Towards a conceptual framework for requirements interoperability in complex systems engineering

Anderson Luis Szejka[1,2,3], Alexis Aubry[1,2], Hervé Panetto[1,2], Osiris Canciglieri Júnior[3], Eduardo Rocha Loures[3]

[1]Centre de Recherche en Automatique de Nancy (CRAN), University of Lorraine, Vandoeuvre-lès-Nancy Cedex, France
[2]CRAN UMR 7039, CNRS, France
{anderson-luis.szejka, alexis.aubry, herve.panetto}@univ-lorraine.fr
[3]Graduate Program in Production Engineering and Systems, Pontifical Catholic University of Parana, Curitiba, Brazil
{osiris.canciglieri, eduardo.loures}@pucpr.br

Abstract. Requirements Engineering (RE) is an important activity in system engineering and produces, from the users' needs, specifications related to what the final system must be. This process in complex systems engineering is extremely intense, because there is a large number of stakeholders involved, with expertise deriving from heterogeneous domains. Moreover, requirements' improvements and variations are common during system life cycle phases. Thus, there is a risk of inconsistency of requirements during the engineering of a system. This paper provides a contribution in requirements engineering as it explores requirements interoperability in complex systems when multiples dimensions are involved. It discusses requirement management according to the cross-domains dimension, the cross-systems life cycle dimension, the cross-requirements dimension and the risk of inconsistency when three dimensions are involved simultaneously during the life cycle phases. The main result is an overview of the existing gaps in one and/or more dimensions allowing a discussion on the possibilities to cope with the problem of requirements inconsistency in multiples dimensions.

Keywords. Requirements Engineering; Requirements Interoperability; Complex Systems Engineering; Requirements Consistency.

1 Introduction

Enterprises have been specializing in specific domains and establishing partnerships with other companies to complement their initial skills to face globalization and consequently its intensified competition. This approach resulted in the so-called collaborative network that allows the development of complex systems and collaborative activities in many industrial domains like aeronautics, nanotechnology, aerospace, bioengineering, etc. According to [1], for succeeding in these collaborative engineering processes, it is important to formalize how different partners can work

together and, through their interactions, how a common objective can be achieved within different perspectives. These engineering processes follow best practices generally defined in the so-called systems engineering domain.

System Engineering (SE) is "an interdisciplinary approach and means to enable the realization of successful systems" [2]. It focuses on holistically and concurrently understanding stakeholder needs; exploring opportunities; documenting requirements; and, synthesizing, verifying, validating and evolving solutions while considering the complete problem, from system concept exploration throughout all phases until system disposal. One of the SE processes is dedicated to analysing users and systems requirements, denominated Requirement Engineering (RE). RE refers to activities of formulating, documenting and maintaining systems requirements [3] to produce, from the users' needs, a set of specification related to what the final system must be.

Requirements provide the basis for all phases of the system development and must be controlled inside all these phases and domains to avoid misinterpretation and mistakes that would compromise the final results [2,4,5]. While approaches such as model-based systems engineering (MBSE) have been studied in [6,7,8], for improving the definition of requirements based on models, there is still a semantic gap between all requirements definitions when they are defined in different domains for the same engineering project and requirement consistency management in different systems life cycle phases. In order to cope with this challenge, we are working to define a conceptual framework that aims to formally model requirements interoperation in term of impact and semantic equivalence or subsumption. This formal definition will facilitate the verification of the system requirements coherence taking into account the technical constraints defined by appropriate experts along the systems life cycle phases.

The paper is structured as follow: Section 2 addresses the problem statement regarding to the management of requirements when multiple information come from multiple stakeholders' needs during the system life cycle phases. Section 3 presents a literature review concerning the main issues on system requirements considering the cross-domains dimension, the cross-systems life cycle dimension and the cross-requirements dimension. Section 4 is devoted to discuss the main drawbacks and existing gaps in related works. Finally, section 5 concludes and presents perspectives for the research continuation.

2 Problem Statement

RE is a key activity in the process of engineering a system. Indeed, complex systems with multidisciplinary perspectives require special attention to ensure that all requirements are fulfilled and misinterpretation and mistakes do not occur during phase's evolution of the system life cycle [9]. In fact, the misinterpretation and mistakes may cause significant *a posteriori* system refactoring, which result in scheduling overruns and increasing the projects costs [10].

The traditional system requirement approach does not support [11,12,13]:

- the cascading impacts of frequent changes or updates of requirements;
- the dispersion of responsibility and the risk of non-consistency of requirements due to the number of stakeholders involved in the development process.

Specialists normally define each requirement using different expertise from heterogeneous domains focused on a single domain and a single life cycle phase. This fact leads to risk of misunderstanding among specialists due to semantic gaps. However, it is important to enhance the system requirements engineering activity that identifies the potential risk for the system-of-interest if one requirement is not satisfied. RE standards, approaches and tools are not able to deal with the risk if the non-satisfied requirement affects other life cycle phases and/or others domains.

For analysing these issues, the authors intend to consider three dimensions of the requirements analysis process as illustrated in Figure 1: (i) the domains dimension; (ii) the system lifecycle phase's dimension; and (iii) the requirements dimension. The first dimension concerns the set of domains involved in system engineering process, for instance mechanical domain, electrical domain, computer science domain. For this particular case, each expert in these domains must define specific requirements based on their particular skills. The second dimension is related to different phases of the systems life cycle, where each phase has its proper constraints represented by specific requirements. The last dimension represents different requirements as basic elements defined by the requirements analysis process, which this requirement will represent accurately the stakeholder's needs. Each one of these requirements is associated to a single domain and a single life cycle phase.

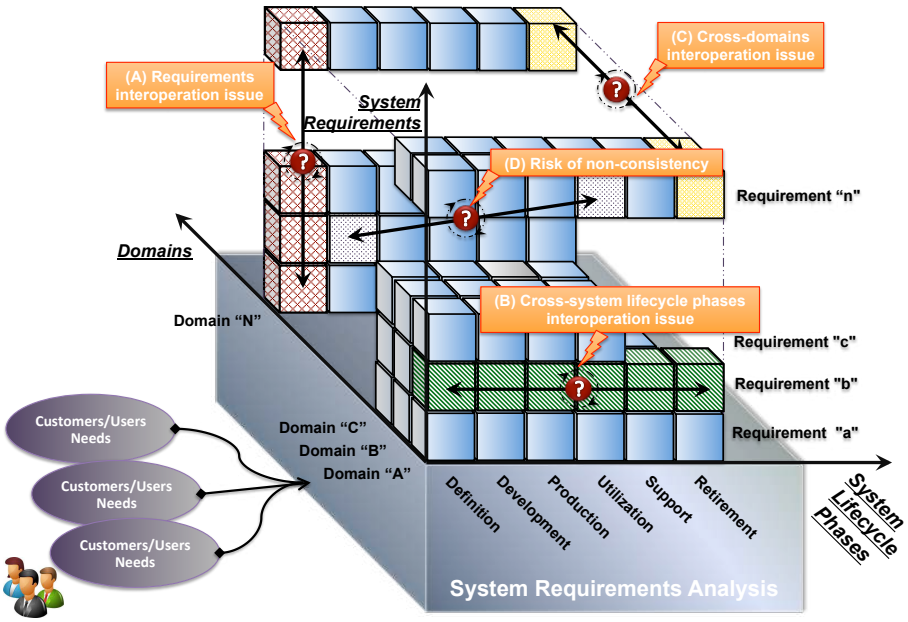

Fig. 1. Requirements analysis dimensions and issues.

For each dimension, an interoperation issue can be identified. Within the requirements dimension there are problems of completeness, coherency, uniqueness, univocity, feasibility, traceability and verifiability (Detail A – Figure 1). The dimension related to the systems life cycle phases may have some issues concerning

the impact analysis between all phases (Detail B – Figure 1). Finally, the main scientific issue comes from the heterogeneity of the domains, which imposes some knowledge representation and analysis for managing requirements and their semantic relationships (Detail C – Figure 1).

The authors also identified a fourth issue that illustrates the interrelationships among the three dimensions presented (Detail D – Figure 1). This last issue, which takes also into account dynamical, interactive and recursive properties of the requirement analysis process, is the most critical one. For example, if a specific requirement in one particular domain for a single phase is added or updated, it may impact other requirements already defined in other domains and/or other phases.

3 Related Works

The related works were structured according to the three issues of this research: (i) cross-domains requirement interoperation, (ii) cross-systems life cycle requirement interoperation and (iii) cross-requirement interoperation in a single domain/systems life cycle phases.

3.1 Cross-domains requirements interoperation issue

The complex systems development requires the involvement of specialists from multiples domains to capture the system's overview as well as the overviews within the domains and their interactions [14]. This generates a multi-heterogeneous information environment from different groups of stakeholders, suppliers, analysts' engineers, etc., to define complex systems. However, the heterogeneity of information from different domains has generated divergences with requirements like misinterpretation and mistakes due to a lack of requirements formalism and impacting in different system life cycle phases [15]. According to [16], the requirement analysts have expertise in systems development, but their knowledge remains restricted to their domains. On the other hand, the stakeholders and other customers involved in the project have different expertise and knowledge that creates a semantic problem, which reduces the chances of success of the systems development.

Additionally, in [6] was verified an increasing in complex systems development and in systems related with other system. It occurs because simple system does not support all stakeholders' needs and different expertise involved in stakeholders' requirements, resulting in the intensification of heterogeneous domains issues. Thus, the cross-domains requirements interoperation issue is to manage the complexity of this heterogeneous knowledge in different systems life cycle phases, ensuring the requirement coherence and compromising the final outcomes.

Thereby, in [17], the authors designed a conceptual multiple view approach model using object oriented model and UML (Unified Modelling Language) to structure information relationships between mechanical and manufacturing domain. Translating mechanisms propitiated the relationship between different domains. Each mechanism dealt with a specific knowledge, which is responsible for translating the information from product view to manufacturing view. Despite the cross-domain approach/solution

presented in this research, the mechanisms were restricted to specific domains. In [18], it was proposed the integration between Model-Driven Engineering (MDE) and Domain Specific Language (DSL), creating a common language and a reasoner to analyse information in multiple domains. DSL formalizes the application structure, behaviour and requirements in a single domain and MDE structures the link between information through reasoning mechanisms in multiples domains. This allowed the exchanging information between heterogeneous domains. However, this approach did not present how to model the domains knowledge in different phases of the system life cycle and the impact of environment changes, which the domain is associated.

In [19], a model-driven domain was proposed and described as part of ontology without axioms and rules. This model provides a common reference point and is used to manage objects development of the system and automatically supports the discovery dependency link. It was limited to early system development life cycle phases (definition and concept) and did not have a mechanism to ensure the consistency of the requirements after the automatic discovery of dependency links. [20] employed MBSE to structure requirements from multiples domains during the system life cycle phases to ensure the requirement consistency. This approach adapted the Vee-model to specific driven to MBSE models supporting the system building. However, this approach did not address the model performance in systems that suffer from frequent requirement changes. In [21], the authors proposed a model-based design (MBD) methodology adapted from MBSE, integrated to the W model proposed by [22] to support the complex system development in multiples domains. For each domain the methodology created a model with their requirements and specific information allowing in a SysML environment the interaction between different domains. The information follows the W model that ensures the consistency of requirements, verification and validation. However, this methodology worked with early phases of the systems life cycle and did not address the requirements control and management in different phases of the systems life cycle.

3.2 Cross-systems life cycle requirements interoperation issue

The systems life cycle phases relate all activities of engineering of system, from definition until retirement as well as rules or verifications to confirm the system maturity [2]. According to [23], there are standards and models (ISO/IEC/IEEE 29148:2011 [24], ISO/IEC/IEEE 15288:2002 [25], etc.) that standardize each phase of systems life cycle and rules that define the evolution and verification of the system. However, for [5] a single view model of the system does not explicitly fit all situation of the system life cycle. According to [5,7,26], these models or standards can be used to determine all phases of the life cycle, but they contain particular characteristics that make them more suitable for specific phases. For instance, the waterfall is suitable for defining phases, because this model uses the feedback concept ensuring and revising the information integrity during a single phase [27]. Moreover, the traditional models ensure the information consistency in the direct flow according to representative life cycle model proposed by ISO/IEC TR 24748-1:2010 [28], i.e., if it is necessary to change some information in previous phases, these models are not able to manage the new information [29]. Nevertheless, the systems life cycle does not follow a linear

progression, i.e., iteration and recursion will occur modifying the life cycle flow as illustrate in Figure 2.

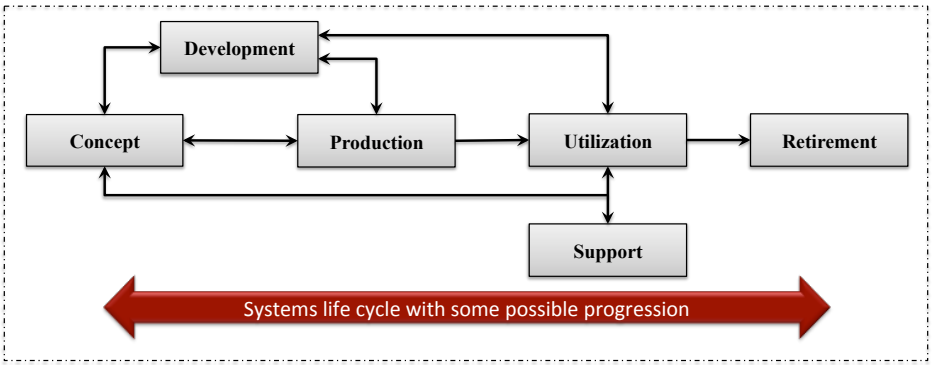

Fig. 2. Life cycle model representation adapted from ISO/IEC TR 24748-1:2010 [28].

In this context, [7] proposed a framework for a model-based requirements engineering to structure the system requirements in a SysML modelling. Moreover, they extend this approach to system life cycle, proposing solutions to integrate requirement in different phases of the system life cycle. However, this approach did not depict the evolution of the requirement in different phases and if the framework is able to analysis the change impact in different phases of the system life cycle. In [30] a methodology to verify the requirement consistency during the system life cycle phases in a dynamic manner is proposed. The methodology, named vVDR (virtual Verification of System Design against System Requirement), contributed to three main steps in the system life cycle phases: system requirement analysis, system design and system testing. Although, the methodology covered different system life cycle phases and analysed the requirements consistency, it did not report if there are consistency checking when the requirements are replaced or if there are impact analyses in different requirements occasioned by their replacement.

In [31], the authors proposed a formalization of semantic annotation for system interoperability from different domains views in a Product Life Cycle Management (PLM) environment. The formalization made explicit the tacit knowledge intrinsic in application models and act to support all activities during the product life cycle. Nevertheless, this approach did not depict the annotations in requirements that change frequently along the life cycle and how to ensure the semantic of these requirements. [32] proposed a model-driven ontology, which integrate the model-driven architecture (MDA) and an ontology, to create a manufacturing system interoperable between design domain and manufacturing domain. The solution emphasized the need of designing the knowledge in a common-logic-based ontology language to allow information exchange between domains. But this solution was limited to two heterogeneous domains and there was no evidence of possibilities to expand information exchange to multi-domains and integrate them.

3.3 Cross-requirements interoperation issue in a single domain/system life cycle phases

Requirements are necessary attribute in a system, a statement that identifies capabilities, characteristics and quality factor of a system to ensure its value and utility for a customer or a user [2,4,5]. Based on literature and standards, there are two main types of requirements: Functional Requirements and Non-Functional Requirements (Quality requirements, Constraints, etc.) [2,5,33]. It is necessary to certify that among requirements will not have problems with completeness, coherency, uniqueness and univocity, as well as the traceability between requirements.

In [12], the complexity of translating customer needs in functional or non-functional requirements is demonstrated, because the customers or stakeholders environment was associated with a different requirement environment than the requirements analyst (RA). Thus, RE emerges as a cooperative, interactive and incremental process to elicitation, negotiation and documentation of the requirements and constraints of complex systems. The RE aims to solve the requirements problem in early stages of the requirements process [3,5,34]. Within RE, beyond elicitation and negotiation, the traceability stands for a relevant problem. Requirements traceability is responsible for tracking information from stakeholder to all level of the engineering of the system as well as providing an understanding about any requirement change [5]. However, requirements traceability relations are not automatic generated and maintained [35, 44] and the identification typically occurs manually [36, 43] making traceability relations susceptible to errors, if changes occur during the engineering of the system [37].

According to [37], the lacks of automated traceability become a prominent problem in complex systems once there is a need to establish traceability between large collections of requirements and other systems documentation. To [38], changes can be required in any phases of the system life cycle (design, implementation or use). However, in Dynamic Adaptive Systems (DAS) a large numbers of requirements are faced changes of environment. Thus, the traditional traceability approach, which works with static and simple system, does not support this new system development once it is necessary to analyse simultaneously the changed requirements, identifying them and tracing the impact of the change in other requirements.

In [39], the authors advocate that if the traceability is consistently maintained it would prevent a dissemination of potential requirements inconsistencies into different system life cycle phases. Thus, according to the authors further researches are necessary to ensure the requirement traceability, making sure that the requirements information is complete, coherent, unique and univocal. According to [40], consistency of requirements can be ensured through validation and verification methods. In this context, [41] proposed an interoperation meta-model to structure the information transforming from stakeholder requirements (problem space) to specifications (solution space). This meta-model was responsible to control the exchange information in collaborative domains, ensuring their consistency and traceability during all this process. However, this meta-model was limited to early phases of the system life cycle and did not ensure the requirement exchange in different phases.

In [35], the authors presented a systematization approach to ensure the requirement consistency in different phases of system life cycle. This approach consists of some mechanisms: (i) mechanism to formalize the requirements and its features, (ii)

mechanisms to consistency checking and (iii) mechanisms to correct the inconsistency problems. Moreover, the authors proposed a mechanism to manage the variability of information in different phases of the system life cycle, its consistency in all system life cycle. The authors did not depict if there are consistency impacts with requirements changing during the system life cycle and if this systematization is able to identify these impacts. In [42], the authors proposed a model to integrate the goal-oriented approach to RELAX, based on KAOS (Knowledge Acquisition in automated specification) and DSL (Domain Specific Language). This model supports the constant requirements changing, but it does not support the requirements evolution during different life cycle phases.

4 Discussion

This research is working to evidence the relevant issues to the requirements engineering in order to ensure all requirements coherency and consistency in all systems life cycle phases. These issues provided support to a conceptual framework proposal that aims to formally model requirements interoperation in term of impact and semantic equivalence or subsuming. Therefore, the authors proposed three dimensions to investigate the related issues: the cross-domain, the cross-systems life cycle phases' and the cross-requirements dimensions.

Related works were found for each dimension regarding requirements engineering and particular solutions proposals. Thus, based on the related works issues/solutions, the Table 1 is proposed, which shows specific analysis by categorization, positioning each paper according to their subjects and degree of importance for the research. The adopted classification criteria were:

- (D1) *Particular cases* – Papers/articles concerning the requirements exchange limited to two specific domains;
- (D2) *Ability to be generic* – Papers/articles concerning the requirements exchange among different domains and that can be adapted to other domains;
- (D3) *Generality of the approach* – Papers/articles concerning the requirements exchange among different domains whose approaches do not need any adaptation;
- (LC4) *Yes* – For papers/articles that concerns the requirement exchange among one or more phases of the system life cycle;
- (LC5) *No* – For papers/articles that do not concern the requirement exchange among one or more phases of the system life cycle;
- (R6) *Requirements Traceability* - Papers/articles regarding the requirements traceability in one or more system life cycle phases and different domains;
- (R7) *Requirements Interoperability* – Papers/articles regarding the exchange of requirements between one or more systems lifecycle phases and different domains. This interoperability issue does not consider any requirements changes during the systems life cycle phases;
- (R8) *Requirements Impacts* - Papers/articles regarding the exchange of requirements between one or more systems lifecycle phases and different domains. This interoperability issue considers the impacts caused by any requirements changes during the systems life cycle phases.

Table 1. Related works classification according to each research issue.

Authors and Publication Year	*Cross-Domains issue*			*Cross-Systems Life Cycle issue*		*Cross-Requirements issue*		
	(D1)	(D2)	(D3)	(LC4)	(LC5)	(R6)	(R7)	(R8)
ADELSON and SOLOWAY, 1985 [45]	✔				✔			
RAMESH and JARK, 2001 [35]					✔	✔		
EGYED and GRÜNBACHER, 2002 [44]					✔	✔		
CLELAND-HUANG et al., 2002 [36]					✔	✔		
CANCIGLIERI JR. and YOUNG, 2003 [17]	✔	✔		✔			✔	
SPANOUDAKIS *et al.*, 2004 [37]					✔	✔		
RATCHEV, URWIN, MULLER, PAWAR and MOULEK, 2003 [12]				✔			✔	
KECECI, GARBAJOSA and BOURQUE, 2006 [43]					✔	✔	✔	
SCHMIDT, 2006 [18]	✔	✔			✔			
STECHERT and FRANKE, 2008 [46]	✔	✔			✔	✔	✔	
WELSH AND SAWYER, 2009 [38]					✔	✔		✔
HOLT and PIERRY, 2010 [7]				✔		✔		
SCHAMAI et al., 2010 [30]				✔		✔	✔	
MONEVA, HAMBERG AND PUNTER, 2011 [15]	✔	✔			✔			
AHMAD and BRUEL, 2012 [42]					✔	✔	✔	✔
BOUFFARON et al., 2012 [41]	✔			✔			✔	
CMYREV *et al.*, 2012 [39]				✔		✔	✔	
STRASUNSKAS and HAKKARAINEN, 2012 [19]	✔	✔			✔	✔		
OERTEL and JOSKO, 2012 [40]					✔	✔	✔	
LIAO *et al.*, 2012 [31]	✔	✔		✔				
CHANDLER and MATTHEWS, 2013 [26]	✔	✔		✔				
CHUNGOORA *et al.*, 2013 [32]	✔	✔		✔			✔	
HAVEMAN and BONNEMA, 2013 [20]	✔	✔	✔	✔				
BARBIERI *et al.*, 2014 [21]	✔	✔		✔				

It is observed in Table 1 that there are some poorly explored gaps: in cross-domain issue, items (D2) and (D3) and in cross-requirement issue, item (R8). In cross-domain issue, it was verified that existing approaches proposed by the literature solve specific information exchange between domains. But, when this approach is extended to multiples domains (more than three), there are strict and/or limited solutions. This issue makes evident the problem with the semantic gap in multiples domains as well as the risk of mistakes and misinterpretation. In cross-requirement issue was noticed that there are researches addressing the requirement traceability and interoperability. Nevertheless, these researches did not consider the impact, which frequents requirements improvements and variations may cause to the consistency and coherency among requirements as well as ensuring the requirement consistency during different systems life cycle phases.

These results represent a preliminary evaluation about the models, frameworks and methodologies found in the literature, concerning the three dimensions. However, it is important to consider that requirements are not static, i.e. requirements' variations, advances and improvements may occur during the system life cycle phases' evolution. Although, the three dimensions consider the inherent relationship to each of them, it is

important address all three issues simultaneously. Therefore, it is necessary explore or develop methodologies to support the systems-of-system, focusing on the three dimensions concurrently. The authors consider this approach as the fourth issue in order to ensuring the system requirements consistency and coherence.

5 Conclusion

This research points towards a conceptual framework for requirements interoperability in complex system engineering in order ensure the system requirements consistency and coherence in all life cycle phases. Requirements are not static, i.e., they may suffer changes, updates or removals during the system life cycle phase's evolution. Thus, it is necessary to manage these relationships to avoid misinterpretation and mistakes with requirements.

The authors proposed four issues to be investigated. Three of them are directly generated from the different presented dimensions (cross-domain, cross-systems life cycle phases and cross-requirements). The last one is the interrelationship among these issues. From these issues, an extensive literature review has been provided and the related works has been classified in order to identify the gaps that were not explored and/or need further researches. Whilst, the literature review and its classification highlighted the gaps in the same issue and/or the relationship among them such as: the need of requirement's language formalization or standardization in order to avoid misinterpretation and mistakes in multiples domains and the impact that frequent requirement changes/updates cause in the system requirements during system life cycle.

The continuity of the research should therefore identify and determine scientific methods identification and determination that are able to conceptually represent these 3 dimensions and the dependencies existing among them. It should also explore how to cope with the impact of requirements changes can cause among requirements in multiples domains during the system life cycle phases.

References

1. Mallek, S., Daclin, N., Chapurlat, V.: The application of interoperability requirement specification and verification to collaborative process in industry. Computers in Industry. 63, 643-658 (2002).
2. INCOSE, INCOSE Systems Engineering Handbook: A Guide for Life Cycle Processes and Activities, *The International Council on Systems Engineering*, C. Haskins, 3 ed. (2006).
3. Young, R.R.: The Requirements Engineering Handbook, 1 ed, Artech House, Boston (2004).
4. AFIS C.: Guide Bonnes Pratiques en Ingénierie des Exigences. 1ed, Cépadules, Paris (2012).
5. Sebok, Guide to the Systems Engineering Body of Knowledge (SEBoK), version 1.2, (2013). http://www.sebokwiki.org.
6. INCOSE, INCOSE Systems Engineering Vision 2020, *The International Council on Systems Engineering*, C. Haskins, 2 ed., (2007).
7. Holt, J., Perry S., Brownsword M.: Model-Based Requirement Engineering, 1 ed., pp. 340 The Institution of Engineering and Technology, London (2011).

8. OBJECT MANAGEMENT GROUP (OMG): MDA Guide, Version 1.0.1 (2003).
9. Colombo P., Khendek, F., Lavazza L.: Bridging the gap between requirements and design: An approach based on Problem Frames and SysML. Journal of Systems and Software. 85, (3), 717–745 (2012).
10. Flanigan, D., Brouse P.: System of Systems Requirements Capacity Allocation. Procedia Computer Science, 8, 112–117 (2012).
11. Brooks, R.T., Sage A.P.: System of System Integration and Test. Information Knowledge and Systems Management. 5, 261-280, (2006).
12. Ratchev, S., Urwin, E., Muller, D., Pawar K.S., Moulek L.: Knowledge based requirement engineering for one-of-a-kind complex systems. Knowledge-Based Systems. 16, (1), 1–5 (2003).
13. Bernard, Y.: Requirements management within a full model-based engineering approach. Systems Engineering. 15, 119-139 (2012).
14. Bjørner, D.: From Domains to Requirements in Concurrency. Graphs and Models: Essays dedicated to Ugo Montanari on the Occasion of his 65th Birthday, pp. 300, Springer (2008).
15. Moneva, H., Hamberg, R. Punter, T.: A Design Framework for Model-Based Development of Complex Systems. In: 32nd IEEE Real-Time Systems Symposium and 2nd Analytical Virtual Integration of Cyber-Physical Systems Workshop, pp. 1-8, Vienna, (2011).
16. Vitharana, P., Jain H., Zahedi, F.M.: A knowledge based component/service repository to enhance analysts' domain knowledge for requirements analysis. Information & Management. 49, 1, 24–35 (2012).
17. Canciglieri Jr. O., Young R.I.M.: Information sharing in multiviewpoint injection moulding design and manufacturing. International Journal of Production Research. 41, (7), 1565-1586 (2003).
18. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer. 39, (2), 25-31 (2006).
19. Strasunskas, D., Hakkarainen, S.E.: Domain model-driven software engineering: A method for discovery of dependency links. Information and Software Technology. 54, (11), 1239–1249 (2012).
20. Haveman, S.P., Bonnema G.M.: Requirements for High Level Models Supporting Design Space Exploration in Model-based Systems Engineering. Procedia Computer Science. 16, 293–302 (2013).
21. Barbieri, G., Fantuzzi, C., Borsari, R.: A model-based design methodology for the development of mechatronic systems. Mechatronics – In Press, (2014).
22. Nattermann, R., Reiner, A.: Approach for a data-management-system and a proceeding-model for the development of adaptronic systems. In: International mechanical engineering congress and exposition (ASME), pp. 1-10, Vancouver (2010).
23. Schneider, F., Berenbach, B.: A Literature Survey on International Standards for Systems Requirements Engineering. Procedia Computer Science. 16, 796-805 (2013).
24. International Organization for Standardization: ISO/IEC/IEEE 29148:2008 – Systems and software engineering – Life cycle processes and Requirement Engineering, ISO/IEC (2011).
25. International Organization for Standardization: ISO/IEC/IEEE 15288:2002 – Systems and software engineering – System life cycle processes, ISO/IEC (2002).
26. Chandler S.R., Matthews P.C.: Through-Life Systems Engineering Design & Support with SysML. Procedia CIRP. 11, 425–430 (2013).
27. Ruparelia, N.B.: Software Development Lifecycle Models. ACM SIGSOFT Software Engineering Notes. 35, (3), 8-13 (2010).
28. International Organization for Standardization: ISO/IEC TR 24748:2010 – Systems and software engineering – Life cycle management – Part 1: Guide for life cycle management, ISO/IEC (2011).
29. Gausemeier, J., Gaukstern, T., Tschirner, C.: System Engineering Management Based on a Discipline-Spanning System Model. Procedia Computer Science. 16, 303-312 (2013).

30. Schamai, W., Helle, P., Fritzson, P., Paredis, C.J.J.: Virtual Verification of System Designs against System Requirements. In: Models 2010 ACES-MB Workshop Proceedings. 1, 53-67, (2010).
31. Liao, Y., Lezoche, M., Loures, E.F.R., Panetto H., Boudjlida, N.: Formalization of Semantic Annotation for System Interoperability in a PLM environment. In: Proceeding of OTM Federated conferences and workshops and 2nd workshop on Industrial and Business Application of Semantic Web Technologies (INBAST), pp. 1-7, (1), Rome (2002).
32. Chungoora, N., Young, R.I., Gunendran, G., Palmer C., Usman, Z., Anjum, N.A., Cutting-Decelle A.F., Harding, J.A., Case, K.: A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. Computers in Industry. 64, (4), 392–401 (2013).
33. Insitute of Electrical and Electronics Engineers, IEEE Std 830-1998 - Recommended Practice for Software Requirements Specifications, IEEE Computer Society, New York (1998).
34. Pohl, K.: The three Dimensions of Requirements Engineering: A framework and its applications. Informatic Systems. 19, (3), 243-258 (1994).
35. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Transactions on Software Engineering. 27, 58-93 (2001).
36. Cleland-Huang, J., Chang, C.K., Sethi, G., Javvaji, K., Haijian, H., Jinchun, X.: Automating speculative queries through event-based requirements traceability. In: Proceedings of IEEE Joint International Conference on Requirements Engineering, pp.289-296 (2002).
37. Spanoudakis, G., Zisman, A., Pérez-Miñana, E., Krause P.: Rule-based generation of requirements traceability relations. Journal of Systems and Software. 72, (2), 105–127 (2004).
38. Welsh, K., Sawyer, P.: Requirements tracing to support change in dynamically adaptive systems. Requirements Engineering: Foundation for Software Quality. pp. 59-73, Springer Berlin Heidelberg, Berlin (2009).
39. Cmyrev, A., Noerenberg, R., Hopp, D., Reissing, R.: Consistency Checking of Feature Mapping between Requirements and Test Artefacts. CESAR Project. 1, 1-12 (2012).
40. Oertel, M., Josko B.: Interoperable Requirements Engineering: Tool Independent Specification, Validation and Impact Analysis. In: Embedded World 2012 Exhibition and Conference, pp. 3-7, Nuremberg (2012).
41. Bouffaron, F., Gouyon, D., Dobre D., Morel G.: Revisiting the interoperation relationships between System Engineering collaborative processes. In: 14th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1-6, INCOM2012, Romania (2012).
42. Ahmad, M., Bruel, J.M., Laleau, R., Gnaho, C.: Using RELAX, SysML and KAOS for Ambient Systems Requirements Modeling. Procedia Computer Science. 10, 474-481 (2012).
43. Kececi, N., Garbajosa, J., Bourque, P.: Modelling functional requirements to support traceability analysis. IEEE International Symposium on Industrial Electronics, 4, 3305-3310 (2006).
44. Egyed, A., Grunbacher, P.: Automating requirements traceability: Beyond the record & replay paradigm. In: 17th IEEE International Conference on Automated Software Engineering, pp. 163-171, IEEE press, New York (2002).
45. Adelson, B., Soloway, E.: The Role of Domain Expenence in Software Design. IEEE Transactions on Software Engineering. 11, 1351-1360 (1985).
46. Stechert, C., Franke, H.J.: Managing requirements as the core of multi-disciplinary product development. CIRP Journal of Manufacturing Science and Technology. 1, (3), 153–158 (2009).