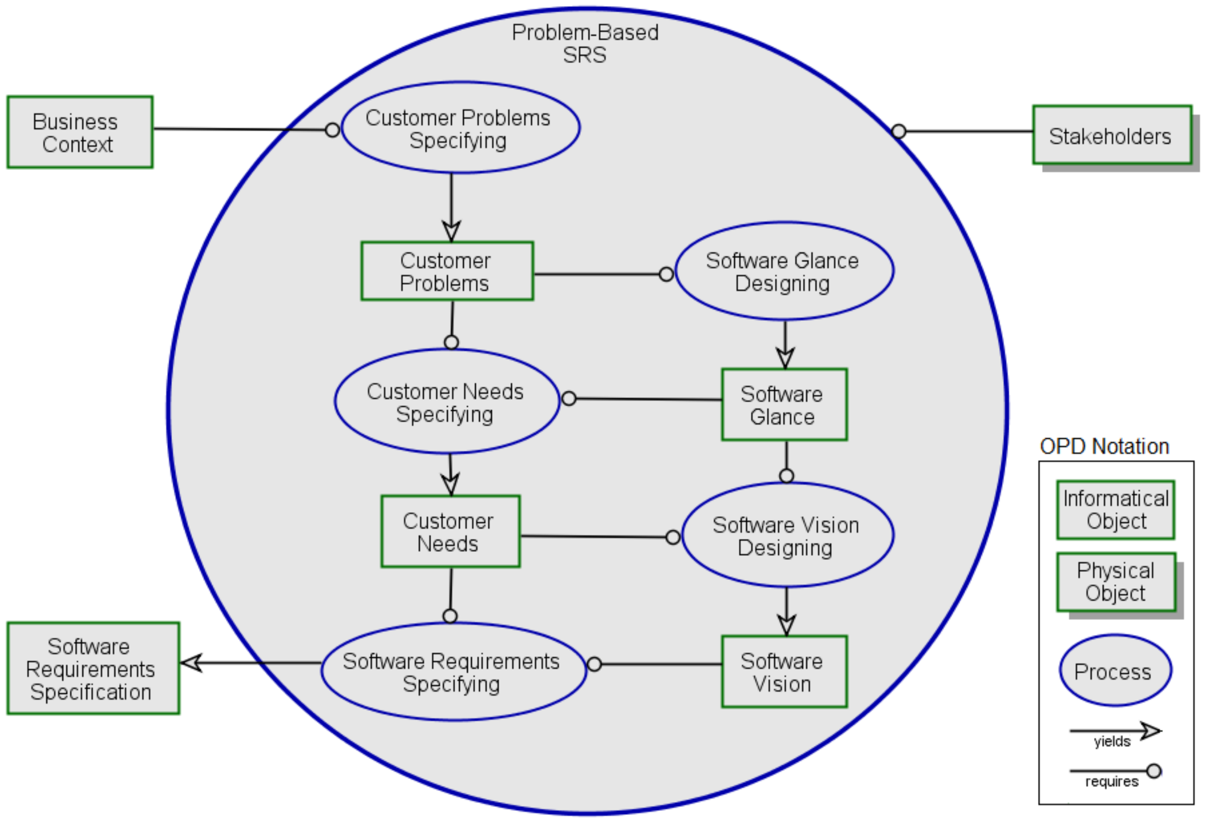

Figure 2. Problem-Based SRS Process Diagram.

Source: the authors

3.3.1 CUSTOMER PROBLEM SPECIFYING

The *Customer Problem Specifying* process aims at studying the business context to infer CPs and to write them in the form of statements. To determine the CPs, the software analyst may explore the business context. One can make use of questions such as: What problem is self-evident in the business context? Can specific problems, involving some pain or penalty be observed? How does the company notice difficulties? Are there metrics or indexes pointing out the current difficulties? Are there envisaged actions to mitigate current difficulties? Are there any felling from the stakeholders about the difficulties in the business context? What is the level of pain the company is experiencing? Is there any imminent risk? (OSTERWALDER *et al.*, 2014).

Authors suggest a new notation to write the CPs statements. Each CP is written as a sentence in natural language and should include a noun, a verb, and object, and a penalty. The *noun* states the subject that suffers or feels the problem (*e.g.* the company, a stakeholder, a customer). The *verb* indicates the intensity and the form a problem reaches the subject. For instance, the problem may be an "obligation" indicating it represents a severe problem to the subject. The problem *object* describes the difficulty that is the source of the problem. Finally, the *penalty* associated to the problem states the cost, pain, punishment or gain that is a consequence of the problem object. Problem penalty is optional in the problem sentences because sometimes it is evident.

With respect to the verb in the definition of a problem, authors propose three alternative classes as standards to be used in order to write the CPs, as follows.

- Obligation class

This class indicates that the *object* of the problem is an obligation to the *subject*. This means that the *subjec*t must accomplish the *object* because a penalty is otherwise imposed. This is the more severe class of problem as the customer has no choice but doing what is imposed. Verbs such as "must to", "have to" or "is obligated to" can be used to express this class of problems. This obliged may result from the legislation, the *subject´*s client, or from legal or regulatory standards, for instance.

- Expectation class

This class indicates that there is an expectation (from the *subject´*s clients) to the *subject* about the *object*. An expectation is not as severe as an obligation, however it represents a strong belief that the *object* could be accomplished. Consequently, the *subject* should seriously consider the *penalty* because the level of expectation can be significant for a certain stakeholder (*e.g.*, a *subject´*s client).

- Hope class

This class indicates that there is hope (from the point of view of the *subject´*s client) to the *subject* about the *object*. In other words, a hope is a prospect or possibility that the *subject* will accomplish the *object*. This is the least severe of the customer problem classes, however it deserves attention because a hope could influence a certain stakeholder (*e.g.*, *subject´s* client). Table 3 shows the proposed notation and examples of problem statements.

Table 3. Example of CP Syntax

[Noun] [Verb] [Object] [Penalty]
Example 1: The Company **[Noun]** must **[Verb]** submit a report within ten days of the event **[Object]** otherwise a punishment of 10% of the closing balance applies **[Penalty]**.
Example 2: The Company´s **[Noun]** client has an expectation **[Verb]** that the waiting time of a call center is less than two minutes **[Object]** otherwise one may make an official complain to regulatory authorities **[Penalty]**.
Example 3: The Company **[Noun]** has a hope **[Verb]** that a sale order can be canceled even after confirmation **[Object]** otherwise the client may feel unhappy **[Penalty]**.

Source: the authors

3.3.2 SOFTWARE GLANCE DESIGNING

The *Software Glance Designing* is the process through which the software analyst builds the very first draft of the software solution. The goal of this activity is to provide an early understanding of the system border, high-level system architecture, main software interfaces and software actors. During this activity, the software analyst can use diagrams as a resource to visually represent the software glance. Figure 3 presents a simple example of a software glance illustrated in the form of a block diagram. Additionally, the software glance can be documented in the form of sentences (as illustrated in Table 4) and complementary explanations.

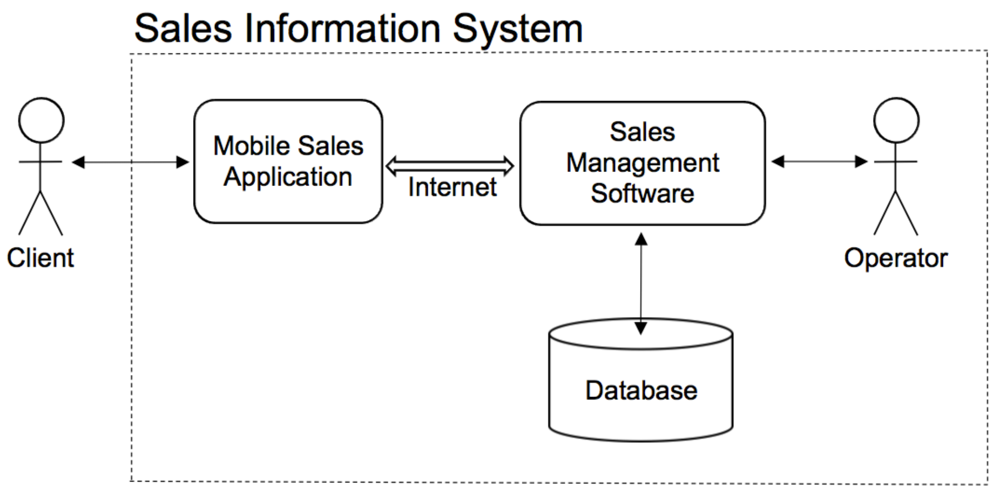

Figure 3. Example of block diagram describing a software glance

Source: authors

Table 4. Example of Software Glance

Software Glance Definition:
The sales information system will support purchasing products by clients using mobile platforms. The mobile application will provide customers with products information and provide the sales management software with purchase data. The system operator will interact with the management software for analyzing the sales progression and possible feedbacks from clients. All information generated shall be persisted in a local database. The communication between sales management software and the mobile application will be over the Internet.

Source: the authors

3.3.3 CUSTOMER NEEDS SPECIFYING

The *Customer Needs Specifying* process identifies and maps the CNs to satisfy the CPs using the software glance as to guideline. Each need statement is the result of direct mapping from a specific problem, that is, the problem represents the origin of that particular need. The software glance creates boundaries and gives direction to choose the appropriate CNs. Each CP creates a demand for CNs to solve or mitigate the problem.

Authors consider that a software solution can provide basically four classes of outcomes: information, control, construction and entertainment. Thus, CNs are expressed using these outcomes, as discussed as follows:

- *Information* is a common outcome from a large number of software systems in different forms, as for example: printed reports, data tables and graphs on screen, alarm sounds, warning messages, and light signs. In this manner, a CN may state that a customer needs to know, to be aware, to have knowledge, or to be reminded with respect of some information.
- Software systems also provide *control* that means a continuous supervision of a running logic. For instance, an astronaut needs control of the pressure inside the spacecraft cabin provided by a software solution.
- More and more, software systems provide us with means to *build* digital things, as for example: texts, models, drafts, images and videos. Therefore, building or modeling things are software solutions for a number of CNs.
- Finally, a software system can provide *entertainment* as its outcome in the form of music, video and games, for instance.

Authors suggest a new notation to write the need statements using natural language and including a *noun*, a *verb*, a *means*, an *object*, and a *condition*. The *noun* states the subject (*e.g.* the company, a manager, a customer) that has the necessity. The *verb* indicates a necessity from a problem using verbs such as to want, to intend, to aim, to desire or to need. The *means* defines the thing that will provide the *object* of the need.

In the context of this study, the *means* will always be the software system under design (outlined in the software glance). The *object* represents the intent of the need, that is, what the *means* should provide in terms of information, control, construction or entertainment. The *condition* states constraints on the *object* of the need, as the period or accuracy of the *object*, for instance.

Table 5 shows the proposed notation and gives some examples of CNs using this notation.

Table 5. Example of Software Glance

[Noun][Verb][Means][Object][Condition]
Example 1: The manager **[Noun]** needs **[Verb]** a software system **[Means]** to know the balance of the client's accounts **[Object]** every quarter **[Condition]**.
Example 2: The human resources director **[Noun]** wants **[Verb]** a software application **[Means]** to be aware about the employee absenteeism **[Object]** every month **[Condition]**.

Source: the authors

3.3.4 SOFTWARE VISION DESIGNING

Through the *Software Vision Designing* process the solution developer and the stakeholders aim to reach agreement about the high-level scope and position of the software. The software glance provides the starting point and the initial draft with initial directives the software. This process produces the vision document, as discussed in section 3.1, and it involves common activities (LEFFINGWELL AND WIDRIG, 2003; WIEGERS AND BEATTY, 2013) as, for instance, defining the positioning of the software solution, detailing the stakeholders, describing the product overview and its high level features, and defining the environment and constraints.

3.3.5 SOFTWARE REQUIREMENTS SPECIFYING

The *Software Requirements Specifying* is the process of clearly identifying the functional and nonfunctional demands that the software shall meet. This process is related to a well-known knowledge area (BOURQUE AND FAIRLEY, 2014) concerned with the elicitation, analysis, specification, and validation of software requirements. It also includes the requirements management during the software life cycle.

The software requirements must meet customer needs and they must have accurate characteristics as stated in the ISO/IEC/IEEE 29148 (2011). Particularly, they shall be complete (*i.e.*, all customer needs must be met by the requirements) and correct (*i.e.*, all requirements must meet some customer need).

ISO/IEC/IEEE 29148 (2011) provides a syntax to write software requirement. Authors adopt that syntax but suggest to write the requirement´s conditions at the end of the sentence in order to always start the sentences with the subject, as illustrated in Table 6.

Table 6. Requirements Specification Notation

[Subject] [Verb] [Object] [Constraint] [Condition]
Example 1: The system [**Subject**] shall set [**Verb**] the signal x received bit [**Object**] within 2 seconds [**Constraint**], when signal x is received [**Condition**].
Example 2: The Radar System [**Subject**] shall detect [**Verb**] targets [**Object**] at ranges out to 100 nautical miles [**Constrain**t] at sea state 1 [**Condition**].
Example 3: The Invoice System [**Subject**] shall display [**Verb**] pending customer invoices [**Object**] in the order invoices are to be paid [**Condition**].

Source: the authors

4 THE CRM SOFTWARE SYSTEM EXAMPLE

This section presents an example of software requirements specification using the proposed Problem-Based SRS approach. The example under consideration refers is a Customer Relationship Management (CRM) software. The following subsections are organized according to the activities of the process defined in section 3.2 and briefly describe how the software analyst can conduct them.

4.1 BUSINESS CONTEXT

Business context is a set of information that characterizes a portion of a business domain and defines the scope for conducting some action, as an analysis, change or improvement. A business context can be described using different modeling approaches like context diagrams, feature trees (WIEGERS AND BEATTY, 2013), natural language sentences, UML, or ontology-based models (NOVAKOVIC AND HUEMER, 2014).

For the CRM example, the business context was defined in the form of a statement in natural language illustrated in Table 7.

Table 7. Business Context – CRM Example

A company has strong difficulties to effectively build relationships with its clients and it is convinced that an information system such a CRM can contribute to reduce these difficulties.

Source: the authors

This short business context statement points out that the company is encountering “*strong difficulties*”, what indicates that problems may exist in this business scope and that they are significant to the business. This discomfort in face of the difficulties leads the company to search for a software solution to help to reduce these difficulties (*i.e.*, to solve the underlying problems). Also, this discomfort relates to issues about the "*relationship with its clients"*. Finally, the stakeholders know that CRM is a suitable type of solution, perhaps because they have saw other companies using such technology.

4.2 CUSTOMER PROBLEMS

The software analyst starts conducting the process "Customer Problem Specifying" performing an analysis of the business context to determine the customer problems related to the CRM software. Using the business context, the analyst can identify the source of problems that, for the considered example, is "the difficulties of relationship with its clients". Different elicitation techniques can be used in order to determine the customer problems based on the problem source. Particularly, when applying the Problem-Based SRS approach, the focus is on analyzing the company obligations, expectations, hopes, and associated penalties.

For the considered example, the analyst could for instance, ask the stakeholders "How does the company currently supports customer relationships?", "Are there statistics about the clients' feedbacks?", "How does the company fulfill clients' expectations?", "Are there company´s policies about customer relationship?", "Are there particular regulations with respect the customer relationship?". This kind of questions can help unfolding the problems associated to the business context. Table 8 presents some examples of CPs for the CRM specification.

Table 8. CPs for CRM software.

CP.1 - The company must ensure the existence of a communication channel with all clients. Otherwise, loss of contact with customers will occur affecting marketing, promotions, feedback, and future sales.
CP.2 - The company must consider the statistics about customers' feedback in the company planning. Otherwise, provoking customer dissatisfaction and consequent losses in sales and market-share.
CP.3 - Clients have the expectation that the company addresses theirs feedback. Otherwise, the customers may become frustrated and the company may suffer a penalty of decrease of reputation.
CP.4 - The company must align its sales strategies and campaigns with customers' behavior. Otherwise, the company risks missing sales opportunities or failing with its strategies.
CP.5 - The company must forecast sales. Otherwise, the company may miss sales opportunities and make wrong provisions.

Source: the authors

4.3 SOFTWARE GLANCE FOR THE CRM

The software glance for the CRM software must clarify the early view of the software to face the identified problems (CPs). It is the very first representation of the software solution. In this example, the business context clearly suggests a CRM software as an envisaged solution by the stakeholders. However, CRM software does not follow a standard. It may to be customized in different ways. Because CP.1 pointed out a problem associated to communication channels with the company´s clients, the software analyst considered that the software solution could have a web interface with the clients for marketing, promotions and to receive and respond to feedbacks (highlighted in CP.3). CP.2 in turn referred to a problem related to be aware about statistics on the client’s feedback. This made the analyst consider a local interface with the Manager to provide those statistics. The analyst also considered that to be able to analyze customers´ behavior (CP.4) and to be able to forecast sales (CP.5), the software shall have a database to register the interactions with clients and their purchases history. In addition to an interface to interact with the manager, the software could also have a LAN interface with the Sales Management Software. Based on this reasoning, Table 9 states the software glance defined for the CRM software example and Figure 4 shows a block diagram of the software glance.

Table 9. Software Glance definition of the CRM software.

The CRM software will interact with the clients through a web interface allowing marketing, promotions, receiving feedbacks, and responding to the clients. In addition, the CRM software will provide local interfaces to interact with the manager. Data about the clients, feedback, and sales history will be stored in a local database. The CRM software will also include a LAN interface with the Sales Management Software.

Source: the authors

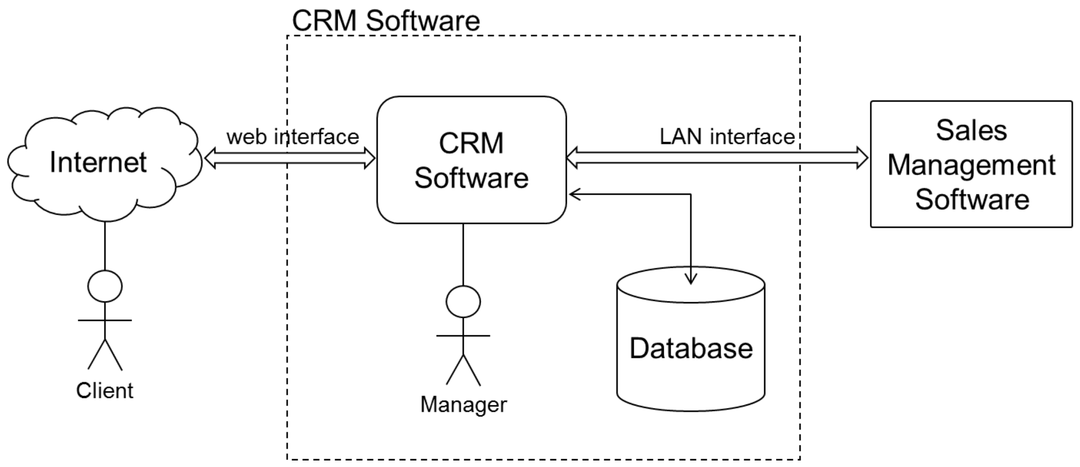

Figure 4. CRM software glance block diagram

Source: the authors

4.4 CUSTOMER NEEDS WITH RESPECT TO THE CRM

At this point, the software analyst knows the identified CPs and has a “glance” of the envisaged CRM software. The next step is to determine what the customer needs are as outcomes from the software solution to solve the CPs. In the proposed approach, the needs are expressed as information, control, entertainment or constructions that the software solution shall provide.

Considering CP.1, the analyst concluded that the company should know who its clients are and be aware about their current contact information. Additionally, the analyst observed that for ensuring a communication channel with its customers the company should contact them regularly and be aware of the ongoing company-client relationship. Then, the analyst specified CN.1, CN.2, and CN.3 as outcomes needed from the CRM software in order to solve or help to solve CP.1. Table 10 shows the resulting CNs obtained from the analysis of the CPs.

Table 10. Customer needs for the CRM

CN.1 - The company needs a CRM software to identify its clients and keep track of their contact information are.
CN.2 - The company needs a CRM software to be aware of the clients that have to be contacted in order to maintain active the communication with them.
CN.3 - The CRM manager wants a CRM software system to be aware of the ongoing company-client relationships.
CN.4 - The CRM manager needs a CRM software to have knowledge the statistics on clients' feedback.
CN.5 - The Account manager needs a CRM software to receive and manage feedback from its clients.
CN.6 - The Account manager wants a CRM software to be aware of the received feedback to be answered.
CN.7 - The Account manager aims a CRM software to understand the client's behavior.
CN.8 - The Account manager needs a CRM software to forecast sales.

Source: the authors

4.5 SOFTWARE VISION FOR THE CRM

After knowing the customer needs and having a “glance” on the software solution, the software analyst can produce (*i.e.*, design) the vision of the CRM software. This software vision details the software glance taking into account the outcomes of the software (CNs) that are needed by the customer. For the CRM software example, the analyst decided the vision document will include the software positioning, involved stakeholders, high-level software features, and software environment, in addition to the high-level software architecture diagram. Table 11 shows a simplified version of the CRM software vision document and Figure 5 presents the CRM software high-level architecture.

Table 11. Software vision of the CRM

Positioning: The CRM software is for account and customer relationship managers who have to maintain close relationship with company clients. The CRM software is an intuitive platform that ensures continuous communication channels with clients and helps understanding clients behavior and forecast sales.

Stakeholders: Account manager, CRM manager, and company clients.

High-level features:

- The CRM software shall provide communication channels with clients through the Internet.
- The CRM software shall provide sales forecasts, analysis of clients behavior, and statistics on clients feedback.
- The CRM software shall ensure appropriate security standards.
- The CRM software shall be compatible with IT technologies currently employed by the company.
- The CRM software shall be structured in front-end and back-end subsystems.

Environment: The CRM Managers mainly use mobile devices and high-speed internet connectivity. Account managers are co-allocated in the company office using standard corporate computers. As corporate standard, all software solutions require to use cloud-based infrastructure and web access.

Source: the authors

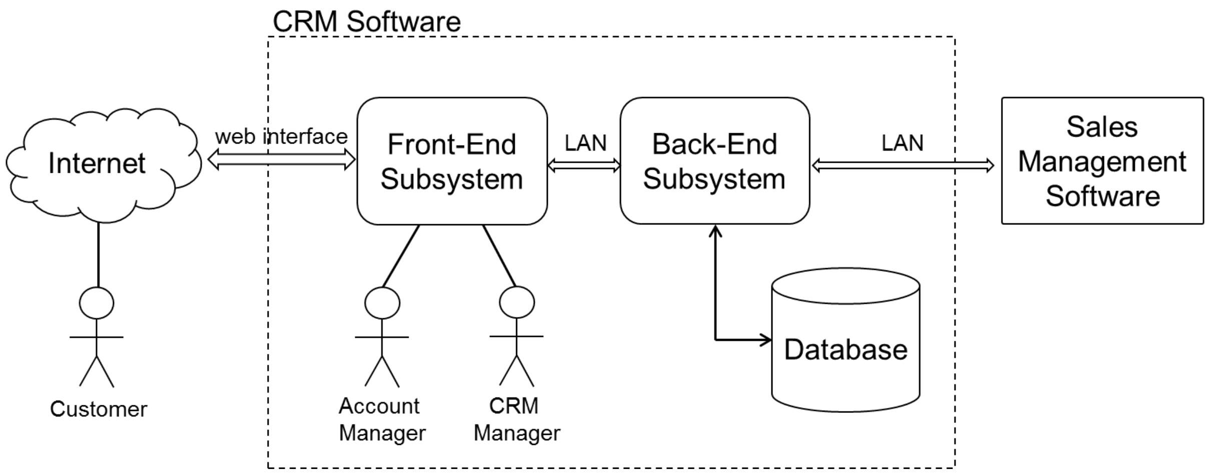

Figure 5. CRM software vision block diagram

Source: the authors

4.6 SOFTWARE REQUIREMENTS OF CRM

Given the software vision and knowing the customer needs, the software analyst can define the functional and non-functional requirements for the CRM software to ensure complete and correct coverage of the specified customer needs.

For the CRM example, the analyst specified four requirements to address the outcome related to CN.1 (*i.e.,* need to identify the company´s clients and their contact information). These requirements are related to registering a new client (FR.1), changing data of a client (FR.2), viewing data of a client (FR.3), and making searches on the client database (RF.4).

Considering CN.2, the analyst specified two requirements to address the needed outcome (*i.e.,* be aware of the clients that have to be contacted). These requirements are related to registering a new contact made with a client (FR.5) and viewing contacts that have to be made with clients (FR.6). Table 12 presents a list of FRs specified based on CN.1 e CN.2.

Table 12. Software requirements specification for CN.1 and CN.2 of the CRM

FR.1 - The CRM software shall allow the Account Manager to register a new client in the database.
FR.2 - The CRM software shall allow the Account Manager to change data of a client in the database.
FR.3 - The CRM software shall allow the Account Manager and the CRM Manager to view data of a client in the database.
FR.4 - The CRM software shall allow the Account Manager and the CRM Manager to make searches on the client database.
FR.5 - The CRM software shall allow the Account Manager to register a new contact made with a client in the database.
FR.6 - The CRM software shall allow the Account Manager and the CRM Manager to view the contacts that have to be made with clients.

Source: the authors

5 DISCUSSION AND MAIN CONTRIBUTIONS

This section discusses the three major contributions of the Problem-Based SRS approach.

Distinguishing the problem from the needs: Problem-Based SRS approach proposes an alternative meaning for problems and needs. As discussed in the related works section, the notions of problem and need are often confusing and even ambiguous. Thus, the proposed concepts contribute to enriching the comprehension and, consequently, the precision of the specification.

Software Glance concept: Authors proposed explicitly capturing the early view of the envisage solution in form of a rough idea called software glance. It represents what is in the software analyst’s mind in terms of abstract view of the software solution after understanding the customer’s problem. Therefore, the software glance provides an early guidance for software analysts in the specification of software customer needs and further software vision.

Comprehensive approach: the processes and concepts that encompass the Problem-based SRS constitute a comprehensive specification approach through a systematic organization of activities, specification artifacts, and logical flow. The approach does not dictate a specific work flow. Activities can be conducted sequentially or following iterative and incremental flows (*e.g.* agile methods).

6 CONCLUSION AND FUTURE WORK

This paper introduced a new and comprehensive approach for the specification of software requirements. The benefits of using a Problem-based SRS approach are: (i) it provides a method that describes the structure and links between concepts making clear the path to elaborate the requirements specification based on the definition of the customer’s problem, (ii) it clarifies and provides distinct concepts and notations for the definition of customer problems and customer needs, and (iii) it structures the early definition of the software through the concepts of software glance and software vision.

The Problem-Based SRS approach enriches the process of software requirements specification by making clearer how software requirements are specified from the identification of the customer’s problems and needs. Indeed, this may have a significant impact in the quality of software requirements specification and in providing the right software to the stakeholders in a business context. Moreover, the concepts and the proposed process for the Problem-Based SRS approach are flexible enough to be used together with current methodologies discussed in the related works section, such as Gore and VPD.

Although this paper presented an entire process for the specification of software requirements, it did not present any technique to conduct each process of the Problem-Based SRS approach. For instance, different elicitation techniques may be used within the problem specifying process like workshops and interviews. Developing techniques for the Problem-Based SRS approach is in progress and the results will be presented in future publications.

REFERENCES

BLANK, S; DORF, B. *The Startup Owner's Manual.* K&S; Ranch. NBR 6023. 2012.

BOURQUE P.; FAIRLEY R. E. Guide to the Software Engineering Body of Knowledge, Version 3.0, IEEE Computer Society, 2014. Available in: http://www.computer.org/portal/web/swebok. Accessed in 17/11/2014.

DANEVA, M.; DAMIAN, D.; MARCHETTO, A.; PASTOR, O. Empirical research methodologies and studies in Requirements Engineering: How far did we come? *Journal of Systems and Software*, n. 95, p. 1-9, 2014. http://dx.doi.org/10.1016/j.jss.2014.06.035

DARDENNE, A.; VAN LAMSWEERDE A.; FICKAS, S. Goal-directed requirements acquisition. *Science of Computer Programming.* 20.1: 3-50. 1993. http://dx.doi.org/10.1016/0167-6423(93)90021-G

DAVEY, B.; PARKER, K. Requirements elicitation problems: A literature analysis. *Issues in Informing Science and Information Technology*, n. 12, p. 71-82, 2015.

DAVIS, A. M.; DIESTE, O.; HICKEY, A. M.; JURISTO, N.; MORENO, A. M. Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. 14th IEEE International Requirements Engineering Conference (RE'06), 2006. http://dx.doi.org/10.1109/RE.2006.17

DORI, Dov. *Object-process methodology: A holistic systems paradigm.* Springer Science & Business Media. 2011.

GIORGINI, P.; MYLOPOULOS, J.; NICCHIARELLI, E.; SEBASTIANI, R. Reasoning with goal models. In: Conceptual Modeling-ER 2002 (pp. 167-181). Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-45816-6_22

HOFMANN, H.F.; LEHNER, F. Requirements engineering as a success factor in software projects in IEEE Software, vol.18, no.4, pp.58-66, Aug 2001. http://dx.doi.org/10.1109/MS.2001.936219

IBM. Knowledge Center, Rational Team Concert 4.0.5, Requirement Management Application. 2015. Available in http://www-01.ibm.com/support/knowledgecenter/SSCP65_4.0.5/com.ibm.rational.rrm. help.doc/topics/r_vision_doc.html?lang=en-us. Accessed in 17/11/2015.

IEEE Computer Society. Software Engineering Standards Committee, and IEEE-SA Standards Board. IEEE recommended practice for software requirements specifications. Institute of Electrical and Electronics Engineers, 1998. http://dx.doi.org/10.1109/IEEESTD.1998.88286

IIBA, *A. Guide to the Business Analysis Body of Knowledge (BABOK Guide)*. International Institute of Business Analysis (IIBA), 2009.

INCOSE, System Engineering Handbook, *A Guide for System Life Cycle Processes and Activities*, Hoboken, Wiley, 2015.

ISO, IEC. IEEE, Systems and Software Engineering - Vocabulary. ISO/IEC/IEEE 24765: 2010 (E) Piscataway, NJ: IEEE computer society. http://dx.doi.org/10.1109/IEEESTD.2010.5733835

ISO, IEC. IEEE. 29148: 2011, Systems and Software Engineering - Requirements Engineering. Technical report. 2011. http://dx.doi.org/10.1109/IEEESTD.2011.6146379

JACKSON, M. A. Problem analysis using small problem frames. *South African Computer Journal*, p. 47-60. 1999.

JACKSON, M. A. Information and Software Technology, Special Issue on Problem Frames, v. 47, n. 14, p. 903-912, November 2005. http://dx.doi.org/10.1016/j.infsof.2005.08.004

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *Unified Software Development Process.* Addison-Wesley. 1999.

LEFFINGWELL, D.; WIDRIG, D. *Managing software requirements: a use case approach.* Addison-Wesley Professional. 2003.

MCKEAN, Erin. *The new oxford American dictionary.* Vol. 2. New York: Oxford University Press. 2005.

NOVAKOVIC, D.; HUEMER, C. *A survey on business context. Intelligent Computing, Networking, and Informatics*, p. 199-211. Springer India, 2014.

OSTERWALDER, A; PIGNEUR, Y. Modeling value propositions in e-Business. Proceedings of the 5th international conference on Electronic commerce. ACM, 2003. http://dx.doi.org/10.1145/948005.948061

OSTERWALDER, A.; PIGNEUR, Y.; BERNARDA, G.; SMITH, *A. Value Proposition Design: How to Create Products and Services Customers Want.* Hoboken, John Wiley & Sons. 2014.

VALASKI, J; STANCKE, W; REINEHR S., MALUCELLI A. WER Overview: Retrospective, Trends and Relevance. CLEIej, Montevideo, v. 17, n. 3, dec. 2014. Available in http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002014000300004&lng=es&nrm=iso. Accessed in 17/11/2015.

VAN LAMSWEERDE, A. Goal-oriented requirements engineering: a roundtrip from research to practice [engineering read engineering]. In Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International (pp. 4-7). IEEE. http://dx.doi.org/10.1109/ICRE.2004.1335648

WIEGERS, K; BEATTY, J. *Software Requirement. 3*. ed. Pearson Education: Microsoft Press, U.S. 2013.

YU, E; GIORGINI, P; MAIDEN, N; MYLOPOULOS, J. *Social modeling for requirements engineering*, 1st ed. Cambridge: MIT Press. 2011.

ZOWGHI, D.; COULIN, C. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools, Engineering and Managing Software Requirements. pp. 19-46). Springer Berlin Heidelberg, 2005. http://dx.doi.org/10.1007/3-540-28244-0_2