Towards a Goal-oriented Method for Software Solutions Prioritization

Prisca Petelo, Abderrahmane Leshob[a], Benzarti Imen[b] and Hafedh Mili[c]
University of Quebec at Montreal, Montreal, Canada

Keywords: Goal Modeling, Archimate, Goal-oriented Requirement Language, Software Prioritization, Model-driven Design, Enterprise Architecture, Solutions Architecture.

Abstract: Architecture practitioners, such as enterprise architects, solutions architects, and application architects are often faced with the problem of selecting the best software solutions that implement the requirements and satisfy the business objectives. Examples of these solutions are: web services, software components, and full software applications. To identify the best solution, architects often have to prioritize the candidate solutions according to a set of criteria, such as their quality attributes, their contributions to satisfy the (business) objectives, and their cost of implementation. This work aims to design a method that helps architects to identify the optimal solution that achieves the requirements and efficiently satisfies the business objectives. The proposed method is composed of three steps. First, it builds a goal model that links each candidate solution to: i) the functional requirements to be implemented and ii) the desired objectives to be satisfied. The goal model uses the Archimate language. It connects the requirements, goals and solutions together according to the Goal-oriented Requirement Language (GRL) rules. Second, the method computes automatically satisfaction scores that measure the effectiveness of each solution. Third, the method prioritizes the solutions according to their satisfaction scores. This work presents the principles underlying the proposed method and discusses its possible application in the practice.

1 INTRODUCTION

Architecture practitioners are often faced with the problem of selecting the software solution that best meets their organization needs among a set of candidate solutions. Business Architects, working with business analysts are responsible for elaborating business cases that propose different candidate solutions and then do a comparative study in order to select the best one that achieves stakeholders requirements and objectives. Solutions and application architects must select the best components, services and frameworks that implement the software functional requirements and that efficiently achieve the quality attributes. Enterprise architects must identify the best solutions to automate business processes. For example, to automate a business process, an enterprise architect may choose either a Service-Oriented Architecture (SOA)-based solution, a BPM (Business Process Management) solution or an ERP (enterprise resource planning).

To select the best solution(s), architects need to prioritize the candidate solutions according to a set of criteria, such as their quality attributes, their contributions to satisfy the (business) objectives, and their cost of implementation. Although prioritization of the solutions plays a crucial role in enterprise architecture (EA), business and solutions architecture, few methods were proposed in the literature. Existing prioritization approaches, including (Badidi, 2013; Czekster et al., 2019; Kwong et al., 2010; Maximilien and Singh, 2004; Min, 1992; Wei et al., 2005; Ziaee et al., 2006) suffer from a number of limitations including complexity and usability.

In this paper, we propose a new easy-to-use method that helps architecture practitioners to prioritize software solutions. The prioritization process is based on the requirements that the solutions must implement and the business objectives they need to satisfy. The proposed method is goal-oriented. It combines two languages: the Archimate language, a language for modeling EA and the Goal-oriented Requirement Language (GRL) that allows the assessment of the relative effectiveness of design alternatives, such as candidate solutions. First, the

[a] https://orcid.org/0000-0002-4066-3111
[b] https://orcid.org/0000-0003-0658-9605
[c] https://orcid.org/0000-0002-1220-9042

Petelo, P., Leshob, A., Imen, B. and Mili, H.
Towards a Goal-oriented Method for Software Solutions Prioritization.
DOI: 10.5220/0010891500003119
In *Proceedings of the 10th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2022)*, pages 287-293
ISBN: 978-989-758-550-0; ISSN: 2184-4348
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

method builds an Archimate-based goal model that links candidate solutions (e.g., payment web service) to the functional requirements (e.g., pay by credit card) to be implemented and the desired objectives (e.g., Secure payment) to be satisfied. The goal model connects Archimate requirements, goals and solutions according to the GRL rules and constraints. Second, it computes automatically satisfaction scores that measure the effectiveness of the solutions. Third, the method prioritizes the solutions according to their satisfaction score obtained in the previous step. The higher the score is, the better is the solution.

This research has been conducted following the design science methodology (Peffers et al., 2007). The design science research approach aims to answer questions related to relevant issues through the creation of innovative artifacts (Peffers et al., 2007). The artifact proposed in this research is a prioritization method.

The remainder of the paper is organized as follows. Section 2 describes our goal-oriented method for software solutions prioritization using a document management system. It also presents the basic concepts of the Archimate and the GRL languages. Section 3 surveys related work. Section 4 draws our conclusions and outlines directions for future research.

2 A GOAL-ORIENTED METHOD FOR SOFTWARE SOLUTIONS PRIORITIZATION

Our goal is to design an easy-to-use method that helps architecture practitioners such as enterprise architects and solutions architects to prioritize candidates solutions according to their ability to implement the requirements and their contributions to satisfy the software and stakeholders goals.

To provide an easy-to-use method that targets a large community of architecture practitioners, we chose the Open Group Archimate language (The Open Group, 2019) to design the models. We also needed a goal language that i) links the requirements, solutions and goals, and ii) provides a mechanism to evaluate the impact of the choice of solutions on the business objectives. Among goal modeling languages KAOS (Keep All Objects Satisfied) (van Lamsweerde, 2004), GRL (ITU-T, 2012), and *i** (Yu, 1997), we found that GRL is suitable to design our method as it is a standard, lightweight, and easy to integrate with the Archimate language.

2.1 ArchiMate Language

ArchiMate is an EA modeling language that has been adopted as a standard by the Open Group (Jonkers et al., 2011). Archimate allows the creation of models to build an EA (Jonkers et al., 2017). It is made up of two dimensions: the layer and aspect dimensions (Gaydamaka, 2019; Jonkers et al., 2017). The layers represent the successive levels of abstraction at which a business is modeled and the aspects represent the different concerns to be modeled (Gaydamaka, 2019). Archimate is composed of five layers, namely Strategy, Business, Application, Technology, Implementation and Migration and four aspects, namely Passive structure, Behavior, Active Structure, and Motivation (The Open Group, 2019).

The Archimate aspects have been inspired by the natural language. Active structure elements (e.g., actor, role, application component) are the subjects that can perform behavior. Behavior elements represent the dynamic aspects (e.g., business process, business interaction, event) of the organization. Passive structure elements (e.g., product, business object) can be accessed by behavior elements (The Open Group, 2019). Motivation elements (e.g., goal, requirement) represent the reason behind the architecture or the behavior (The Open Group, 2019).

Strategy layer elements are used to model the strategic direction of the organization. These elements are used to model how the organization wants to create value, the resources needed, and how it plans to use these resources. Business layer elements (e.g., business service) allow to model the operational aspects of the organization (The Open Group, 2019). Business layer models are technology-independent. Application layer elements (e.g., application service, application component) model the application architecture (The Open Group, 2019). Technology layer elements are used to model the technology architecture of the organization (The Open Group, 2019). Physical layer elements allow modeling the physical world (The Open Group, 2019). These elements are included as an extension to the technology layer. The Implementation and Migration layers support the implementation of the architectures and their migration (The Open Group, 2019).

2.2 GRL Language

GRL is a goal-oriented modeling language (ITU-T, 2012). It allows to model the requirements to be achieved by the solutions, the goals to be satisfied, the tasks and their relationships. Figure 2 presents

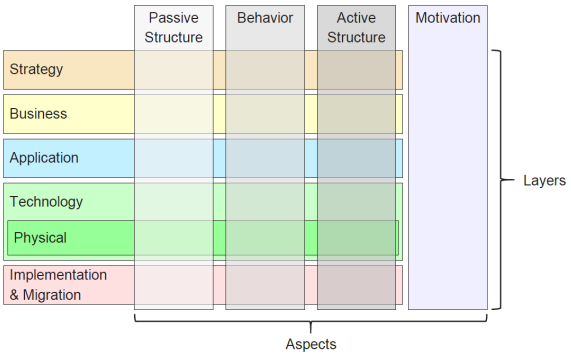

Figure 1: Archimate language layers and aspects (The Open Group, 2019).

the subset of the GRL intentional elements. A goals are quantifiable elements. They usually refer to functional requirements. Softgoals refer to qualitative aspects (Amyot et al., 2010). Soft-goals are usually related to non-functional requirements. A task is a solution which achieves goals or satisfies softgoals (Amyot et al., 2010).

Figure 3 shows the contribution and means-end links that are used to connect goals, softgoals, and solutions in a GRL model. It also illustrates the contribution types (Section b). Means-End links describe how goals (e.g., functional requirements) are achieved (Amyot et al., 2010). Contribution links specify desired impacts of one element on another element (e.g., softgoal) (Amyot et al., 2010). As shown in Figure 3, the contribution link can have a qualitative contribution type, or a quantitative contribution (e.g., integer values) (Amyot et al., 2010). As shown in the Section b of Figure 3, contribution links can be labeled using texts or icons.

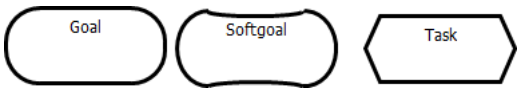

Figure 2: Basic GRL intentional elements (adapted from (Amyot et al., 2010)).

2.3 The Approach

Figure 4 illustrates the process of the proposed method. The first step builds a goal model that combines constructs from the Archimate and GRL languages. This model connects the candidate solutions, such as web services, to i) the objectives they satisfy and ii) the requirements they implement. The second step computes a satisfaction score for each solution using a GRL-based evaluation algorithm. This score measures the extent to which each solution satisfies the objectives. The last step ranks the solutions according to their satisfaction score.

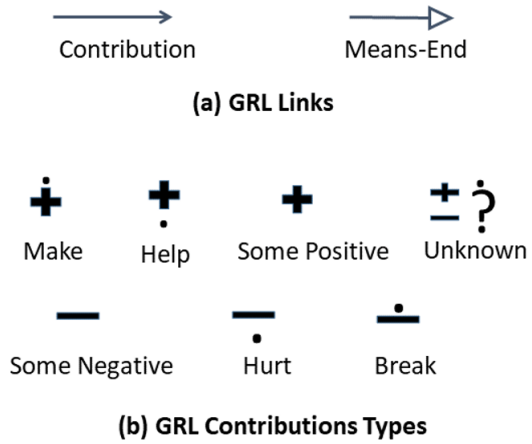

Figure 3: GRL links and contributions types (adapted from (Amyot et al., 2010)).

To explain the proposed method, let's consider a medical clinic that wants to implement an efficient document management system (DMS) to improve the management of its patients medical files. Currently, all client files are paper-based. The main goal (higher business objective) to satisfy by implementing an automated document management solution is *efficient file management*. To satisfy this goal efficiently, the architect has identified three candidates solutions: Integrated Document Management (IDM) service, Amazon Workdocs service, and Miscrosoft SharePoint Document Management Service.

2.4 Build the Archimate Goal Model

The goal of this subprocess step is to create an Archimate-based goal model that links the solutions (e.g., services) to the goals/objectives they satisfy and to the requirements they implement. This step is composed of two activities as shown in Figure 5. The first activity builds a high-level goal model that links the goals and sub-goals. The second activity connects the solutions to i) the requirements they implement and ii) the goal model obtained in the previous activity.

2.4.1 Create the High-level Goal Model

This step consists of creating the high-level goal model that contains the Archimate goals (GRL soft-goals) to satisfy. A goal is an Archimate motivation aspect. In this model, sub-goals are linked to high-level goals using Archimate *influence* link (GRL contribution link). Lets take the example of the medical clinic. The highest goal to satisfy by implementing an automated document management

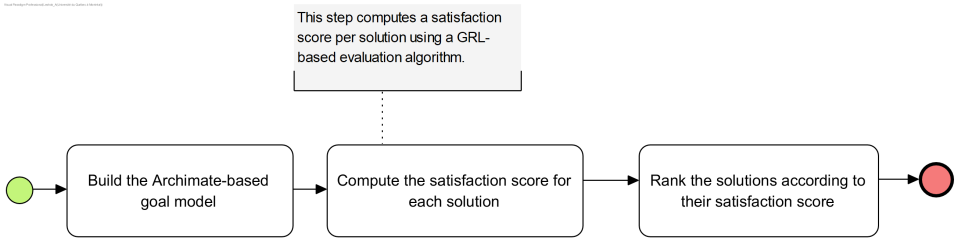

Figure 4: Overall process of the proposed method.

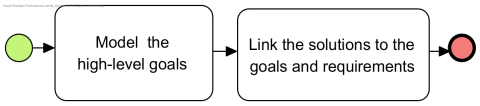

Figure 5: The process of modeling the goal model.

system is *efficient file management*. This high-level goal has four sub-goals, namely *Secure solution*, *Usable solution*, *Auditable solution*, and *Unified solution*. To create the high-level goal model, the architect proceeds as follows:

1. Specify the *importance* of high-level goals. The importance attribute of a goal indicates how important it is for the system or the organization. The proposed method uses importance values from 0 to 100. However, the modeler can use non-integer values from any range. The importance value of a goal is shown in brackets. In our example, the architect assigned an *importance value* of 100 to the highest goal (i.e., *efficient file management*) (see Figure 6).
2. Link sub-goals to the highest goals using the Archimate *influence* link (GRL contribution link) and specify the *contribution value* for each influence link. The contribution value indicates the desired impacts of a sub-goal on a higher-level goal. As for importance values, we use contribution values from 0 to 100. However, the architect can use non-integer values from any range.
3. Calculate the importance values for each sub-goal using their contribution links' values. Importance values are propagated down from high-level goals to sub-goals using the following formula (see (Shamsaei et al., 2011)):

$$sGoal.iValue = \frac{pGoal.iValue \times sGoal.cValue}{100}$$

where $sGoal.iValue$ and $sGoal.cValue$ represent the importance value and the evaluation value of the sub-goal respectively and $pGoal.iValue$ is the importance value of the parent goal. For example, the importance value of the sub-goal *Usable solution* is $(\frac{100 \times 80}{100}) = 80$.

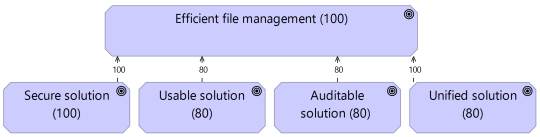

Figure 6: DMS high-level goal model.

The resulting DMS high-level goal model is shown in the Figure 6.

2.4.2 Link the Solutions to the Goals and Requirements

A complete goal model contains goals, solutions and requirements. The previous steps built an Archimate-based goal model that contains only the goals (high-level goals and sub-goals) to be satisfied. This step, links the candidate solutions to the requirements and the goal model obtained in the previous step. The solutions are Archimate concepts that satisfy goals. Solutions may refer to Archimate concepts such as business services (from business layer), application services or application components (from application layer). The requirements are Archimate concepts from the motivation layer. According to the GRL language, the requirements elements can not be linked directly to the goals elements. Thus, the proposed method links the requirements to the goals through the solutions. To add the solutions and the requirements to high-level goal model, the architect proceeds as follows:

1. Link the solutions to the goals using the Archimate *influence* link (GRL contribution link). Each link must be quantified using a contribution value which indicates the impact of the solution on the achievement of the goal.
2. Link the solutions to the requirements using the Archimate *realization* link (GRL means-end link).

For the sake of simplicity and to better explain the proposed method, we will retain two solutions for the DMS: Integrated Document Management (IDM) service and Amazon Workdocs service) and two sub-objective: *Secure solution* and *Usable solution*.

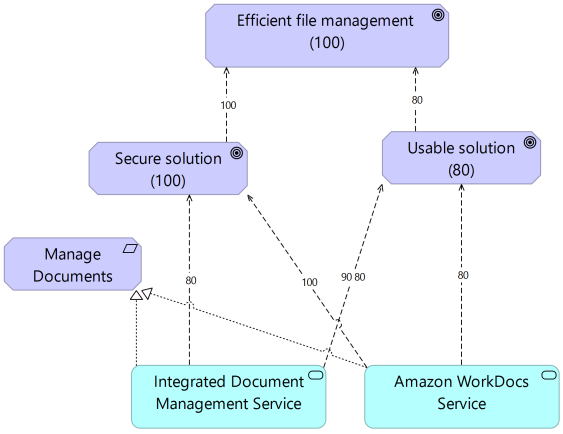

Figure 7: DMS complete goal model.

Figure 7 illustrates the complete goal model of the DMS.

2.5 Compute the Satisfaction Score

GRL provides a quantitative algorithm to evaluate goal models, allowing us to compute the satisfaction score of each candidate solution. Our approach uses an adaptation of the evaluation algorithm described in (Amyot et al., 2010). The first step initializes the evaluation values for each solution. By default, our method gives all solutions an initial neutral value of 100, meaning that all solutions have the same initial satisfaction score. The algorithm then propagates the evaluation values using a bottom-up approach to obtain *evaluation values* for each goal. The evaluation values are propagated through the influence links. For example, the evaluation value of the sub-goal *Usable solution* is computed by i) multiplying the evaluation value of the solution *Amazon Workdocs service* (i.e., 100) by the value of the influence link that connects the solution (i.e., *Amazon Workdocs service*) to the sub-goal (i.e., *Usable solution*) and ii) then dividing the result by 100 (i.e., $(\frac{100\times80}{100})$).

Therefore, the evaluation value of a goal (g) reached by *one* source element (solution or sub-goal) denoted *src* through an influence link is computed as follows.

$$g.eValue = \frac{src.eValue \times src.cValue}{100}$$

where $src.eValue$ refers to the evaluation value of the source element *src* and $src.cValue$ refers the contribution value of the influence link that connects *src* to the goal g.

When the goal is reached by n contribution links, the evaluation value is computed as the average of the calculated evaluation values of each influence links.

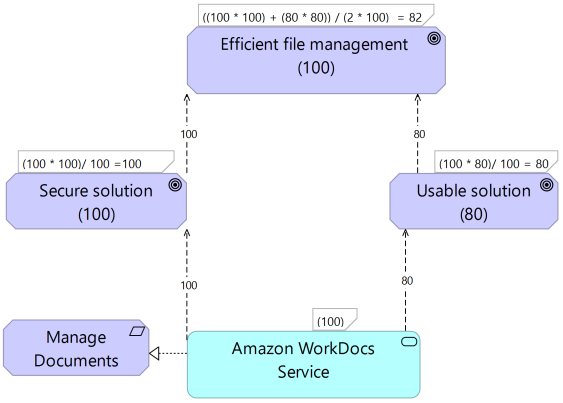

Figure 8: Evaluation of the DMS goal model.

Thus, the evaluation value of the high-level goal *Efficient file management* is computed as follows.

$$\frac{((100\times100)+(80\times80))}{2\times100} = 82$$

After propagating the evaluation values, the algorithm computes the satisfaction score for each solution using importance values and the evaluation values of the goals. The satisfaction score of a solution S is computed as follows.

$$S.score = \frac{\sum_{i=1}^{N} g_i.iValue \times g_i.eValue_i}{\sum_{i=1}^{N} g_i.iValue}$$

where $g.iValue$ and $g.eValue$ are the importance value and the evaluation value of the goal g respectively. Thus, the satisfaction score of the solution *Amazon Workdocs service* is computed as follows.

$$\frac{((100\times100)+(80\times80)+(100\times82))}{100+80+100} = 87.86$$

3 RELATED WORK

A few research works so far proposed approaches to prioritize software solutions. Kwong *et al.*, (Kwong et al., 2010) designed an interesting approach that optimizes software components selection for component-based software system (CBSS) development. The approach uses an genetic algorithm that maximizes the functional performance of the CBSS and the cohesion and minimizes the coupling of software modules. In (Min, 1992), (Wei et al., 2005), authors proposed approaches to select a suitable software using the analytic hierarchy process (AHP), a structured technique which can be effectively used with both qualitative and quantitative factors to analyze complex decisions.

In (Badidi, 2013), Badidi designed a framework for the selection of Software-as-a-Service (SAAS) that relies on service level agreements (SLA) between SaaS providers and consumers. The framework uses an algorithm that ranks potential SaaS providers by matching their offerings against the requirements of the SLA.

In (Leshob et al., 2020), authors proposed an approach to prioritize software requirements. They used their method to rank the software solutions based on the requirements they implement. These approaches are based on a method that uses a one-to-one mapping between candidate solutions and the requirements. In the business analysis domain, the BABOK (Business Analysis Body of Knowledge) guide (IIBA, 2015) provided a technique based on the value delivered by a solution to assess its performance and prioritize it.

In (Radulescu, 2016), Radulescu provided guidelines that allow business analysts to select and prioritize security solutions. The guidelines were summarized in a solution selection and prioritization matrix based on security cost efficiency of the solution and the average compliance gap subsequent to its implementation. The author proposed to use these guidelines to design a generic methodology or process that can be uniformly applicable to any organizational context, regardless of how information security is addressed.

Maximilien and Singh (Maximilien and Singh, 2004) designed a framework for dynamic web services selection. The framework uses an agent that selects the web services based on a quality of service (QOS) ontology.

Baker *et al.,* (Baker et al., 2006) proposed a method for ranking and selection of candidate software components based on a series of features. The method uses a greedy and simulated annealing algorithms to determine a subset of components that maximizes the total sum of weights (customer desirability and expected revenue) while minimizing the total cost (the cost of acquisition and development time) of the selected components from a catalog of components. In a similar work, Haghpanah *et al.,* (Haghpanah et al., 2008) designed an algorithms to automate component selection for component-based software. The software component selection is performed through a process of identifying a minimal set of components that satisfy a set of requirements and objectives while minimizing cost.

These approaches are interesting. However, they all suffer from a number of limitations including their usability and difficulty of implementation. The contribution of this work is twofold. First, it proposes a novel end-to-end method that takes into account the contribution values of the solution to the business objectives (softgoals). Second, from a usability point of view, the method is easy to use by non-technical users, such as business architects or analysts.

4 CONCLUSION AND FUTURE WORKS

In this work, we designed a new method to prioritize software solutions such as, services, components, or any reusable software artefact in order to select the solution that best meets business and stakeholder needs. The prioritization process of the candidate solutions is based on i) the requirements they implement and ii) the goals (business objectives) they satisfy. The proposed method is goal-oriented. It uses two languages: the Archimate language, a language for modeling EA and GRL, a goal language that allows the assessment of the effectiveness of design alternatives. First, the method builds an Archimate-based goal model that links candidate solutions to the functional requirements to be implemented and the desired goals to be satisfied. The goal model connects requirements, goals and solutions according to GRL rules. Second, it computes automatically satisfaction scores that measure the effectiveness of each candidate solution. In the final step, the method ranks the solutions according to their satisfaction score obtained in the previous step.

This paper establishes guidelines and directions to design an automatic end-to-end method that helps architects and even non-technical users, such as business analysts to identify effective solutions. We plan to conduct experiments to validate the proposed approach. We also plan to extend the method in order to support other prioritization factors, such as the cost and the complexity of implementation.

ACKNOWLEDGMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E. (2010). Evaluating goal models within the goal-oriented requirement

language. *International Journal of Intelligent Systems*, 25(8):841–877.

Badidi, E. (2013). A Framework for SOFTWARE-AS-A-SERVICE Selection and Provisioning. *International journal of Computer Networks and Communications*.

Baker, P., Harman, M., Steinhöfel, K., and Skaliotis, A. (2006). Search based approaches to component selection and prioritization for the next release problem. In *IEEE International Conference on Software Maintenance, ICSM*.

Czekster, R. M., Webber, T., Jandrey, A. H., and Marcon, C. A. M. (2019). Selection of enterprise resource planning software using analytic hierarchy process. *Enterprise Information Systems*.

Gaydamaka, K. (2019). Archimate-based approach to requirements engineering. *International Journal of Mathematical, Engineering and Management Sciences*.

Haghpanah, N., Moaven, S., Habibi, J., Kargar, M., and Yeganeh, S. H. (2008). Approximation Algorithms for Software Component Selection Problem.

IIBA (2015). *A guide to the Business Analysis Body of Knowledge, Version 3*. Lightning Source inc.

ITU-T (2012). ITU-T,User Requirements Notation (URN)–Language definition.

Jonkers, H., Groenewegen, L., Bonsangue, M., van Buuren, R., Quartel, D. A., Lankhorst, M. M., and Aldea, A. (2017). A language for enterprise modelling. In *Enterprise Engineering Series*.

Jonkers, H., Proper, E., Lankhorst, M. M., Quartel, D. A., and Iacob, M. E. (2011). ArchiMate® for integrated modelling throughout the architecture development and implementation cycle. In *Proceedings - 13th IEEE International Conference on Commerce and Enterprise Computing, CEC 2011*.

Kwong, C. K., Mu, L. F., Tang, J. F., and Luo, X. G. (2010). Optimization of software components selection for component-based software system development. *Computers and Industrial Engineering*.

Leshob, A., Hadaya, P., and Renard, L. (2020). Software Requirements Prioritization with the Goal-Oriented Requirement Language. In *Lecture Notes on Data Engineering and Communications Technologies*.

Maximilien, E. M. and Singh, M. P. (2004). A framework and ontology for dynamic web, services selection. *IEEE Internet Computing*.

Min, H. (1992). Selection of Software: The Analytic Hierarchy Process. *International Journal of Physical Distribution and Logistics Management*.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*.

Radulescu, M. C. (2016). Considerations on the selection and prioritization of information security solutions. *Audit Financiar*.

Shamsaei, A., Pourshahid, A., and Amyot, D. (2011). Business process compliance tracking using key performance indicators. In *Lecture Notes in Business Information Processing*.

The Open Group (2019). The Open Group Standard: ArchiMate 3.1 Specification.

van Lamsweerde, A. (2004). Goal-oriented requirements enginering: a roundtrip from research to practice. In *12th IEEE Int Requirements Engineering Conference*, pages 4–7.

Wei, C. C., Chien, C. F., and Wang, M. J. J. (2005). An AHP-based approach to ERP system selection. *International Journal of Production Economics*.

Yu, E. S. K. E. (1997). Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the IEEE International Conference on Requirements Engineering*, pages 226–235, Annapolis, USA. IEEE.

Ziaee, M., Fathian, M., and Sadjadi, S. J. (2006). A modular approach to ERP system selection: A case study. *Information Management and Computer Security*.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS