View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Open Archive Toulouse Archive Ouverte

OATAO
Open Archive Toulouse Archive Ouverte

Open Archive Toulouse Archive Ouverte

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in: https://oatao.univ-toulouse.fr/22253

Official URL
https://doi.org/10.1109/REW.2017.38

To cite this version:

Galinier, Florian and Bruel, Jean-Michel and Ebersold, Sophie and Meyer, Bertrand *Seamless Integration of Multirequirements in Complex Systems.* (2017) In: 25th International Requirements Engineering Conference Workshops (REW 2017), 4 September 2017 - 8 September 2017 (Lisbon, Portugal).

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Seamless Integration of Multirequirements in Complex Systems

Florian Galinier, Jean-Michel Bruel, Sophie Ebersold, Bertrand Meyer†
IRIT, University of Toulouse
Toulouse, France
{Firstname.Lastname}@irit.fr
†Also Software Engineering Lab, Innopolis University, Russia and Politecnico di Milano, Italy
Bertrand.Meyer@inf.ethz.ch

***Abstract*—Requirements are the keystone of complex systems development. In order to reduce inconsistencies, requirements analysis is an important issue of systems engineering. In this context, there is a need for conciliating views of several stakeholders from different domains and for tracing these requirements from specification to realization. The computerization of analysis, with the help of a clearly defined semantics linked to a non-specialist readable language, should lead to overcome this major issue. Several works already go into this direction. The most popular ones are dealing with natural language, easily understandable but with few semantics. Other approaches propose more formal notations, with stronger semantics but then being less affordable by stakeholders. In this paper, we propose a preliminary work that should drive us to define a language dedicated to requirements which combine the best of both worlds in order to ease requirements analysis throughout the system lifecycle.**

I. Introduction

The design of complex systems implies several stakeholders from different domains. Due to this heterogeneity of skills the description of these systems is done with different artifacts (e.g., text documents, requirements databases, models, etc.). One of the main challenge of Systems Engineering (SE) is to be able to define and maintain relationships between these different artifacts.

Indeed, it still exists a lack of coherence between the several views – the different artifacts used to specify the system. Using these unrelated views makes inconsistencies detection harder, such as conflicting requirements. A multiviews approach, with a dedicated unique language or with a common abstraction of specifications' artifacts, would allow to detect inconsistencies upstream.

A traceability problem between design and system realization also exists. The lack of clear correlation between system implementation and requirements does not help handling consequences of requirements modifications. Multirequirements [1] aims to interweave specifications and development in order to reduce the gap between requirements and system implementation in a seamless purpose. The introduction of the concept of multirequirements allows to make the link between several levels of abstraction in order to compute the impact of changes.

Moreover, using a common language for specification and design would help us to easily add new requirements (induced from the system decomposition for example) in the set of existing ones.

Model-Based Systems Engineering (MBSE), which is increasingly used in SE, uses models as central development artifacts. Modeling provides a possible way to define a common interface between different views and abstractions. The main goal of this work is to define methods and tools in order to allow a seamless integration of requirements in these two dimensions. One of the expected contributions is the use of MBSE to express requirements that would be used as a common interface between artifacts.

This paper is organized as follow: section II exposes the major issues of requirements in SE. Section III explores different approaches that aim to interweave artifacts from different formalisms. Section IV introduces requirements formalization and approaches that aim to link systems' specifications with their requirements. In section V, we expose a preliminary approach to combine these two viewpoints in a single one and our planned contributions. Finally, we summarize our viewpoint in section VI.

II. Major issues of requirements in SE

The requirements' analysis primary goal is to ensure the quality of future system. By tracing requirements and system, engineers can check that the system does the right thing (implements the expected behavior), which is referred to as *validation*; and does it right, which is referred to as *verification*. Requirements analysis is part of the Verification and Validation (V&V) process as defined by the IEEE standard 1012-2012 [2]. In order to validate the system's compliance to the stakeholders' requirements at each step, these requirements should be refined in a technical specification of the system. Due to space limitation, the differences between requirements and specification are not developed here, but let us remind that they both describe the "what" a system should do, rather than the "how" it should do it.

Natural Language (NL) is the most common and easy way to express requirements. Its main quality is its universality. It is the common language of all stakeholders and such requirements are human-readable. Nevertheless, some technical parts of the system need to be described with more specific notations – electrical engineers for example should prefer mathematical

DOI 10.1109/REW.2017.38

notation and formulas. The major issue of NL is its ambiguity. This makes the analysis more difficult and has led to famous failures [3], [4].

The ISO/IEC/IEEE 29148:2011 standard [5] defines a number of necessary qualities for requirements expression (requirements should be traceable, verifiable, consistent, unambiguous, etc.) to ease the analysis of requirements. The objective of applying these qualities to requirements is to provide an easier set of requirements to analyze. In the following section, we describe several works that target these objectives.

III. Requirements expression in several views

The International Council on Systems Engineering (INCOSE) highlights the need to conciliate several stakeholders' viewpoints [6]. Our approach is in line with their recommendations. On one hand, using a unique language to express requirements from all the heterogeneous domains is unrealistic by force of habits and the number of different domains and stakeholders involved in nowadays systems. On the other hand, to address the needs for software quality it is sound to target one unique underlying semantics for artifacts manipulation (see Fig. 1). Before exploring our approach in section V, let us explore some existing languages for requirements representation.

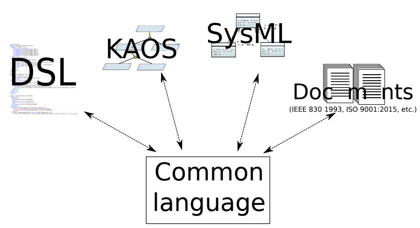

Figure 1. Using a common language between several domains languages

A. Natural language tools

Nowadays, there are numerous industrial solutions [7]. Among the most popular, *IBM Rational DOORS*[1] or *Dassault Systems' Reqtify*[2] provide tools to manage requirements in complex systems. These solutions allows users to make relations between requirements (both functional or not) expressed in several ways to introduce traceability into the systems. For example, in Reqtify, you can import requirements expressed in a Microsoft Word document and link them to some C code that implements these specifications. These tools provide a way to define relationships between requirements as well as relationships between requirements and other artifacts, but does not provide a strong semantic for these links.

The Goal-Oriented Requirements Engineering (GORE) KAOS approach [8] also provides a way to express requirements in natural language and relationship between them, but with a stronger semantics on those relationships. It provides a modeling approach to describe dependencies between requirements. Some tools like Objectiver[3] allow to express user requirements and to refine them using the KAOS approach. It also allows to link requirements artifacts with user requirements specification documents.

Systems Modeling Language (SysML) [9] provides a diagram type to express requirements – requirements diagram – that leads to the incorporation of the specification into the modeling process. While the standard use case diagrams can lead users to express functional requirements, this new type of diagram permits users to express both functional and non-functional requirements (as an element with text) and to define relationships between requirements or between requirements and other diagrams' elements (like blocks, uses cases, etc.). SysML does not offer the possibility to link requirements expressed with other languages unlike the previously presented tools, but the PolarSys[4] development tool for complex systems provides a plugin, named ReqCycle[5], which can reference other specification documents.

Tools introduced in this section support the notion of traceability between requirements. Nevertheless, they mostly work only with textual-only requirements. In the most advanced of them, it is possible to link artifacts with concrete documents (e.g., ReqCycle). The relationships' inference is made harder by the use of natural language and still requires human expertise to translate links expressed in NL and then to analyze the requirements – e.g., to detect inconsistencies.

B. Model-driven approaches

To overcome this issue, the use of a common interface between several views is needed to provide a way to link different artifacts. The Model Driven Engineering (MDE) proposes to use models as a central artifact that can act like an interface. For example, the Generic Model of Computation (GEMOC) initiative [10] aims to provide a common interface for different *Domain Specific Languages* (DSL) used to express the specific needs to different stakeholders. They propose to use models as base artifacts to bridge the gap between DSLs in the same way that MDE uses models as a central artifact between specification and implementation. Moreover, the acceptance of this approach can be eased by the growth of interest of MBSE in system engineering industry.

A MBSE approach would be to express requirements as modeling elements as precisely as other modeling artifacts. This approach can lead us to create links between requirements and other modeling artifacts from several stakeholders. Thus, elements from several domains can be combined in a holistic view, taking into account links between domain specific artifacts and also between these artifacts and requirements.

This is the approach proposed by [11]. Indeed, contrary to more traditional MDE approaches, they propose to use virtual models. Stakeholders' models can be seen as technical spaces used to express specific needs of a domain. There are federated in a common space which is used to make links between interfaces of different domains. Thus, they mapped

[1] http://www-03.ibm.com/software/products/en/ratidoor
[2] http://www.3ds.com/products-services/catia/products/reqtify/
[3] http://www.objectiver.com/
[4] https://www.polarsys.org/
[5] https://www.polarsys.org/projects/polarsys.reqcycle

concepts expressed with several paradigms (EMF, XML, Word documents, etc.) in a common interface. For example, this should allow to link requirements expressed in NL in a Word document to requirements expressed with the KAOS methodology or even with more formal methodologies.

IV. Requirements formalization

Traceability between specification and requirements is an important issue of requirements analysis. Indeed, the ability to link parts of the system with requirements helps to determine if they comply with its requirements and if it is a necessary part. This traceability can be eased by using a dedicated requirements language. Indeed, this kind of language, which is more formal than NL, is a possible way to ensure necessary qualities for requirements expression.

A. Requirements expression

The use of requirements dedicated languages has been studied several times. However, in the approaches presented so far, the NL is still used to express requirements that are consequently ambiguous. Some works try to overcome this issue by proposing a constrained form of NL.

In [12], the authors propose a grammar for an English subset. This constrained language can lead to the avoidance of inherent problems of NL. Indeed, its syntax leads the requirements expression during the elicitation phase and allows to capture component elements from the requirements.

In a similar way, Hähnle et al. [13] propose an interface between another English subset and Object Constraint Language (OCL). OCL is a formal language used to express constraints in Unified Modeling Language (UML) diagrams. This work can be seen as a way to formalize requirements from NL to a language with a stronger semantic. This should let non-experts to express requirements in an understandable way, whereas the OCL representation should allow them to analyze the system. An example for a `Queue` class is given in Fig. 2; preconditions and postconditions of `Queue::getFirst()` operation can be expressed both in English or OCL, and are automatically translated in the other language.

```
Operation  getFirst
OCL:  context Queue::getFirst() : Integer
      pre:  self.size() > 0
      post: result = self.asSequence()
                                ->first
English: for the operation getFirst() : Integer of the class Queue,
      the following precondition should hold:
         the size of the queue is greater than zero
      and the following postcondition should hold:
         the result is equal to the first element of the queue.
```

Figure 2. Example from [13] of a matching between English and OCL

While these approaches should permit to bridge the gap between a language that can be used by non-specialist and a more formal representation of requirements, a major issue appears. Indeed, there is a need for maintaining coherence between these two representations. The change should be propagated on both formalisms, and is not as immediate than with a unique language.

B. Requirements specific languages

To overcome this issue, some works highlight languages dedicated to requirements expression based on a formal semantic.

In [14], the authors propose a language to express requirements in a Complex Adaptive System (CAS) context. For this Witthle et al. present a structured natural language, named RELAX, that allows to specify requirements with some of them that can be relaxed in order to keep safe priority requirements of the CAS. This language is close to NL but it is based on formal methods. It is semantically defined with fuzzy branching temporal logic [15]. These semantics can be used as a validation basis through the benefit of validation tools. Recently, [16] proposed an extension for the Modelica [17] modeling language, named *FORM-L*, to allow formal modeling of requirements.

These languages are designed to be addressed to stakeholders.

However, these approaches are not, according to us, simple enough to be widely used (compared for example to agile user stories [18], more non-specialist readable but not formal). Furthermore, these works were developed for dedicated domains and thus are very specific DSLs.

C. Formal expression of requirements

Formal methods are widely used to express specifications and systems in order to prove their correctness. By nature, they are not addressed to non-specialist stakeholders. Though some works try to link these methods with some less formal representations of requirements.

In [19], the authors propose a translation method from NL requirements to a formal representation. They propose a dedicated intermediate language which can be formalized in OWL [20]. Nevertheless, the authors themselves admit that their language is addressed to requirements engineers. Non-specialist stakeholders, without any domain-knowledge, are not able to understand this formalism.

The authors of [21] proposed to translate requirements expressed in KAOS to the Event-B formal method. For this, they used KAOS relationships (refinement, composition, ...) to infer semantic links between formal representations of requirements. For example, a requirement composed of other requirements will be translated as an `AND` association of component requirements.

In [22], the authors are focusing on the system itself. They aim to link requirements and specification in a unique formalism. This leads to the reduction of issues due to the gap between requirements and specification. The use of a formal syntax can lead users to prove the correctness of the system and, moreover, to validate it – you can prove that the system respects the requirements.

This last approach aims to link requirements and specifications in a unique language. The use of a unique paradigm has been proposed by Paige and Ostroff in [23]. This idea of a single model aims to express requirements, specifications and the implementation in order to avoid the natural gap existing between several formalisms.

This is the approach proposed in [1]. In this paper, the idea is to use the expressiveness of the Eiffel language [24] to express different views of a system. Indeed, the objective of design by contracts [25] is to introduce within the software's code the notion of preconditions (requirements of a routine), postconditions (properties ensured after a routine execution) and invariants (logical expressions always true) in an object-oriented context. Expressing requirements inside contracts allows to directly check the validity of the system and detect a lack of consistency, with the help of tools such as AutoProof [26] – a verifier for Eiffel. Moreover, the author proposes to directly link requirements expressed in different formalisms (natural language, diagram) inside the programming code. This approach should help users to find informations about the source of a piece of the program and to help traceability from specification to realization.

V. Expected contributions

To allow the introduction of seamlessness in complex systems development, a number of avenues for research must be explored. Indeed, in order to produce a methodology and tools that can be used in a real industrial context, our contribution should be easy to handle and as close as possible to languages and tools used by engineers.

Nevertheless, the use of NL as a way to express requirements leads us to ask several questions:

- How can requirements be expressed in a non-specialist readable way, while still being computerizable?
- How to make links between requirements expressed by several stakeholders?
- What is the semantic of requirements relationships? Of relationships between requirements and the system?
- How to use a requirements formalization to prove their properties (soundness, completeness, etc.)?

Works previously mentioned introduced some avenues of research. However none of them gives answers to all these questions. One of our main objective will be to propose a unique paradigm that can conciliate these two visions – multiviews and multirequirements. We also aim to provide tools to assist requirements engineers in quality control and the system validation with help from techniques such as: traces, requirement coverage (in the same sense modern tools can provide test coverage), formalization, etc.

Another important goal is to provide tools usable in a natural way or at least a way close to industrial practices. Indeed, requirements concern both technical team and non-specialist stakeholders from several domains. The proposed approach and tools should help them to conciliate their viewpoints.

Firstly, a requirements language will be proposed. It will be close to NL, allowing to extract requirements concepts into a requirements model. These requirements artifacts will be formalized into the Eiffel language in order to interweave requirements, specification and implementation. This language allows us to conciliate the power of proof – with a verifier – with an executive language that can be used for simulation for example. Moreover, the Eiffel language interweave in a single paradigm both the programming language and modeling language. It also supports a mechanism named Eiffel Information System (EIS) which allows to add links to other paradigms such as Word documents. This can be used to enact the process from requirements to implementation in a seamless manner.

In order to experiment this approach, a landing gear use case is currently explored. Proposed in [27], it provides a realistic system and its requirements. This example was treated by several formalization works that could be used to compare our approach in ABZ2014 conference [28].

```
r21
    -- When the command line is working (normal mode), if
    -- the landing gear command handle remains in the DOWN
    -- position, then retraction sequence is not observed.
    note
      EIS: "name=URD", "protocol=URI",
      "src=/path/to/URD.pdf", "nameddest=R21"
    require
      handle_status = is_handle_down
    do
      main
    ensure
      gear_status /= is_gear_retracting
end
```

Listing 1. Eiffel representation of R21 requirements

An extract of the Eiffel representation of requirement `R21` from this use case is given List. 1. This requirement is linked to its NL form – given in the comment – through the EIS. EIS provides a way to link the Eiffel representation of requirements to the user requirements document directly inside EiffelStudio – the Eiffel main Integrated Development Environment (IDE). Requirements are expressed with Eiffel contracts. Preconditions (`require` part of the code) can be used to express the state that the system should reach to check the requirement (the landing gear command is in `DOWN` position), while postconditions (`ensure` part of code) can be used to ensure that the requirement is respected (the gear is not retracting). The routine body (the `do` part of code) should provide the implementation of the requirement.

One objective will be to extract a basic specification and requirements expressed in (as close as possible to) NL and to transform them into an Eiffel representation. `handle_status`, `main` and `gear_status` features in List. 1 are part of specification model – not represented here –, while the contracts are used to express the requirement itself. In an incremental way, the obtained model could be enriched. These early models can then be used for a simulation purpose or for a software implementation. Moreover, the Eiffel formal representation of requirements should lead us to analyze requirements – with the help of Autoproof and techniques such as proof by contradictions.

In a second time, we will introduce translation schemes from several viewpoints to our abstraction of requirements. We aim to provide an interface between a formal representation of requirements (in Eiffel) and NL – the Requirements-Specific Modeling Language (RSML) in Fig. 3. The expressiveness of

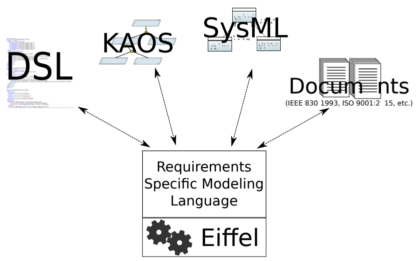

Figure 3. Multiviews dimension: create translation links between common tools and an interface (the RSML) with more formal language (Eiffel).

Eiffel could lead us to provide an embedded DSL. However, we would also like to propose a more abstract DSL – an external DSL –, more affordable to non-software engineers.

This approach could allow to conciliate several viewpoints and formalisms. Indeed, the more technical parts of the system could be expressed with specific tools addressed to specialists, while requirements will be addressed through a common language. The objective is to ease the communication between specialists of different domains.

Thereafter, the approach will be validated through a real industrial case.

VI. Conclusion

In the world of complex systems, taking multirequirements into account is critical. It is important to propose usable tools, therefore user-centric, that support requirements engineering as a whole. We intend to introduce an approach allowing a seamless integration of multirequirements. We believe that the Eiffel language and the associated formal verifiers and tools should allow, first to express requirements in a non-specialist readable way, then to prove requirements properties, and finally to facilitate automation of requirements. We first checked our proposal on a current case study and we got interesting results.

References

[1] B. Meyer. Multirequirements. *Modelling and Quality in Requirements Engineering (Martin Glinz Festscrhift)*, 2013.

[2] IEEE Standard for System and Software Verification and Validation. *IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004)*, pages 1–223, May 2012.

[3] P. G. Neumann. *Computer-related risks*. Addison-Wesley Professional, 1994.

[4] F. Modugno, N. G. Leveson, J. D. Reese, K. Partridge, and S. D. Sandys. Integrated safety analysis of requirements specifications. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 148–159. IEEE, 1997.

[5] ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, pages 1–94, December 2011.

[6] INCOSE. *SE Vision 2025*. 2014. http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf.

[7] J. M. Carrillo de Gea, J. Nicolás, J. L. F. Alemán, A. Toval, C. Ebert, and A. Vizcaíno. Requirements Engineering Tools. *IEEE Software*, 28(4):86–91, July 2011.

[8] A. van Lamsweerde. Goal-oriented requirements engineering: a guided tour. In *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, pages 249–262, 2001.

[9] Object Management Group (OMG). OMG Systems Modeling Language (OMG SysML™), V1.0, 2007. OMG Document Number: formal/2007-09-01 Standard document URL: http://www.omg.org/spec/SysML/1.0/PDF.

[10] B. Combemale, J. Deantoni, B. Baudry, R. B. France, J.-M. Jézéquel, and J. Gray. Globalizing Modeling Languages. *Computer*, pages 10–13, June 2014.

[11] F. R. Golra, A. Beugnard, F. Dagnat, S. Guerin, and C. Guychard. Continuous Requirements Engineering using Model Federation. *RE:Next! Track at 24th IEEE International Requirements Engineering Conference 2016*, 2016.

[12] W. Scott and S. C. Cook. *A Context-free Requirements Grammar to Facilitate Automatic Assessment*. PhD thesis, UniSA, 2004.

[13] R. Hähnle, K. Johannisson, and A. Ranta. An Authoring Tool for Informal and Formal Requirements Specifications. In Ralf-Detlef Kutsche and Herbert Weber, editors, *Fundamental Approaches to Software Engineering*, number 2306 in Lecture Notes in Computer Science, pages 233–248. Springer Berlin Heidelberg, April 2002.

[14] J. Whittle, P. Sawyer, N. Bencomo, B. H. C. Cheng, and J. M. Bruel. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *2009 17th IEEE International Requirements Engineering Conference*, pages 79–88, August 2009.

[15] S. Moon, K. H. Lee, and D. Lee. Fuzzy branching temporal logic. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):1045–1055, April 2004.

[16] T. Nguyen. Verification of Behavioural Requirements for Complex Systems with FORM-L, a MODELICA Extension. In *26th ICSSEA*, EDF R&D, 6 quai Watier, 78110 Chatou, FRANCE, 2015.

[17] S. E. Mattsson, H. Elmqvist, and M. Otter. Physical system modeling with Modelica. *Control Engineering Practice*, 6(4):501–510, April 1998.

[18] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*, 51, Part B:915–929, October 2015.

[19] F.-L. Li, J. Horkoff, A. Borgida, G. Guizzardi, L. Liu, and J. Mylopoulos. From Stakeholder Requirements to Formal Specifications Through Refinement. In Samuel A. Fricker and Kurt Schneider, editors, *Requirements Engineering: Foundation for Software Quality*, Lecture Notes in Computer Science, pages 164–180. Springer International Publishing, March 2015.

[20] S. Bechhofer. OWL: Web Ontology Language. In LING LIU and M. TAMER OZSU, editors, *Encyclopedia of Database Systems*, pages 2008–2009. Springer US, 2009.

[21] A. Matoussi, F. Gervais, and R. Laleau. An Event-B formalization of KAOS goal refinement patterns. Technical Report Tech. Rep. TRLACL-2010-1, LACL, University of Paris-Est, 2010.

[22] A. Mammar and R. Laleau. On the Use of Domain and System Knowledge Modeling in Goal-Based Event-B Specifications. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, number 9952 in Lecture Notes in Computer Science, pages 325–339. Springer International Publishing, October 2016.

[23] R. Paige and J. Ostroff. The Single Model Principle. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, pages 292–, Washington, DC, USA, 2001. IEEE Computer Society.

[24] B. Meyer. *Eiffel: The Language*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[25] B. Meyer. Applying 'design by contract'. *Computer*, 25(10):40–51, October 1992.

[26] J. Tschannen, C. A. Furia, M. Nordio, and N. Polikarpova. AutoProof: Auto-Active Functional Verification of Object-Oriented Programs. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, number 9035 in Lecture Notes in Computer Science, pages 566–580. Springer Berlin Heidelberg, April 2015.

[27] F. Boniol and V. Wiels. The Landing Gear System Case Study. In F. Boniol, V. Wiels, Y. Ait-Ameur, and K.-D. Schewe, editors, *ABZ 2014: The Landing Gear Case Study*, number 433 in Communications in Computer and Information Science, pages 1–18. Springer International Publishing, June 2014.

[28] F. Boniol, V. Wiels, Y. Ait-Ameur, K.-D. Schewe, S. D. Junqueira Barbosa, P. Chen, A. Cuzzocrea, X. Du, J. Filipe, O. Kara, I. Kotenko, K. M. Sivalingam, D. Slezak, T. Washio, and X. Yang, editors. *ABZ 2014: The Landing Gear Case Study*, volume 433 of *Communications in Computer and Information Science*. Springer International Publishing, Cham, 2014.