Identifying Suitable Representation Techniques for the Prioritization of Requirements and Their Interdependencies for Multiple Software Product Lines

Stephanie Lewellen(✉) and Markus Helfert

Dublin City University, Glasnevin, Dublin 9, Ireland
stephanie.lewellen2@mail.dcu.ie, markus.helfert@dcu.ie

Abstract. Software requirements typically do not exist independently of each other, rather most requirements have some type of dependency on another requirement [4]. For companies developing software products, which depend on each other, in so-called multiple software product lines (SPLs), systematic requirements management, including consideration for prioritization and interdependencies, is a time-consuming and convoluted task. Representation techniques for complex requirements can convey critical requirements interdependency information to make prioritization of requirements quicker and more accurate [1]. Based on reviewing the foremost literature, this paper identifies the representation techniques for requirements management which are most suitable for multiple software product lines (SPLs).

Keywords: Requirements · Prioritization · Multiple software product lines
Interdependencies

1 Introduction

Software release planning is a critical decision-making process which aims to find an optimal subset of software requirements, in which the stakeholders are satisfied while the resource and timeline constraints are met [2, 3]. Software release planning is one of the most important and complex tasks within the practices of requirements engineering, because requirements usually have many dependencies and it is not possible to select requirements based on individual priority alone [4].

One of the catalysts to increased complexity in requirements dependencies is modern software product line engineering, which capitalizes on the recycling of software between software products [5]. In the simplest form, the software product line (SPL) has one common code base, but with two or more higher level code additions resulting in their own software products [5, 6]. In recent years, with even more variation, the concept of SPLs has been extended to offer more software product variation in the form of multiple SPLs [5, 6].

For multiple SPLs, requirements interdependencies which influence the cost of development and the value to the stakeholder could be of particular interest, because if

© Springer International Publishing AG, part of Springer Nature 2018
W. Abramowicz and A. Paschke (Eds.): BIS 2018, LNBIP 320, pp. 412–423, 2018.
https://doi.org/10.1007/978-3-319-93931-5_30

they are not met, the original benefit of diversifying the software product portfolio has not been completely realized.

This paper considers the representation techniques in the prioritization of requirements which have interdependencies between SPLs. Focus is given to the interdependencies of requirements which are most critical to market-driven software [1] – requirements related to value and cost. It starts with an overview of the current research in the areas of multiple SPLs and requirements management, including prioritization of requirements and requirements interdependencies. Following the overview is a summary of five representation techniques for the prioritization of requirements and their interdependencies. Finally, the paper concludes with the authors' assessment of the five representation techniques based on requirement prioritization, requirement interdependencies, and the consideration of cost/value interdependencies between SPLs.

2 Overview

This overview delves deeper into the subject areas and foremost research of multiple SPLs and managing software requirements, including the prioritization of requirements and assessment of requirement interdependencies.

2.1 Multiple Software Product Lines (SPLs)

Much consideration has been given to requirements management for single software products, but far less consideration has been given to requirements between software product lines [7] or to particular domains [8].

A software product line (SPL) is a set of software products sharing a set of common features but containing variation points [9]. One of the advantages to SPLs is the reduced cost of development and testing, with an increased opportunity to address different stakeholder groups [7]. An example for a single SPL, given by Rosenmüller et al. [10], is mail client software which relies on a common mail framework. Variations of the mail client software, to support different protocols, for example, would all rely on the same common mail framework [10].

Multiple SPLs, in comparison, are composed of many interconnected subsystem versions and variants [6]. Multiple SPLs commonly refers to vertically tiered software stacks, with application SPLs and infrastructure SPLs [5], but can also refer to distributed SPLs, as in the case of sensor software [10]. An example of vertically tiered multiple SPLs would be individual application software product lines that all rely on the same database platform, which is itself a software product line [5, 6] (Fig. 1).

The analysis that follows in this paper focuses on requirements representation techniques and their application to requirements prioritization and requirements interdependencies between multiple SPLs, specifically vertically tiered SPLs where the upper tier (application) software requires software functionality from the lower (infrastructure) tiers.

Furthermore, the focus in this paper is on mature products on the market, with development approach that is incremental and market-driven, as in the industrial survey conducted by Carlshamre et al. [1]. The focus has been restricted to market-driven

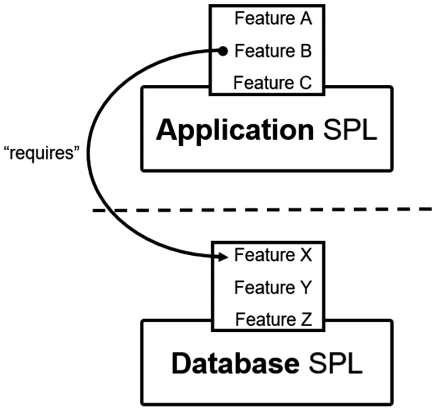

Fig. 1. Dependency model between an application SPL and an infrastructure SPL [5]

software, because most requirements interdependencies for this type of software are of the "cost/value" type [1].

2.2 Managing Software Requirements: Prioritization and Interdependencies

It is not possible to address all functional and non-functional requirements in the next software release, due to constraints from budget, resources or time [7]. Therefore, a prioritization of requirements should be applied [8].

Prioritization Consideration. There are a number of requirements prioritization techniques [11, 12] and the research to the application of these processes to the software release planning process is extensive. However, even the most suitable processes for complex requirement prioritization, like the analytic hierarchy process (AHP), do not consider the interdependency of requirements [8, 12].

Interdependency Consideration. According to Dahlstedt et al., *"despite the need for and potential benefits of systematically taking requirements interdependencies into account, there is little research invested in this topic and more is needed"* [13]. The existing processes [12] provide potentially inaccurate weighting of requirements due to these shortcomings [14].

Carlshamre et al. [1] noted in the result of their 2001 industrial survey that approximately 20% of the requirements assessed in their survey had 75% of the total requirement dependencies. In order to reduce an otherwise time-intensive process of assessing all requirements for interdependencies, it would make sense to identify the requirements with obvious interdependencies first, and represent them visually so that interdependency information could be inferred quickly [1].

Requirement Interdependency Types and Their Suitability to Multiple SPLs. Before we can discuss the most suitable representation for requirements between multiple SPLs, it is important to (1) describe the types of requirements interdependencies

which have been identified in the foremost literature, (2) identify the requirements interdependency types which are most critical to the scenario that one SPL has a requirement dependency to another SPL.

Tables 1 and 2 provide the most referenced sources for requirement interdependency categories [1, 13], and are in general alignment with each other on interdependency types.

Table 1. Interdependency categorization by Dahlstedt et al. [13]

Interdependency categorization	Interdependency type
Structural	Require
	Explain
	Similar to
	Conflict with
	Influences
Cost/Value	Increase/Decrease cost
	Increase/Decrease value

Table 2. Interdependency types by Carlshamre et al. [1]

Priority, lowest number takes precedence	Interdependency type, where R = requirement	Meaning, where R = requirement
1	R_1 AND R_2	R_1 requires R_2 to function, and R_2 requires R_1 to function.
2	R_1 REQUIRES R_2	R_1 requires R_2 to function, but not vice versa.
3	R_1 TEMPORAL R_2	Either R_1 has to be implemented before R_2 or vice versa.
4	R_1 CVALUE R_2	R_1 affects the value of R_2 for a customer. Value can be either positive or negative.
4	R_1 ICOST R_2	R_1 affects the cost of implementing R_2. Value can be either positive or negative.
5	R_1 OR R_2	Only one of $\{R_1, R_2\}$ needs to be implemented.

These requirement interdependency types provide the basis for a more complex analysis of how interdependencies, sometimes also called requirements relations, are considered in representation techniques for multiple SPLs. In Fig. 2, the difference between a requirement interdependency within one SPL and an interdependency between multiple SPLs is depicted. While most of the foremost literature and representation techniques consider the simple case of a requirement requiring another requirement within the same SPL, the case we want to consider is how the foremost representation techniques are suitable to the more complex case that there is a requirement interdependency between product lines.

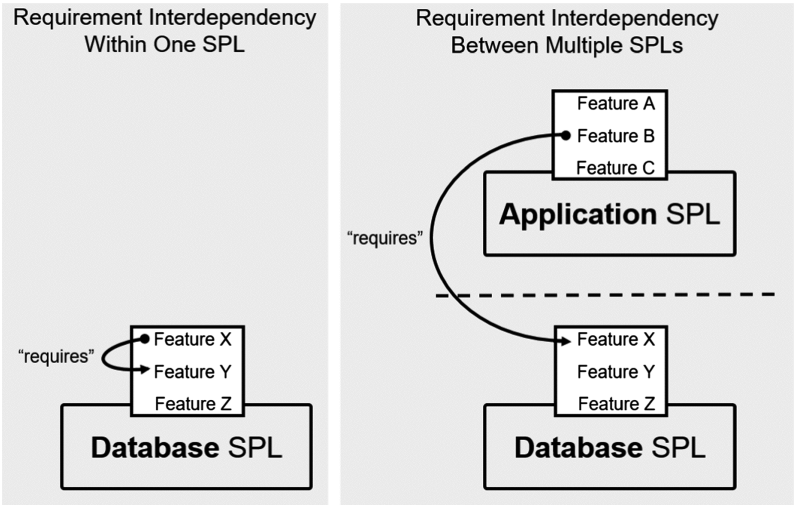

Fig. 2. Requirement interdependency within one SPL as opposed to between multiple SPLs

Because each SPL is, in its software productization, independent of the other one, the separate product lines have the potential to have different release timelines, market segments, share of company operational revenue, etc. [5]. Therefore, we can identify some existing requirement interdependency types to be more critical for requirements between SPLs.

We identify the "requires" type to be critical due to the definition of vertically tiered multiple SPLs. If a requirement between SPLs exist, it will always be a requirement from the top-most-level SPL of the SPL tier beneath it. Additionally, any requirement between SPLs most likely also has a "cost/value" interdependency type due to the nature of SPLs, which are designed to increase the value of software options at reduced overall development cost [1].

With this research, we aim to answer: which of the foremost techniques for representing the prioritization and interdependencies of requirements within one SPL is the most suitable for representing the more complex requirements interdependencies between multiple SPLs?

3 Representation Techniques for Requirements Prioritization and Requirements Interdependencies

We have identified the following representation techniques as part of a systematic literature review to be the most commonly referenced.

3.1 Directed Graph (Digraph) Representation

A directed graph (digraph) represents requirements as shapes connected by arrows. By differentiating requirements prioritization and interdependencies by color, line type, and size of the shapes and arrows, the digraph is able to represent complex relations between the requirements.

A practical example of a digraph requirements representation was made by Carlshamre et al. [1] when they took an industrial survey where requirements managers from five software organizations were asked to perform pairwise assessment on requirements, only considering priority. A pairwise assessment was performed using a spreadsheet designed by Carlshamre et al. and ensured that all requirements were compared with each of the other requirements.

The requirements managers were then asked to identify interdependencies between the requirements. They were also asked to give each interdependency a simple certainty rating (possibly-probably-positively). The interdependency types available in the identification are shown in Table 2 and include the type REQUIRES (dependency), CVALUE (customer value), and ICOST (increases cost).

In addition to identifying interdependencies with certainty ratings, Carlshamre et al. also define a hierarchy between interdependency types for the case that more than one relationship is identified between two requirements. The priority (or hierarchy) of the relationships is also shown in Table 2 with the REQUIRES interdependency type assigned priority 2, and both CVALUE and ICOST assigned priority 4. In the case that CVALUE and ICOST have a conflict, they have to be traded off against each other [1].

Using this data, Carlshamre et al. created digraphs of the requirements priorities and the requirements interdependencies for each software organization. By representing the requirements, their priorities, and their interdependencies by objects and arrows, it is possible to draw important conclusions just from a glance at the digraph. However, the authors of this paper observe that the graphical representation itself does not convey some of the more sophisticated data collected, like the certainty rating for the interdependencies.

3.2 Metamodel Ontology Representation

A metamodel ontology is an appropriate method for describing and visualizing requirements, their prioritization and their interdependencies because the model supports distilling the relationships between requirements to their base elements [7, 9, 15, 16].

Due to the flexibility of the metadata in a metamodel ontology, it is possible to rate the priority on a scale. The common structural interdependencies can be modeled, which include a “requires” interdependency type, however in the foremost literature, there is no example of modelling “cost/value” interdependencies in a metamodel ontology.

3.3 Software Requirements Catalog (SRC) Method

The software requirements catalog (SRC) is a method for collecting and considering software requirements for reuse instead of considering individual software features or

components for reuse. The creation of the SRC includes a classification phase, where the functionality of the requirement to fulfill the goal – the reason for the existence of the project – is described. The description includes a prioritization rating which reflects the suitability of the requirement to the project goal [17–19].

There is a qualitative high-medium-low rating scale for evaluating the priority of the requirement to fulfill the project goal. However, the definition of a "priority" in the SRC context is a variation on the definition of priority we have discussed previously. In the context of SRC, priority is a rating of the requirement in only the parameter of its suitability to fulfill the overall project purpose, and not of its overall criticality [17].

The requirement constraints and dependencies with other software projects are defined and refined in order to continuously update the requirements catalog [17]. The tracking of requirements interdependencies, also called traceability, seems to consider simple relationships, such as the "requires" structural interdependency between projects with common requirements. However, there is no mention of more complex requirements interdependencies between projects like "cost/value" interdependencies.

3.4 Fuzzy Graph Representation

Requirements interdependencies are considered fuzzy relations because the strengths of the dependencies can vary greatly [20]. Mougouei et al. model the influence of value-related requirements interdependencies using fuzzy graphs, which consider the uncertainty of the dependency relations.

Although there are also fuzzy representation techniques available for the prioritization of requirements, including the fuzzy AHP technique [11], none of those techniques take the interdependency of requirements into consideration [12, 13].

3.5 Cost-Value Diagram Representation

The cost-value approach to requirements management involves a pair-wise comparison on requirements in two dimensions: the requirement value and requirement cost [21]. Based on the results, the requirements are depicted on a graph with an axis for value and axis for cost and two delineations to fence off the high-value/low-cost requirements, mid-value/mid-cost requirements, and low-value/high-cost requirements. Karlsson et al. [21] developed a support tool to plot the scored requirements and were also able to take simple structural interdependencies into account, including the "requires" interdependency type [21].

4 Criteria and Comparison of Requirement Prioritization and Interdependency Representation Techniques Suitable for Multiple SPLs

In the following section, we present the criteria we used to assess the requirement prioritization and interdependency representation techniques suitable for multiple SPLs. We then compare the overall rating for the representation types discussed in Sect. 3.

4.1 Criteria for Rating

A simple (SMART) scoring technique has been applied to each of the dimensions for analysis [22].

Priority Consideration. Each representation type in Sect. 3. was evaluated against the sophistication of requirement priority consideration on a decimal scale from zero to one using the criteria in Table 3. A rating of zero corresponds to an absence of consideration. A rating of 0.5 corresponds to a simple scale priority rating, where requirements are given a standalone rating. A rating of 1.0 corresponds to a comparative prioritization where the requirements are compared to one another and then receive a relative priority rating.

Table 3. Criteria for priority consideration rating [22]

Rating	Criteria
0	Absence of priority consideration
0.5	Simple scale priority rating (ex. low, medium, high)
1.0	Relative prioritization using requirement comparison

Interdependency Consideration. Each representation type in Sect. 3 was evaluated against the sophistication of requirement interdependency consideration on a decimal scale from zero to one using the criteria in Table 4. A rating of zero corresponds to an absence of consideration. A rating of 0.5 corresponds to a simple interdependency tracking, where it is represented that requirements are linked. A rating of 0.75 corresponds to an interdependency consideration with a certainty rating. A rating of 1.0 corresponds to a complex interdependency consideration, where, for example, multiple types of interdependencies are represented.

Table 4. Criteria for interdependency consideration rating [22]

Rating	Criteria
0	Absence of interdependency consideration
0.5	Simple interdependency tracking (traceability)
0.75	Interdependency consideration with certainty rating
1.0	Complex interdependency consideration

Suitability for Multiple SPLs. Each representation type in Sect. 3 was evaluated against the suitability of the type for multiple SPLs on a decimal scale from zero to one using the criteria in Table 5. Specifically, the suitability criteria refer to the interdependency types “requires” and “cost/value”, which play a critical role in the assessment of requirements interdependencies between SPLs. A rating of zero corresponds to an absence of consideration for even the most basic structural interdependency type, “requires”. A rating of 0.5 corresponds to a simple consideration for either “requires” or “cost/value” interdependency types. A rating of 0.75 corresponds to either consideration for both “requires” and “cost/value” interdependency types or an in-depth consideration of either one. A rating of 1.0 corresponds to a representation type that allows for

requirement consideration for requirements from external SPLs, which bring potentially their own "cost/value" and "requires" requirements.

Table 5. Criteria for multiple SPL suitability rating [22]

Rating	Criteria
0	Absence of consideration for interdependency type "requires" and "cost/value"
0.5	Simple consideration for interdependency type "requires" or "cost/value"
0.75	Consideration for "requires" and "cost/value", or in-depth handling for one or the other
1.0	Additional parameters for "cost/value" to consider one or more dependent SPLs outside of the assessed SPL itself

4.2 Comparison of Representation Techniques

The following is a comparison of the previously discussed requirements prioritization and requirements interdependencies techniques using the criteria specified in the previous subsection.

The summary of the findings is that even though some of the techniques take into account all three criteria areas, prioritization, interdependencies, and multiple SPL requirements, none of them offer a complete solution to requirements prioritization with interdependencies between multiple SPLs.

The cumulative total ratings from Table 6 are visualized as a multi-dimensional graph in Fig. 3 to add a qualitative perspective to the quantitative ratings.

Table 6. Rated comparison of representation techniques

Technique	Priority consideration	Interdependency consideration	Suitability for multiple SPLs	Cumulative total rating
Digraph	1.0	0.75	0.75	2.5
Metamodel ontology	0.5	0.5	0.5	1.5
SRC	0.5	0.5	0.5	1.5
Fuzzy graph	0	1.0	0.75	1.75
Cost-value diagram	1.0	0.75	0.75	2.5

It was determined based on the given criteria that the SRC method and metamodel ontology both had a simple prioritization method and a simple interdependency tracking method. They both also had no mention of the "cost/value" requirements interdependency type, which is critical to a thorough representation of requirements between SPLs. The metamodel ontology representation rates slightly higher on the graph for multiple SPL suitability because it has greater potential for further extensibility through metadata. The SRC method limits flexibility to requirements interdependency handling between SPLs because the requirement scoring is from the perspective of the requirement to fulfill a specific project, and is therefore better suited to smaller software projects with fewer diverse stakeholders.

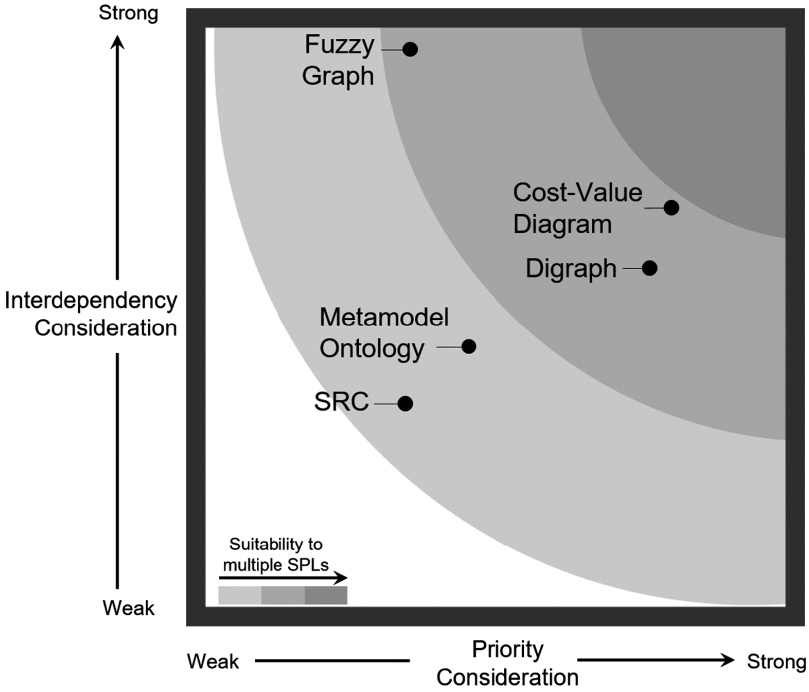

Fig. 3. Multi-dimensional graph of rated representation techniques

Slightly more suited for requirements between SPLs is the fuzzy graph representation, which gives very detailed information about requirements interdependencies and their uncertainties, but does not take prioritization into consideration at all.

Most suitable to requirements between SPLs are the directed graph (digraph) model and the cost-value diagram. Both are capable of distilling the required requirements interdependency types in a compact and straightforward way. Both could potentially be extended to represent relationships between SPLs. However, the cost-value diagram adds an additional perspective of potential investment areas (ex. high-value/low-cost requirements), which could valuable if extended to multiple SPL requirements support.

5 Conclusion

The assessment of the representation techniques in Sect. 4.2 Comparison of Representation Techniques shows that the cost-value diagram and the digraph representations are the closest to being suitable for multiple SPLs. More research is necessary into these representation techniques, specifically in how they could be extended to more accurately represent requirement interdependencies and the priorities thereof for multiple SPLs.

References

1. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.: An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of the Fifth IEEE International Symposium, Requirements Engineering (2001)
2. Mougouei, D., Powers, D.M., Moeini, A.: Dependency-aware software release planning. In: IEEE/ACM 39th International Conference Software Engineering Companion (ICSE-C), 2017 (2017)
3. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Inf. Softw. Technol. **43**(14), 883–890 (2001)
4. Carlshamre, P., Regnell, B.: Requirements lifecycle management and release planning in market-driven requirements engineering processes. In: 11th International Workshop Database and Expert Systems Applications, Proceedings (2000)
5. Schirmeier, H., Spinczyk, O.: Challenges in software product line composition. In: 42nd Hawaii International Conference, System Sciences, HICSS 2009 (2009)
6. Damiani, F., Schaefer, I., Winkelmann, T.: Delta-oriented multi software product lines. In: Proceedings of the 18th International Software Product Line Conference, vol. 1 (2014)
7. Soomro, S., Hafeez, A., Shaikh, A., Musavi, S.H.A.: Ontology based requirement interdependency representation and visualization. In: Shaikh, F.K., Chowdhry, B.S., Zeadally, S., Hussain, D.M.A., Memon, A.A., Uqaili, M.A. (eds.) IMTIC 2013. CCIS, vol. 414, pp. 259–270. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10987-9_24
8. Berander, P., Andrews, A.: Requirements prioritization. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 69–91. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28244-0_4
9. Sellier, D., Mannion, M., Mansell, J.X.: Managing requirements inter-dependency for software product line derivation. Requirements Eng. **13**(4), 299–313 (2008)
10. Rosenmüller, M., Siegmund, N., Kästner, C., ur Rahman, S.: Modeling dependent software product lines. In: Proceedings of the GPCE Workshop on Modularization, Composition and Generative Techniques for Product Line Engineering (McGPLE) (2008)
11. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N.: A systematic literature review of software requirements prioritization. Inf. Softw. Technol. **56**(6), 568–585 (2014)
12. Kahn, J., Rehman, I.: Comparison of requirement prioritization technique to find the best priorization technique. Modern Educ. Comput. Sci. **7**(11), 53–59 (2015)
13. Dahlstedt, A., Persson, A.: Requirements Interdependencies: State of the Art and Future Challenges. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 95–116. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-28244-0_5
14. Zhang, H., Li, J., Zhu, L., Jeffery, R., Liu, Y., Wang, Q., Li, M.: Investigating dependencies in software requirements for change propagation analysis. Inf. Softw. Technol. **56**(1), 40–53 (2014)
15. Goknil, A., Kurtev, I., van den Berg, K., Veldhuis, J.: Semantics of trace relations in requirements models for consistency checking and inferencing. Softw. Syst. Model. **10**(1), 31–54 (2011)
16. Saeki, M., Kaiya, H.: On Relationships among models, meta models, and ontologies. In: Proceedings of the Proceedings of the 6th OOPSLA Workshop on Domain-Specific Modeling (DSM 2006) (2006)
17. Pacheco, C., Garcia, I., Calvo-Manzano, J., Arcilla, M.: Reusing fuctional software requirements in small-sized software enterprises: a model oriented to the catalog of requirements. Requirements Eng. **22**(2), 275–287 (2017)

18. Robertson, S., Robertson, J.: Mastering the Requirements Process, 6th edn. Addison-Wesley, Munich (2013)
19. Wiegers, K.: First things first: prioritizing requirements. Software Dev. **7**(9), 48–53 (1999)
20. Mougouei, D., Powers, D.: Modeling and selection of interdependent software requirements using fuzzy graphs. Int. J. Fuzzy Syst. **19**(6), 1812–1828 (2017)
21. Karlsson, J., Olsson, S., Ryan, K.: Improved practical support for large-scale requirements prioritising. Requirements Eng. **2**(1), 51–60 (1997)
22. Valiris, G., Chytas, P., Glykas, M.: Making decisions using the balanced scorecard and the simple multi-attribute rating technique. Perform. Meas. Metrics **6**(3), 159–171 (2005)