A Conceptual Model for the Selection of Methods for Software Engineering Process Improvement

Tiago Amorim[0000−0003−0930−2307] and Andreas Vogelsang[0000−0003−1041−0815]

University of Cologne, Germany

Abstract. One way of improving the efficiency of system development is through the adoption of new methods. These, upon adoption, can provide the development team with the capabilities to address the current challenges. Once a team decides to follow this path, selecting which methods to adopt is a task ahead. The state-of-the-art provides a plethora of options. The criteria should be choosing methods yielding the highest net benefit towards the adoption goal. Many criteria influence the assessment of methods' value. Knowing which ones are relevant and how they are related is essential to complete this task with excellence. In this paper, we propose a conceptual model describing elements of the decision-making process when selecting software engineering methods to be adopted by development teams. We aim to make explicit much of the knowledge involved in this process, i.e., mechanisms and influencing factors, to foster proper value assessment of methods. For researchers, our work can serve as guidelines to describe methods, for the industry, the model allows the comparison of assessment methods and better-motivated business plans.

Keywords: Process improvement · Decision-making · Conceptual model

1 Introduction

As technology evolves, development teams face problems with the increasing complexity of software-intensive systems. New functionalities are replicated by market competitors, which soon become a commodity, pushing teams to deliver in less time to gain competitive advantage, and nevertheless with top quality. All this must be addressed at a global-dictated market-compatible cost that shrinks at every new development cycle.

One solution to the aforementioned problems is the replacement of development methods with more appropriate ones. The software engineering community has produced many methods to address most current challenges [9]. Thus, teams must select methods yielding the most significant returns toward alleviating their problems.

However, the constructs and their relations relevant to this type of analysis are not explicit. Usually, teams must infer method suitability in an ad-hoc manner (e.g., comparing the contextual characteristics of their teams and needs to eventual case studies provided). Missing proper appraisal can cloud the decision-making process. Consequences range from the inability to replicate the rationale to selecting inappropriate methods. The importance of alignment with the team's adoption goal is emphasized in a report released by the Project Management Institute (PMI) [1]. Projects and programs aligned with a team's strategy are completed successfully more often than misaligned projects (77% vs. 56%). At the same time, only 60% of strategic initiatives meet their original goals and business intent. The report states that most executives admit a disconnection between strategy formulation and implementation [19].

In previous work, we have studied change management in software engineering process adoption and improvement. First, we investigated the forces felt by stakeholders that play a role in the decision to undergo a process improvement endeavor [25]. Later we studied strategies and best practices that increase the success of the endeavor [5]. And finally, we devised an approach to prioritize candidates according to the adoption goal and the development team's context [4]. In this work, we expand the operationalization of strategic goals with a conceptual model bearing relations and properties (e.g., associated sacrifices, environment's context) not addressed by our previous studies while keeping semantic similarity.

The contribution of this paper is a conceptual model for selecting methods to be adopted by software development teams. Conceptual models are schematic descriptions of a phenomenon. They explicitly represent constructs, activities, properties, and relations within a specific problem domain in a reasonably complete manner [23]. We find two issues specially relevant: proper association of process improvement goals with the candidate methods and their contextual suitability towards achieving the defined goal. Additionally, related sacrifices must be considered to find methods bringing the best net benefit (i.e., benefits minus sacrifices). The model can improve the interoperability of methods that function at different granularity levels (i.e., vertically related), or the same level (i.e., horizontally related) through semantic interpretation of the languages' constructs [21]. Moreover, it can serve as a reference model to assess modeling approaches regarding the selection of methods based on value.

The remainder of the current paper is structured as follows: Section 2 describes the relevant theories backing up the development of the conceptual model. Section 3 presents the approach. Section 4 provides a small example to illustrate the model elements. In Section 5, we discuss the model and its implication for industry and research, and Section 6 brings the concluding remarks.

2 Background

2.1 Relevant theories

The perception of value results from a conceptualization of the object being assessed in terms of a desired end, i.e., whether the object's qualities allows the agent assessing its value to fulfill an end [20]. Thus, different agents will assess different values to the same object according to their goals, i.e., value is a relational and emergent characteristic. Additionally, the definition of the goal and the benefit harvested from the qualities is context-dependent [6]. Value is composed not only of benefits but the relation of benefits and the associated sacrifices involved in acquiring and using the object, i.e., the net benefit. This value theory underpins the model proposed in this paper.

Process improvement initiatives aim at selecting and implementing new methods. These initiatives are guided by goals that are strongly related to context [3], either because there is a need to change the status quo (e.g., become market leader, improve code quality) or to keep it as it is (e.g., maintain the market share). Thus, properly assessing the value of candidate methods allows for a higher goal achievement rate.

2.2 Related work

The Business Motivation Model (BMM) [17] aims to model why an enterprise chooses a particular approach for its business activities. The model achieves this through two elements, namely ”ends” and ”means.” The former is a placeholder for the goal or objective an enterprise wishes to achieve. The latter describes ways of attaining those ends (e.g., tactics, strategies), and directives from the organization or the business. Since the model focuses on the enterprise, it has elements to define the organization's Vision and Mission. Our model focuses on the method adoption goals of a smaller organization unit, namely the development team. The granularity level that this model is represented is the same as our approach.

Papatheocharous et al. created a taxonomy to document architectural decisions, the GRADE taxonomy [18]. Five dimensions for architecture decision-making were defined: goal, roles, assets, decision methods and criteria, and environment. The authors claim that the knowledge they provide is important for replicating successful architectural decisions or avoiding inefficient ones. Since their contribution is limited to a taxonomy, the relations between elements are out of scope. Additionally, they only consider the environment context.

Andersson et al. [6] propose an ontology of value ascription for enterprise modeling focusing on economic resources. Sales et al. [22] extended the previous approach to value proposition (i.e., defines what a company delivers to its customers). In a further work [21], they analyzed the risk and its relation to use value. We extended these works by adapting them to the selection of methods for software engineering process improvement.

On describing attributes to support method selection, Ågerfalk and Wistrand [2] propose including the rationality dimension in describing methods to store the author's values and assumptions about the problem domain upon method creation. It is divided into two kinds of sub-rationale: method prescriptions anchored in goals, referred to as goal rationale, and goals anchored in values, namely value rationale. The author must define a method's value and connection to goals in this approach. A similar reasoning is used in our model, i.e., a method helps to achieve goals. However, the relevant criteria (e.g., contextual characteristics) are only implicitly considered, which requires the method creator to deliver information ad-hoc.

In Gonzalez-Perez et al. [12], the authors propose a goal-based approach to select so-called method fragments. Their approach proposes to model the adoption goal based on prioritizing specific attributes. The authors propose ten attributes grouped into three areas: Product, Project, and Organization. The method fragments are evaluated towards enhancing or deteriorating each attribute using a five-level scale: strongly enhances, enhances, neutral, deteriorates, and strongly deteriorates. Finally, Goal analysis is used to select the set of method fragments that most enhance the prioritized attributes. This approach includes benefits and sacrifices through a scale ranging from negative to positive influence. They use pre-defined attributes to link goals and the method fragments. The relation between elements, although sometimes implicit, is similar to the one described in our model.

Many goal-oriented requirements engineering (GORE) methods have been proposed [14] (e.g., KAOS, i*, Troppos). These focus on goals, sub-goal refinement, soft goals, and requirements generation. Some also provide reasoning techniques to decide between alternatives for goals' refinement. In these approaches, sacrifices, benefits, and context might appear in the refinement. However, these elements are not explicit, thus being considered ad-hoc. Our model promotes these elements to first-class citizens, giving them more importance.

Current approaches from the literature recognize the need to link high-level goals and context with the implementation. However, these relations and relevant elements are sometimes implicit or incomplete, thus, requiring an ad-hoc effort. Once addressed, these shortcomings can foster more successful projects, which we would like to achieve with our model.

3 Proposal

This section describes the conceptual model for selecting methods for software engineering process improvement (CMSM). The selection criteria are based on the net benefits of adopting these methods. The benefits help the team achieve an envisioned future state described by the adoption goal and are composed by the tuple {Adoption goal, Candidate method qualities, Context}. The Candidate method qualities are intrinsic to the method and generate benefits. The Context influences the adoption goal and describes the characteristics of the team, environment (e.g., new regulations need to be followed), project, and product. The Context also influences how the method qualities can contribute to the adoption goal. The team should also consider possible sacrifices for the new method (e.g., running costs). The net benefit of a method is the benefit minus sacrifices, which is the result of the method value assessment. The outcome of the value assessment can be used to compare methods and decide on the ones that better achieve the goal.

We use the Unified Modeling Language (UML) [8] notation to describe the model, which is depicted in Figure 1. Three labels are used in the model's associations, namely *q dep*, *+q dep*, and *-q dep*. The first characterizes relationships with a qualitative influence on other elements, which can be positive or negative. The second label represents positive qualitative influence. The third label represents negative influence relationships. For instance, `Sacrifices` has a *-q dep* relation to the `Assessment relationship` while `Benefits` has a *+q dep* relationship. `Context` has a *q dep* relationship to *Sacrifices*, meaning it can have either positive or negative influence.

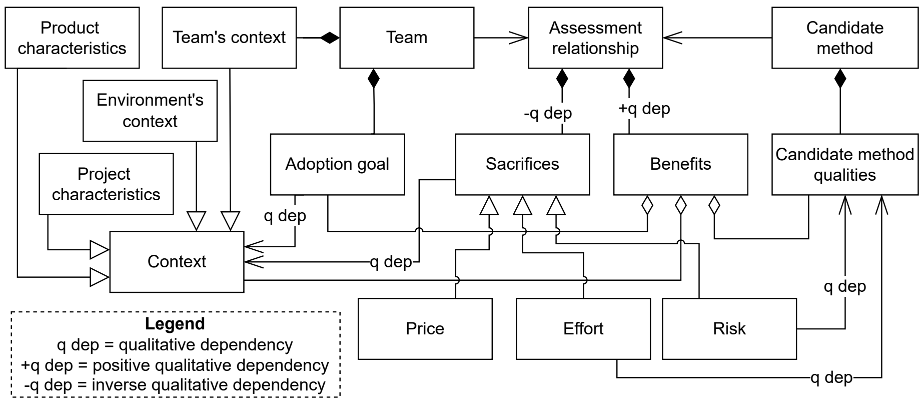

Fig. 1. Conceptual model for the selection of methods (CMSM).

In the following, we describe the model constructs. They are written in ordinary text using `Courier` font. We provide a simple example to illustrate the concepts in Section 4. The goal of the example is to instantiate each of the constructs. Since this is a conceptual model, we do not provide a decision on the example.

Adoption goal. describes why a software development team (i.e., `Team`) is undergoing process change based on a future state it wants to achieve. This element is influenced by the `Context` through a qualitative dependency relation. For instance, the `Team's context` describes capabilities and characteristics that need to be improved or are lacking in the `Team`, or the `Environment's context` drives the `Team` to reach out to new regulations or develop capabilities to better compete with other market players.

Team. is the software development team undergoing process change and performing the value assessment in the `Candidate methods`, which is related through the `Assessment relationship` element. The `Adoption goal` and the `Team's Context` have a composite relation to the team.

Candidate method. is the method appraised by the `Team` that might be implemented in the process change endeavor. The term method includes tasks, activities, processes, methodologies, method fragments, and related. The `Candidate` `method` has a connection to two other elements; it shares a composition relationship with the `Candidate method qualities` and an association with the `Assessment relationship`.

Candidate method qualities. are intrinsic properties of the `Candidate method` that upon adoption help the `Team` to achieve the `Adoption goal`, thus, directly influencing the perception of `Benefits`. Qualities may refer to the kind of products that the method can help build, the project type used to tackle such activities, and the team's characteristics where these projects may take place. They influence `Effort` (i.e., adoption and execution efforts), and can increase or decrease the `Risk` of not achieving the `Adoption goal`.

Context. is the current state of affairs or the interrelated conditions in which something exists or occurs [16]. It influences the method assessment because they change the `Benefits` perception, how the `Adoption goal` is defined, and the `Sacrifices` (e.g., a technically weak team increases effort and risk). The Context is constantly changing and can have a multitude of factors [10], whose relevance is dependent on the `Candidate method qualities` and `Adoption goal`. For instance, team size, a `Team's Context` factor, is relevant when adopting agile methodologies. In our model, Context is divided into four sub-types described in the following:

- **Environment's context** represents everything outside the boundaries of the `Team` and the `Candidate method` that influence the perception of value. For instance, methods compliant with regulations, preferences of stakeholders, requirements from interacting systems, or market competitors who are constantly raising the bar on what is necessary to fulfill the consumers' needs.
- **Team's context** is the team's intrinsic characteristics that influence the perception of value, including the methods the team employs before the method value assessment. For instance, larger teams might value processes and methods that develop a minimal amount of documents, whether smaller teams would praise agile more. Teams having many skillful members might perceive more advanced methods as more beneficial. If the employee turnover is high, documentation-oriented methods might be preferable.
- **Project characteristics** are elements srelated to the project's characteristics the `Team` is currently developing, which can influence the assessment of `Benefits` or be the `Adoption goal`'s reason. For instance, the availability of stakeholders belongs to this element and influences the value of agile methods. Instances of this element are time or budget constraints.
- **Product characteristics** describe the software developed by the team relevant to the method assessment. For instance, a product handling sensitive information would require methods to test its vulnerability. This element encompasses the code and artifacts, such as requirements documents and user manuals.

Benefits. are the perceived positive aspects of adopting the `Candidate method`. They stem from the `Candidate method qualities` within a `Context` that allows a `Team` to achieve an `Adoption goal`. It composes the `Assessment relationship` together with the `Sacrifices`.

Sacrifices. are the resources the `Team` must concede in order to adopt and exercise the `Candidate method`. Together with the `Benefits`, it composes the `Assessment relationship`. They can be influenced by the `Context`. Sacrifices can be of three different types (i.e., inheritance relationship):

- **Price.** is the monetary expenditure required to have the `Candidate method` implemented in a `Team`. Whatever is needed to have the method running and can be acquired through financial means is covered by this element. A few examples are tool acquisition, investments in training, setting up infrastructure, and licenses in general.
- **Effort.** is the time required to learn, put into use, or acquire the `Candidate method`. It has a qualitative dependency on the `Candidate method qualities` and can be of the following types [15] (not represented in the model diagram):
  - **Acquisition effort:** the sacrifice (e.g., time) needed to search for `Candidate methods`, evaluate and implement the method. For instance, some methods might require lengthy training sessions, thus increasing the associated efforts.
  - **Operations and maintenance efforts:** the maintenance and disposal costs, the time to learn how to use the `Candidate method`, the wait for it to perform, and monitoring. For instance, static analysis and code inspection aim to improve code quality, but the latter is much more effort-intensive.
  - **Complementary effort:** The time and cost needed to find and acquire complementary products or services associated with the `Candidate methods`.
- **Risk.** is related to the probability of the sacrifices to be more strenuous than predicted or even not fulfilling the goal at all with the `Candidate method`, which in both conditions incur on increased `Sacrifices` (e.g., money wasted). It has a qualitative dependency on the `Candidate method qualities` and can be classified in the following dimensions [15] (not represented in the model diagram):
  - **Safety:** physical risks related to the application of the `Candidate method`.
  - **Financial:** the risk that the financial expenditure is higher than usual. A possible kind is fluctuations in the exchange rate.
  - **Selection:** the risk of not choosing the best alternative for fulfilling the `Adoption goal`. This construct is relevant when informed decisions are not possible.
  - **Delay:** the risk that the `Candidate method` will take more time than expected to be implemented or not perform on time, thus, incurring opportunity costs. This element is very relevant for time-critical projects.

- **Functional:** is the risk related to the possibility that the `Candidate method` will not perform as predicted, now or in the future.

Assessment relationship. represents the significance attached to a `Candidate method` by the `Team`. It is influenced by two opposed elements: the `Sacrifices` and the `Benefits`. These two connect to the Assessment relationship through a composite relation, but the former has inverse qualitative dependency while the latter has a positive qualitative dependency.

4 Exemplification

This section provides a small example to illustrate the model elements. For this, let us consider a software development team that needs help with the quality of its code. Thus, they have defined the following `Adoption goal`: *Improve the code quality.* This team is considering between two `Candidate methods`, static program analysis, and code review.

Static program analysis consists of programs or algorithms designed to extract facts from another program's source code, which can be used to further understand, evaluate, and modify the associated code base [24]. The `Candidate method qualities` from this method are:

- Facts from different categories can be extracted from the source code.
- Less knowledge of developers is required to use the tools and find errors.
- Less time is required to perform de analysis.
- False positives can consume time to be investigated.
- Limited reach.

Code review (CRW) is a software quality assurance practice widely employed in open source and commercial software projects to detect defects, transfer knowledge and encourage adherence to coding standards [11]. The `Candidate method qualities` from this method are:

- Decreases the number of post-release defects.
- Improves the software quality.
- Promotes knowledge transfer.
- Promotes adherence to the project coding standards.
- Requires more experienced developers.

Market competitors of the development team are going for shorter release cycles, and there is a need to keep up, which is a `Environment's context` fact. `Team's context` characteristics relevant for assessing value are: personnel experience, since code review requires more experienced developers, team turnover rate, since the benefits of having knowledge transfer are lost once the employee leaves the team. `Project characteristics` that are relevant is the available time for project development. CRW is effort intensive, and if there is little availability, this can be a problem. Considering the `Sacrifices`, the Acquisition `Effort` for Static Program Analysis is slightly higher than the CRW. The Operation `Effort` is higher for CRW, which can sometimes be 15% of development time [11]. Two elements of `Risk` type can be elicited from the `Candidate` `method qualities`. The CRW has the risk of an over-optimistic evaluation of the time required, incurring a Delay risk. Static Program Analysis might miss the type of errors the team injects, incurring Functional risk.

5 Discussion

The CMSM conceptual description level allows it to be used for the interoperability of methods through semantic interpretation of the languages' constructs. This capability is possible because language integration is a semantic interoperability problem [21], and this can be applied to methods at different levels of granularity (i.e., vertically related, e.g., [12, 14]), or at the same level (i.e., horizontally related, e.g., [2, 17, 18, 6, 22, 21]). The CMSM is developed on level 2 of the Technical Readiness Level [13], which stands for *Technology concept formulated*.

The scope for method selection of the CMSM is the team. Other models consider bigger scope (e.g., organization, enterprise [17]). We understand an organization can have many teams, each requiring different methods. The CMSM considers a single adoption goal since the value assessment is related to how well the method can help achieve the goal. Thus, different goals provide different assessment outcomes. The modeling of goals (i.e., refinement into sub-goals and soft-goals) is not considered by this model. Additionally, the goal influences the relevance of more fine-grained characteristics.

Correctly categorizing the `Context` element in its sub-types allows the team to perceive what they can change (i.e., `Team's context`). The organization dictates some team characteristics, which are considered `Environment's context` since method adoption will not change these characteristics.

The proposed model describes the influencing factors for the method selection based on value. The model is useful for connecting goal models, process models, and value models. Some theories suggest the principle of separating strategy from implementation. Choosing the best method to achieve the strategy is not separate from the strategy itself. Additionally, the lack of traceability with associated rationale increases the risk of implementing the wrong solutions and failing to achieve the strategy. Thus the focus is on the interrelation of the elements that support the task of defining how the goal is to be achieved.

A limitation of our model is the need to use it together with other modeling approaches since it is described at the conceptual level. However, the state-of-the-art provides many modeling approaches that support the elements described in the CMSM. Some elements are more popular than others. `Goal`, `Context`, and `Risk` elements have many modeling approaches [14, 10, 21], while `Price` and `Effort` models are less popular.

Impact for the industry. The proposed model can help decision-makers to assess the coverage of their process change roadmaps, i.e., whether some detail needs to be considered, thus, enabling more robust business plans.

Impact for academia. Researchers can use our model to verify whether important aspects regarding adoption guidance based on value and context are considered when proposing new software engineering methods [7]. Additionally, the model stems further philosophical development of the elements and their relations. By providing direction for researchers on what to consider when suggesting new methods, the CMSM can impact method adoption research and stimulate discussion on the completeness of frameworks.

6 Conclusion

Selecting appropriate methods for software engineering process improvement is a complex and important task. By understanding the mechanics of the assessment process, better decisions can be made for effective method adoption. This paper has presented a conceptual model that can help decision-makers in this process by linking adoption goals and contextual characteristics with the benefits of method implementation. The discussion in this paper has highlighted the advantages of bridging the gap between process and decision modeling. Future work can refine the model by assigning attributes and redefining relationships. Another possibility is developing an operationalization method to perform the assessment as described. Finally, the model can be used to integrate different modeling approaches.

Acknowledgments

This work has been supported by the German Ministry of Research and Education (BMBF) within project SpesML (Sysml workbench für die SPES methodik) under grant 01IS20092C.

References

1. Project Management Institute, https://www.pmi.org/, Accessed: 07.05.2023
2. Ågerfalk, P.J., Wistrand, K.: Systems development method rationale: A conceptual framework for analysis. In: Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS'03) (2003)
3. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering **15**(4), 439–458 (Jul 2010)
4. Amorim, T., Vogelsang, A., Dias Canedo, E.: Decision support for process maturity improvement in model-based systems engineering. In: Proceedings of the 16th International Conference on Software and System Processes (ICSSP'22) (2022)
5. Amorim, T., Vogelsang, A., Pudlitz, F., Gersing, P., Philipps, J.: Strategies and best practices for model-based systems engineering adoption in embedded systems industry. In: Proceedings of the 41st ACM/IEEE International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP'19) (2019)
6. Andersson, B., Guarino, N., Johannesson, P., Livieri, B.: Towards an ontology of value ascription. In: Proceedings of the 9th International Conference on Formal Ontology in Information Systems (FOIS'16). vol. 283 (2016)

7. Arora, C., Sabetzadeh, M., Briand, L.C.: An empirical study on the potential usefulness of domain models for completeness checking of requirements. Empirical Software Engineering **24**(4), 2509–2539 (Apr 2019)
8. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series). Addison-Wesley Professional (2005)
9. Bourque, P., Fairley, R.E. (eds.): SWEBOK: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, Los Alamitos, CA, version 3.0 edn. (2014), http://www.swebok.org/, Accessed: 07.05.2023
10. Clarke, P., O'Connor, R.: The situational factors that affect the software development process: Towards a comprehensive reference framework. Information and Software Technology **54** (2012)
11. Ebert, F., Castor, F., Novielli, N., Serebrenik, A.: Confusion in code reviews: Reasons, impacts, and coping strategies. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 49–60 (2019)
12. Gonzalez-Perez, C., Giorgini, P., Henderson-Sellers, B.: Method construction by goal analysis. In: Information Systems Development, pp. 79–91. Springer US, Boston, MA (2009)
13. Héder, M.: From nasa to eu: the evolution of the trl scale in public sector innovation. The Innovation Journal **22**, 1 (2017)
14. Horkoff, J., Aydemir, F.B., Cardoso, E., Li, T., Maté, A., Paja, E., Salnitri, M., Piras, L., Mylopoulos, J., Giorgini, P.: Goal-oriented requirements engineering: an extended systematic mapping study. Requirements Engineering **24**(2) (2017)
15. Kambil, A., Ginsberg, A., Bloch, M.: Re-inventing value propositions. NYU Stern School of Business Research Paper Series
16. Merriam-Webster: Context. In: Merriam-Webster.com dictionary (2021), https://www.merriam-webster.com/dictionary/context, Accessed: 07.05.2023
17. Pankowska, M.: Business motivation model for information system architecture development support. Journal of Software and Systems Development (2021)
18. Papatheocharous, E., Wnuk, K., Petersen, K., Sentilles, S., Cicchetti, A., Gorschek, T., Shah, S.M.A.: The grade taxonomy for supporting decision-making of asset selection in software-intensive system development. Information and Software Technology **100**, 1–17 (2018)
19. PMI: PMI's Pulse of the Profession (2017)
20. Zúñiga y Postigo, G.: An ontology of economic objects. Mpra paper, University Library of Munich, Germany (1999)
21. Sales, T.P., Baião, F.A., Guizzardi, G., Almeida, J.P.A., Guarino, N., Mylopoulos, J.: The common ontology of value and risk. In: Proceedings of the 37th International Conference on Conceptual Modeling (ER'18) (2018)
22. Sales, T.P., Guarino, N., Guizzardi, G., Mylopoulos, J.: An ontological analysis of value propositions. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC). pp. 184–193 (2017)
23. Thalheim, B.: Towards a theory of conceptual modelling. In: Lecture Notes in Computer Science, pp. 45–54. Springer Berlin Heidelberg (2009)
24. Thomson, P.: Static analysis. Commun. ACM **65**(1), 50–54 (dec 2021)
25. Vogelsang, A., Amorim, T., Pudlitz, F., Gersing, P., Philipps, J.: Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry (2017)