Transdisciplinary Engineering: Crossing Boundaries
M. Borsato et al. (Eds.)

doi:10.3233/978-1-61499-703-0-229

A Notification-Oriented Approach for Systems Requirements Engineering

Jean M. SIMÃO [a,1], Hervé PANETTO [a,b,c], Yongxin LIAO [d] and Paulo Cézar STADZISZ[a]

[a] *Federal University of Technology, Paraná, UTFPR – Program of Electrical Engineering and Industrial Computer Science (CPGEI), Brazil*
[b] *CNRS, Research Centre for Automatic Control (CRAN UMR 7039), France*
[c] *Université de Lorraine, CRAN UMR 7039, France*
[d] *Pontifical Catholic University of Paraná, PPGEPS, Brazil*

Abstract. Systems Engineering (SE) is an approach for designing complex systems in a multidisciplinary universe, based on concepts from the systemic paradigm and promoting languages, methods, and standardized processes. Requirements engineering is one of the main steps in SE processes. The current research presented in this paper aims at focusing on the open issue related to the formalization of systems requirements for their verification, ensuring the coherence of the whole set of requirements in each contextual engineering domain and their validation against the initial stakeholders' needs. Moreover, requirements coming from different domains are generally linked by non-formalised traceability relationships. It is even difficult to trace any change in their definition and their impact to the whole set of specifications. The paper discusses and proposes an approach for systems requirements engineering based on a rule and notification oriented approach for ensuring the effective coherence and understanding of these requirements throughout the life cycle of any complex system. This proposed notification approach is derived from the so-called Notification Oriented Paradigm (NOP), a new rule and event driven approach for software and hardware specification and execution.

Keywords. Systems Requirements Engineering, System Specification, Notification Oriented Paradigm

Introduction

Currently, in order to face globalization and the resulting increased competition, enterprises have specialized in specific domains and have established partnerships with other companies to complement their initial skills. These enterprises are thus forming a so-called collaborative and distributed network. These approaches have allowed them to develop complex systems and collaborative activities in many industrial domains like aeronautics, nanotechnology, aerospace, and bioengineering.

According to [1], it is important, for succeeding in these collaborative engineering processes, to formalize how different partners can work with others and, through their interactions, how they can achieve a common objective within different perspectives.

[1] Corresponding Author, E-Mail: jeansimao@utfpr.edu.br

These engineering processes follow best practises generally defined in the so-called systems engineering domain.

Systems Engineering (SE) is an approach for designing complex systems in a multidisciplinary universe, based on concepts from the systemic paradigm and promoting languages, methods, and standardized processes (ISO/IEC 15288) [1]. It aims at consolidating, identifying, and formalizing new methods and frameworks that support engineering phases in a better consistent way [2]. It is "an interdisciplinary approach and means to enable the realization of successful systems" [3]. One of the SE processes is dedicated to analysing users' and systems' requirements.

Requirements Engineering (RE) refers to the activity of formulating, documenting and maintaining systems requirements [4] in order to produce, from users' needs, a set of specification related to what the final system should be. Requirements provide the basis for all phases of the development system. Thus, it is necessary to control these requirements in all phases of the development cycle and in all domains to avoid some misinterpretation and mistakes committing the final results.

A requirement is a statement from the stakeholder's needs in order to define a product, a system or a process and it must be unambiguous, clear, unique, consistent, stand-alone, and verifiable [5]. Each requirement matches a single part of the future product, system or process and it is grouped in an appropriate combination of textual statement views. Whereas approaches such as the Model-Based Systems Engineering (MBSE) have been studied [6][7] for improving the definition and the coherency of requirements [8], there is still difficulties to ensure that coherency from a semantic point of view and to identify all impact relationships when any of these requirements change during the system development lifecycle.

1. Related Works

Requirements Engineering has long been recognized as critical activity in systems and software development processes [9][10][11]. A large amount of studies addresses theoretical aspects and propositions of techniques and recommended practices for RE [12]. Hofmann and Lehner [13] identified RE practices that clearly contribute to (software) project success and concluded that successful projects allocate a significantly higher amount of resources to RE (28%). Many (37%) of projects failures [14] are caused by the problems of requirements misinterpretation among stakeholders. Most studies have focused on requirements elicitation, modelling, and processes/methods 1395–1410
[15][16].

A number of researches reported in the literature focus on the elicitation of requirements. According to [17], this activity is the process of seeking, uncovering, acquiring, and elaborating requirements for (software) systems. It concerns learning and understanding the needs of the customer with the aim to communicate these needs to the developers [18]. Additionally, there is a general agreement [19] that fixing the results of poor requirements elicitation is more expensive than for other mistakes. It is a common sense in most studies that systems requirements have to satisfy the customer´s (i.e. stakeholders) intents. The ISO/IEC/IEEE standard uses the expression "in a way that is (the requirement) acceptable to the customer" [20] to emphasize that the requirements must achieve the customer´s intentions. Thus, some authors propose to validate the specified requirements by determining their conformity with stakeholders´

needs [13]. The term "need" is often employed to refer to the cause or reason that justifies the specified requirements [18]. The needs would be the source of the requirements. Some authors, inspired by INCOSE[2] observe that the stakeholders' needs in turn contribute to the solution of some real-world problems [21][22].

In the context of systems engineering, the deep understanding of a system's intents and how they maps to systems requirements is as important as for software engineering, and the underlying concepts are also analogous. 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 [5]. In spite of the large number of studies and approaches proposed to specify systems requirements, for any "complex system" development (e.g. software, aerospace, automotive), it is a common agreement that requirements engineering remains an unclear and challenging task. Terms used in scientific literature and even in the industry (as pointed out in this section) are not convergent creating a lack of understanding on the subject. Analysts may feel confused when trying gathering information from the stakeholders and other sources in the business or application domain. Therefore, the specification of the studied system may become incomplete or incorrect because of an inadequate understanding of the project intents.

Generally speaking, the various works on requirements engineering show that there are two unsolved main issues/questions [9] that this paper tackles in the proposed approach: (1) how to "semi-automatically" model a set of requirements taking into account their strong inter-relationships? (2) How to identify/formalise these inter-relationships that are generally domain-dependent and thus related to some deep and implicit knowledge of the related skills of the stakeholders? [23]

This paper discusses and proposes a solution only for the first question. Indeed, the model of all requirements of a system and their inter-relationships can be seen as a bi-graph of interrelated notifications where each requirement is a logical premise manipulating some systems attributes and notifying some functional entities which, in turn, forward new attributes values to any impacted requirement. This operational semantics is quite analogue of the so-called NOP paradigm that we will discuss briefly in the following section.

2. Notification-Oriented Paradigm (NOP)

A new technique, called Notification Oriented Paradigm (NOP), was proposed as new software programming and developing approach. NOP presents a new concept to develop and execute applications. Its essence is an inference process based on small, smart, and decoupled pieces of software (i.e. entities) that collaborate by means of notifications. This solves redundancies and centralization problems of the causal processing, thereby solving processing capacity misuse and coupling issues. In NOP, causal expressions are represented by a set of logic-causal rules and dealt by entities called *Rules*. A *Rule* entity, represented as a logical and causal expression, is illustrated in Figure 1 (a). Structurally, a *Rule* entity has a *Condition* entity and an *Action* entity which respectively concern the decisional part and the execution part. Each element evaluated by a *Rule* set is represented and dealt by an entity called *Fact Base Element* (*FBE*). In the considered example, the *FBEs* are the *Security_System and User_Reader*.

[2] International Council on Systems Engineering, http://www.incose.org

A *FBE* comprises one or more *Attribute* entities that store data. Examples of *Attributes* are the *Status* in the *FBE Security_System* and the *Bio* in the *FBE User Reader*.

The values of *Attributes* are analysable, in an inference process, by the *Conditions* of *Rules*, using other collaborative entities called *Premise*. In the considered *Rule* (Figure 1 (a)), its *Condition* comprises two *Premises*. When each *Premise* of a *Condition* is inferred as true, the related *Rule* becomes enabled and it can activate its *Action* composed of entities called *Instigation*. In the considered *Rule*, the *Action* has one *Instigation*. In fact, *Instigations* are linked to *Methods* of *FBEs* to execute services. In the *Action* of that *Rule*, the *Method Activate_Alarm()* is instigated. Commonly, instigating a *Method* of an FBE changes the values of *Attributes* [24][25].

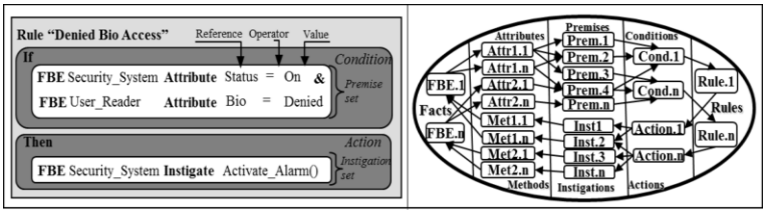

Figure 1. (a) The Representation of a NOP *Rule*. (b) NOP Components and Notification Chain.

The NOP inference process is innovative. Indeed, the *Rules* do not become enabled by matching *Attribute* values by means of some usual search approach, but by evaluating their *Conditions* when they are notified by *FBEs* that *Attributes*' values changed. Inference happens in the following way: for each change in an *Attribute* value of a *FBE*, based on the new value, just the very pertinent *Premises* are notified and make its logical analysis; for each change in a Premise logical state, it notify just the very pertinent *Conditions*. In turn, *Conditions* enable new *Rules* that may execute *Actions* by notifying *Instigations* which in turn execute *Methods* of *FBEs*. The collaboration between NOP elements by means of notifications is illustrated in Figure 1 (b). This notification chain is created during the software compilation phase [24][25].

3. Proposed Approach for Systems Requirements Engineering

3.1. NOP Modelling Primitives

According to the NOP specification [25], the NOP modelling primitives essentially includes *Rules*, *FBEs*, and *Notifications*. These building blocks allow describing the entire logic of software systems. Additional primitives may be employed to specify particular constraints, aspects of software flow control, and conflict solving, among other features of the system being modelled. In the proposed approach for systems requirements engineering, the authors make use of the three main NOP primitives in order to describe those requirements at the same level of detail then those sentences expressed previously by the system engineer, according to the information expressed by the stakeholders. The meaning of the three NOP primitives in the context of this paper are described in the next topics.

Rule - A *Rule* in NOP is defined as a logical unit, which commands a set of actions when its conditions are satisfied. In the proposed approach, a *Rule* represents part of or an entire system requirement, including the constraints with regard the needed conditions (i.e. *Rule* preconditions) for that requirement and the effects or actions to

be accomplished (i.e. *Rule* post-conditions) according to that requirement. Depending on the system requirement complexity, one or more rules may be composed to express each requirement. The notation to draw a *Rule* is depicted in Figure 2 (a).

Fact Base Element (*FBE*) - In NOP, an *FBE* represents a software element, which may contain *Attributes*, can carry out functions in *Methods*, and can interface with external elements (e.g. user interface, sensors, and devices) also by means of *Methods*. In this paper, when specifying systems requirements, authors also propose to use *FBE* to represent system elements identified in the requirements *statements*. This way, *FBEs* are limited to the elements known at the requirements specification phase. *FBEs* are drawn as dashed rectangles in the proposed notation as illustrated in Figure 2 (b). They must contain one or more exposed *Attributes* that represent notified events or variables. A *FBE* may also contain incoming arrows that represent functions called by other FBEs.

Notification - A *notification* is an explicit advice from a NOP element to another one, indicating that a change in the system state occurred. This may means that an entity changed its value or an event happened, for instance. In this paper, authors consider that a notification represent a link between *Rules* and *FBEs*. These links may have two meanings, according to the link direction. A notification from a *FBE* to a *Rule* (i.e. *FBE* → *Rule*) describes a given precondition for that *Rule* related to the indicated *Attribute* of the *FBE*. A logical expression from a defined algebra must be assigned over the link to describe the logical condition. Figure 2 (c) illustrates this type of link. On the other hand, a notification from a *Rule* to a *FBE* (i.e. *Rule* → *FBE*) describes that the *Rule* invokes a function from a *Method* of an *FBE*. A reference to the *Action* is to be written over the link as illustrated in Figure 2 (d).

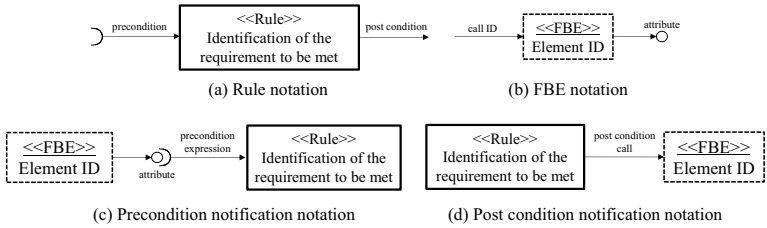

Figure 2. Notation used for drawing systems requirements models.

3.2. NOP Based Requirements Modelling

From the primitives established in the previous section, authors propose to construct the system requirements specification in the form of a system requirements model. This model may include a single or multiple diagrams using the NOP notation. The input for the modelling process is the system requirements sentences. A general view of the adopted modelling technique follows.

For each requirement statement in the system requirements specification:

1. *To analyse the requirement sentence aiming at:*
   - i. *Identifying the functional or non-functional request in the requirement.*
   - ii. *Identifying the Conditions for the functional or non-functional request.*
   - iii. *Identifying the Attributes involved in the Conditions.*

iv. *Identifying the Actions for the functional or non-functional request*
v. *Identifying the functions related to Methods instigated in the Actions.*
vi. *Identifying the FBEs related to the Attributes for the request.*
vii. *Identifying the FBEs related to the Methods indicated by the request.*

2. *To create a Rule for every request identified in step 1.*
3. *To create a FBE for every entity identified in step 1.*
4. *To create links (i.e. notifications between Rules and FBEs according to the Conditions and Actions related to rules) identified in step 1*
5. *To merge FBEs and Rules with analogous FBEs and Rules previously created.*

Step 1 involves analysing every requirement sentence in order to identify the specific request inside the stated requirement. Commonly, it should exist only one request per requirement. However, sentences in natural language expressing requirements for a system may explicitly or implicitly refer to more than one request. Additionally, the request may specify an intended action to be carried out by the system (referred to as functional request) or it may specify constraints, properties or conditions for the system (referred to as non-functional requests). In this step, the elements related to the conditions and actions of the request are also identified considering the references to aspects, objects, devices or interfaces with the external elements. Steps from 2 to 4 are related to the construction of the requirements model from the elements identified in step 1, i.e. the identified *Rules*, *FBEs* and links. Step 5, particularly, concerns integrating models constructed from each system requirement. This may involve merging *Rules* and *FBEs* and reconnecting links between them.

To illustrate the proposed modelling technique it is taken into account the following example of system requirement statement: "The system shall activate the fire alarm when the temperature sensor indicates more than 60°C in the room". Analysing this requirement sentence, the system analyst can identify "activate the fire alarm" as the stated functional request. The condition of the *Rule* for that request is "temperature more than 60°C" and the element associated to this condition is "temperature sensor". The *FBE* temperature sensor shall expose an *Attribute* that contains the current temperature in the room. Finally, the element responsible for the function "activate" is the "Fire Alarm" instigated by the action of the rule. Thus, following the steps 2, 3 and 4 of the proposed technique, the resulting model is presented in Figure 3.

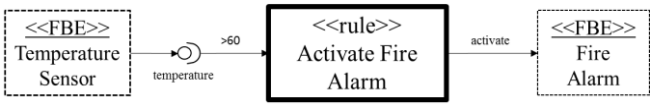

Figure 3. System requirement model example.

4. Case Study: Access Security System

This section presents the modelling example of an "Access Security System" extracted from the INCOSE SE Handbook v3.2 [3]. This study is based on six system requirements that specify the intended functional and non-functional characteristics of the system.

4.1 Case description

According to the SS11 Stakeholder Requirement presented in [3], the secure areas (i.e. rooms that have limited access) are to be protected by two independent security checks. One of them is based upon an employee ID and the other one is based upon biometric data. The time between the two independent security checks shall not exceed a configurable period. The user is allowed three attempts at biometric and/or card identification before access is completely disabled. Any denied access attempt is to be sent to the administrator.

The system requirements statements are:

- *SS11 – a: Secure areas shall be protected by security check based upon employee ID.*
- *SS11 – b: Secure areas shall be protected by a second independent security check based upon biometric data.*
- *SS11 – c: The time between the two independent security checks shall not exceed a configurable period.*
- *SS11 – d: The user shall be allowed three attempts at biometric identification.*
- *SS11 – e: The user shall be allowed three attempts at card identification.*
- *SS11 – f: Any denied access attempt shall be sent to the administrator.*

4.2. NOP-Based Requirements Models

This section applies the proposed technique showing the main phases of the Access Security System requirements modelling.

Requirement SS11-a:

This requirement indicates that the system shall "protect secure areas" what means that the system shall meet two requests: enabling and disabling access to the secure area. As consequence, the system will have to command an element (e.g. a blocker) that carry out these two actions. This requirement also states that a "security check based upon employee ID" will provide the condition to enable or disable access to secure areas. The identified attribute for this condition is the "employee ID" read from an external element (i.e. an ID card reader). Figure 4 shows the model of this requirement where three *Rules* and two *FBEs* are identified.

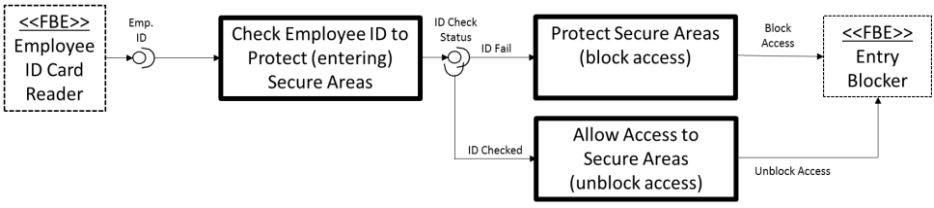

Figure 4. Model of the SS11-a requirement.

Requirement SS11-b:

This requirement is similar to SS11-a and its model will also include two *FBEs* and three *Rules*. The first *FBE* represents the *Biometric Reader* element and the second the *Entry Blocker*. The last is the same *FBE* indicated in the model of SS11-a because the *Entry Blocker* will be obviously the same. The first *Rule* will represent request for

checking the employee biometric. The two other *Rules* represent the request to enable and disable access to the secure area as modelled to SS11-a.

Because of their inter-relationship, SS11-a and SS11-b models can be integrated, as illustrated in Figure 5. The *Rule* "Protect Secure Areas" receives a <<disjunction>> operator to indicate that any or both conditions enable this *Rule*. The *Rule* "Allow Access" in turn receives a <<conjunction>> operator to indicate that both conditions must be satisfied to enable this rule.

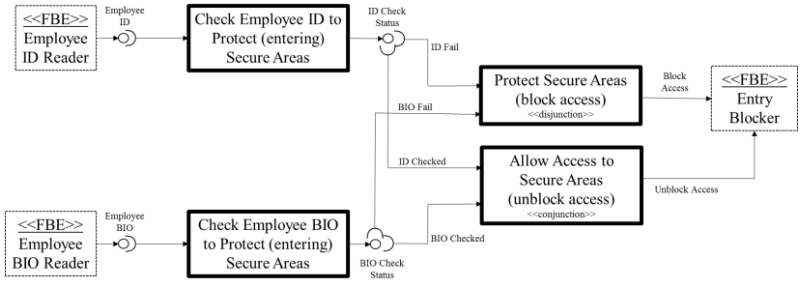

Figure 5. Model of SS11-a and SS11-b requirements.

Requirement SS11-c:

This requirement adds a new time constraint with respect the secure checking of ID and BIO. It leads to a new functional request involving checking the elapsed time between the ID and BIO secure checking. It also sets new conditions for enabling or disabling access and establishes two new *Attributes* for the current system time (*Cur_Time*) and system configured time period (*Conf.Period*). This way a new *Rule* (*Check Elapsed Time*) and two new *FBEs* (*System Clock and System Config*) are inserted in the model as illustrated in **Figure 6**.

Requirement SS11-d and Requirement SS11-e:

These requirements are very similar once both limit the number of user attempts. The request in both requirements can be merged in a single request that counts the number of ID and BIO checking attempts. A new *FBE* (*User Attempts Counter*) is created and new conditions about the status of the user attempts are inserted for the *Rules* that represent the request to enable and disable access to the secure area as illustrated in **Figure 6**.

Requirement SS11-f:

This last requirement defines a new functional request for notifying the administrator when a user access attempt is denied. This leads to a new *Rule* to represent this notification. However, because the conditions for that *Rule* are identical to those of the "*Protect Secure Areas*" *Rule*, they can me merged and a new link is create to command the notification action by the *Administrator Interface FBE*. Figure 6 illustrates the final system requirements model for the considered example.

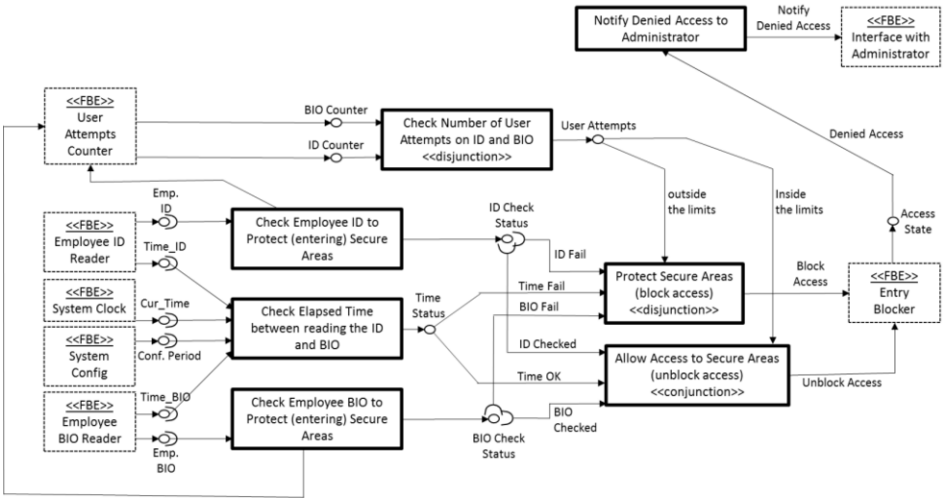

Figure 6. Final system requirements model.

5. Conclusions and Future Works

This paper proposed a novel approach for systems requirements modelling that makes use of concepts and the notation of the previously developed Notification Oriented Paradigm (NOP). NOP has been successfully applied for modelling, programming, and executing software applications based upon three fundamentals primitives: *Rules*, *FBEs*, and notifications. Based on these concepts, this paper presented a technique for constructing a graphical model of the expressed system requirements. An example of an access security system illustrated the proposed approach.

The presented approach uses a graphical notation, thus facilitating the analysis of the requirements and identifying hidden knowledge. Additionally, this approach makes explicit the logical dependencies (*Rules* with their conditions and actions) between the requirements through linked and shared *Attributes* and *Methods* of involved elements (*FBEs*). These characteristics allow modelling the whole set of requirements taking into account their inter-relationships. In the context of larger scaled systems, the proposed approach should provide means and tools for semi-automatically generating systems requirements models.

However, the proposed approach does not yet take into account the meaning behind the requirements in order to identify hidden semantics inter-relationships. Indeed, this semantics is generally domain-dependent and thus related to some deep and implicit knowledge of the related skills of the stakeholders. Authors' working in progress explore using ontologies as a suitable solution for expressing these aspects of requirements together with the modelling approach presented in this paper.

Acknowledgement

This research was partially supported by CAPES (*Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*) from Brazil.

References

[1] ISO/IEC, *ISO/IEC 15288:2012 - Systems and Software Engineering - Software Life Cycle Processes*, 2012, International Organization for Standardisation, Geneva, Switzerland

[2] A. J. Lopes, R. Lezama, and R. Pineda, Model Based Systems Engineering for Smart Grids as Systems of Systems, Procedia Computer Science, vol. 6, pp. 441–450, 2011.

[3] INCOSE, *INCOSE Systems Engineering Handbook: A Guide for Life Cycle Processes and Activities*, The International Council on Systems Engineering, C. Haskins, ed. 3, 2006.

[4] R. R. Young, *The Requirements Engineering Handbook*, 1 ed, Artech House, Boston, 2004.

[5] BKCASE, *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v. 1.6. R.D. Adcock (EIC). Hoboken, Accessed: 12/04/2016. Available: www.sebokwiki.org

[6] INCOSE, *What is Model Based System in Engineering (MBSE)*, version 1.0, 2012. http://www.incoseonline.org.uk/Documents/zGuides/Z9_model_based _WEB.pdf.

[7] J. N. Mazón, J. Pardillo and J. Trujillo, A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In: J.-L- Hainault et al. (eds.) *Advances in Conceptual Modelling – Foundations and Applications*, Springer Berlin / Heidelberg, pp. 255–264, 2007.

[8] A.L. Szejka, A. Aubry, H. Panetto, O. Canciglieri Jr., E. Rocha Loures, Towards a conceptual framework for requirements interoperability in complex systems engineering, *9th International Workshop on Enterprise Integration, Interoperability and Networking (EI2N'2014)*, Amantea, Springer, OTM 2014 Workshops, Oct 27-31, LNCS 8842, pp. 229-240.

[9] G.-C. Roman, A taxonomy of current issues in requirements engineering, *Computer*, 1985, pp. 14–23.

[10] J. Siddiqi, Requirement engineering: The Emerging Wisdom, *IEEE Software*, Vol. 13, 1996, pp. 15-19.

[11] I. Sommerville and G. Kotonya, Requirements engineering: processes and techniques, Wiley, New York, 1998.

[12] M. Daneva, D. Damian, A. Marchetto and O. Pastor, Empirical research methodologies and studies in Requirements Engineering: How far did we come?, *Journal of Systems and Software*, 2014, pp. 1–9.

[13] H.F. Hofmann and F. Lehner, Requirements engineering as a success factor in software projects, *IEEE Software*, Vol. 18, 2001, pp. 58-66.

[14] G. Locatelli, M. Mancini, and E. Romano, Systems Engineering to improve the governance in complex project environments, *International Journal of Project Management*, Vol. 32(8), 2014, pp. 1395–1410

[15] S. Ratchev, E. Urwin, D. Muller, K. S. Pawar and I. Moulek, Knowledge based requirement engineering for one-of-a-kind complex systems, *Knowledge-Based Systems*, Vol. 16, 2003, pp. 1–5

[16] J. Valaski, S. Reinehr and A. Malucelli, Environment for Requirements Elicitation Supported by Ontology-Based Conceptual Models: A Proposal, in: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, 2014.

[17] Y. Bernard, Requirements management within a full model-based engineering approach, *Systems Engineering*, Vol. 15, 2012. doi:10.1002/sys.20198.

[18] D. Zowghi and C. Coulin, Requirements elicitation: A survey of techniques, approaches, and tools, in: A. Aurum et al. (eds.), *Engineering and Managing Software Requirements*, Springer, 2005, pp. 19–46.

[19] B. Davey and K.R. Parker, Requirements elicitation problems: a literature analysis, *Issues in Informing Science and Information Technology*, Vol. 12, 2015, pp 71–82.

[20] ISO/IEC/IEEE, *ISO/IEC/IEEE 24765:2010 - Systems and software engineering – Vocabulary*, ISO, Geneva, 2010

[21] R.J. Wieringa, Requirements Engineering: Problem Analysis and Solution Specification, in: N. Koch, P. Fraternali, M. Wirsing (eds.), *Web Engineering*, Springer, Berlin, Heidelberg, 2004, pp. 13–16.

[22] P. Bourque and R.E. Fairley, *Guide to the software engineering body of knowledge (SWEBOK (R))*, Version 3.0, IEEE Computer Society Press, 2014.

[23] Y. Liao, M. Lezoche, H. Panetto, N. Boudjlida and E.R. Loures, Semantic annotation for knowledge explicitation in a product lifecycle management context: A survey, *Computers in Industry*, 71, 2015, pp. 24–34.

[24] J.M. Simão, R.F. Banaszewski, C.A. Tacla and P.C. Stadzisz, Notification Oriented Paradigm (NOP) and Imperative Paradigm: A Comparative Study, *Journal of Software Engineering and Applications (JSEA)*, 5/6, 2012, pp. 402-416.

[25] J. M. Simão and P. C. Stadzisz, Inference Based on Notifications: A Holonic Meta-Model Applied to Control Issues, *IEEE Transactions on Systems, Man and Cybernetics*, Part A. 39/1, 2009, pp. 238-250.