Towards software reuse through an enterprise architecture-based software capability profile

Abdelhadi Belfadel, Emna Amdouni, Jannik Laval, Chantal Cherifi, Néjib Moalla

▶ **To cite this version:**

Abdelhadi Belfadel, Emna Amdouni, Jannik Laval, Chantal Cherifi, Néjib Moalla. Towards software reuse through an enterprise architecture-based software capability profile. Enterprise Information Systems, 2022, 16 (1), pp.29-70. 10.1080/17517575.2020.1843076 . hal-02997243

HAL Id: hal-02997243
https://hal.science/hal-02997243v1

Submitted on 31 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Towards Software Reuse Through an Enterprise Architecture-based Software Capability Profile

Abdelhadi Belfadel[a], Emna Amdouni[a], Jannik Laval[a], Chantal Bonner Cherifi[a] and Nejib Moalla[a]

[a] University Lumiere Lyon 2, DISP Laboratory, Lyon, France

ARTICLE HISTORY
Compiled May 31, 2023

ABSTRACT
Most of today's software development projects depend on the usage of existing solutions to save time and development cost. We target in this research work the design of a software capability profile that provides a broader view of an organization's internal and external software, along with an exploitation model in line with requirements engineering and enterprise architecture to fill the gap between the goals of the stakeholders and what can be delivered as a practical solution. For this purpose, we define a Framework that offers a qualification that helps to gather the initial requirements that guided the development of existing software. This qualification is based on a proposed Enterprise Architecture Capability Profile and its associated ontology covering business, operational and technical aspects for service-oriented software. Furthermore, an exploitation methodology is proposed and based on the alignment of requirements engineering with software architecting actions that evolve together, to investigate the highest compatibility of the desired functionalities. Our contribution aims to improve the reuse of existing services, by upgrading these technical components to the level of end-user's requirements for accelerating future business application development. An implementation and a case study are proposed to demonstrate the effectiveness of this approach.

KEYWORDS
Software Reuse; Software Capability Profile; Enterprise Architecture; Requirements Specification; Service-Oriented Architecture; Ontology

1. Introduction

In today's world, business organizations need an information system to track all their business activities. These activities related to business processes can be assisted by software tools and enterprise applications whose objective is the automation of the exchanges in the business environment. Business processes are a set of activities that define how specific business tasks are performed, and need to be adapted as a response to evolution in internal and external environments that become more and more complex. In this scenario, variations in stakeholders' needs or a new collaboration development

CONTACT A. B. Author. Email: Abdelhadi.Belfadel@univ-lyon2.fr
E. A. Author. Email: Emna.Amdouni@univ-lyon2.fr
J. L. Author. Email: Jannik.Laval@univ-lyon2.fr
C. C. Author. Email: Chantal.BonnerCherifi@univ-lyon2.fr
N. M. Author. Email: Nejib.Moalla@univ-lyon2.fr

are examples of change inductors [1; 2].

The necessity for companies to modernize or design a new process by reusing features from existing software and integrate their applications within and across organizational boundaries can be of great value. By taking advantage of previous development and considering internal solutions of companies enriched with external ones, this contributes to facilitate the development of complex systems with controlled costs while maintaining delivery schedules.

SOA is one such approach that has received much attention as an architecture for integrating platform, protocol, and legacy systems [3]. As an architectural approach, it decomposes business applications into individual processes and functions as services. These latter are considered as an abstract business concept, which represents the functionalities of business [4]. Each of such discrete components of software functionality can be recomposed to build alternative applications. It can be exposed also to other functions and systems as services that enable different applications to reuse common parts [3].

In this perspective of reusability to lower the development cost and quickly provide values to enterprises, the center Blackduck for open source research and innovation releases in 2017 an open source 360-degree survey [1]. This study was sent to practitioners, consumers and contributors of open source solutions. Over 800 people responded to the survey, spanning industries from financial services through manufacturing and retail to technology companies. The results show that approximately 60% of respondents increased usage of open source software, following strong usage growth in 2016. A lack of vendor lock-in and the ability to build internal applications by customizing, or simply reusing functionalities offered by software were considered as three of the key reasons respondents choose to use open source. However, companies are facing problems to implement new (ICT) solutions, finding challenges both from IT vendors' expensive integrated digital solutions and from a myriad of disparate highly focused technological open source solutions for specific functions.

Many practitioners and research work as [5], [6], [7] and [8] have studied the reuse potential of freely available softwares. Still, no standardized process has been proposed, as practitioners still use ad-hoc methods for identifying the most suitable artifacts to reuse [9]. Indeed, the complexity of the external software ecosystem leads to difficulties in searching, evaluating and retrieving technical components to reuse. Factors such as lack of documentation, uncertainty on the quality of the technical components, and the difficulty in searching and retrieving these components are the most important factors preventing reuse [10]. Moreover, companies' eagerness for a quick result based on a reuse approach is a big challenge, especially if the targeted system must be integrated within a company's software ecosystem with a controlled cost.

We aim in this context to design a software capability container that provides a broader view of an organization's internal and external software. This to respond to stakeholder requirements and efficiently reuse the qualified solutions, by investigating the highest functional and non-functional compatibility of the desired functionalities and related constraints. Therefore, we aim in this research work to achieve three main goals: (i) Define a Software Capability Profile that describes software components capabilities from several perspectives, including organizational, functional, technical and technological viewpoints along with its associated quality and constraints; (ii) Enable the reuse of solutions by considering a process which includes requirements engineering capabilities for formalizing stakeholder requirements and constraints; (iii) Define

[1] https://www.blackducksoftware.com/open-source-360deg-survey

an exploitation model of the software capability profiles in line with a requirement engineering and enterprise architecture (EA), to fill the gap between the goals of the stakeholders and what can be delivered as a practical solution.

However, to attend the indicated objectives we faced several problems. The first problem is how to identify the architecture artifacts and initial requirements used to guide the development of existing solutions? The second problem is how to align requirements and architecture artifacts in an engineering cycle for the consolidation and refinement of requirements, to facilitate the discovery and reuse of existing solutions? The third problem is how to formalize, organize and capitalize artifacts produced during the evaluation and the exploitation phases to guide an organization maximizing the reuse? Therefore, the main research problem that this research addresses is how to manage the complexity of the exploitation of an organization's internal or external software capability profiles, based on an alignment of a requirement engineering and architecting process to select the best candidate components to act as building blocks in a new system.

To respond to the research problem, this paper is organized as follows: Section 2 focuses on the related work. We focus afterward on the principal building blocks of the proposed solution in section 3 and present the proposed EACP Framework with its related meta-model and exploitation plan. Section 4 presents the derived ontology from the proposed meta-model. Section 5 presents a concrete use case and an implementation of the Framework. Section 6 discusses our work and finally a conclusion is drawn in Section 7.

2. Related Work

2.1. *Enterprise information systems engineering and enterprise architecture*

In the highly dynamic industrial and economic environments, enterprises' information systems became an important assets for organizations to obtain integrated support for managerial decision making [11]. However, those information systems require constant reorganization to meet changing market requirements and technological evolution [12].

Organizations are complex systems and are strongly influenced by the environment in which they operate. The design of an enterprise information system that underpins the operations of an enterprise is guided by the development of strategy, management of programs and enterprise architecture (EA) [13]. The latter supports practitioners to handle enterprise information system engineering and changes [14], and align business strategies with IT to catch unexploited opportunities that has placed enterprises in competitive disadvantage in emerging market due to inadequate alignment [15].

EA is sometimes in confusion with traditional architectural approaches such as information system architecture or software architecture. While traditional architectural approaches tend to focus on technological concerns, the EA approach is more business-related than IT-related and therefore promotes an effective alignment of information technology and business.

EA can be employed in several domains. For example in smart cities context, the need to develop complex information systems has been amplified. Therefore, EA helps to address the problems of complexity of information systems deployed in cities [16]. Authors in [17] stated that the growing complexity and lack of integration across systems results in organizational barriers to stakeholder collaboration in smart cities. In

this context, authors identified that EA is used to provide better support for ICT architecture design, evaluation, diagnostics and monitoring for decision-making support and optimization of smart services. In healthcare context, Authors in [18] proposed a case study to build an enterprise information system for a health center [19]. The steps applied in this study are based on the Architecture Development Method of TOGAF Framework [20], that includes the vision, business, information systems, and technology architecture. In the education context, authors in [18] proposed to design a university information system architecture to achieve organizational objectives and provide services to stakeholders and produce a blueprint of information system architecture. This study was designed by applying the TOGAF ADM method, and translated into enterprise architecture modeling. In the context of managing both business architecture and IT architecture, authors in [21] proposed an approach to evaluate IT investment of a business. The authors used ArchiMate framework to assess IT projects value and to model the relationship between business processes, products, services and software applications.

2.2. *Capability Profile for Software Reuse*

Describe, share or exchange capabilities of software through capability profile is an interesting and valuable way to find and select adequate software components which fit the requirement [22]. Based on the systematic analysis of relevant research works published between 2012 and 2020 regarding component or service description with consideration of our needs, Table 1 classifies the related service description works according to the following criteria:

- C1) Technical and Functional description: description of the service interfaces, the business functions and related inputs and outputs.
- C2) Semantic Annotations: additional information that identifies or defines a concept in a semantic model. The annotations can be used during Web service discovery and/or composition.
- C3) QOS description: description or measurement of the overall performance of a service
- C4) Technology platform & environment description: Technology platforms and their decomposition, determining the combinations of technology needed to achieve a particular technology stack. Regarding the environments and locations, it is a grouping of the targeted technology into computing environments (e.g., development, production) in a specific location (cloud, on-premise or hybrid).
- C5) Organizational impact: description of the stakeholders, business problems, goals and objectives that guided the development of the application exposing the services.
- C6) Implementation and standards specification: subset of the architecture requirements specification that provides a quantitative view of the solution, stating measurable criteria that should be met during the implementation of the architecture. The typical content of these architectural requirements contains the implementation guidelines, implementation specifications, implementation standards, and Interoperability requirements.
- C7) Constraints specification: specification of different constraints that guided the development of the services as technical, business, implementation, standard, interoperability or technology constraints.
- C8) Reasoning capabilities: to infer logical consequences from a set of asserted

facts or axioms.

Out of Table 1, the related work based on OWL-S and WSMO have a wide coverage for describing services regarding the needed criteria to achieve a wider view coverage for service capabilities. However, very few works have considered the constraints specification, technology platform and execution environment regarding the software exposing the services, nor the organizational impact that the service has on the organization where it is used.

To maximize and enable the reuse of the exposed features, we should go beyond current research and service descriptions as depicted in Table 1. A representation of a high-level view of IT systems and enterprises business processes consuming the services is needed to provide a wider view qualification. Taking into consideration the business, operational and technical views of software and their related services.

Table 1. Software and Service Description Works

Software/Service Capability Description	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Barros et al., 2012 [23], Ghazouani et al., 2017 [24]	+		+					
Matsuda et al., 2012 [22]	+			+			+	
Mezni et al., 2012 [25], Keppeler et al., 2014 [26]	+		+					
Zheng et al., 2012 [27], Zhang et al., 2013 [28], Wei et al., 2014 [29], Roman et al., 2015 [30], Chhun et al., 2016 [31], Ghazouani et al., 2017 [32]	+	+	+					+
Pedrinaci et al., 2014 [33]	+	+	+					
Narock et al., 2014 [34]	+	+	+			+		+
Khanfir et al., 2015 [35]	+	+	+		+			+
De et al., 2017 [36], Khodadadi et al., 2015 [37]	+							
Benfenatki et al., 2017 [38]	+	+	+	+				
Haniewicz et al., 2012 [39], Verborgh et al., 2013 [40], Bravo et al., 2014 [41], Kapitsaki et al., 2014 [42], Alarcon et al., 2015 [43], Beydoun et al., 2019 [44]	+	+						

2.3. *Requirements Engineering and Feature Selection*

Some research projects addressed the issue of features selection based on consumers' requirements. Xu et al. [45] propose a paradigm for software service engineering to reuse services for developing new applications more rapidly with the aim of satisfying individualized customer requirements. The proposed approach uses service context as a mediating facility to match a service requirement with a service solution. The

requirements are defined by the targeted business functionality, service performance, and value. However, no details about the service pattern description or repository, the requirement template nor an implementation of the approach are proposed. Chen et al. [46] propose a method that allows users of services to express their requirements. The authors propose a meta-model for elements required in service consumption such as process, goal or role. The proposed method helps to discover errors and conflicts during requirement refinement. Zachos et al. [47] propose a service selection algorithm based on textual requirement expressed by the service consumer. The service selection is based on a discovery algorithm, that uses XQuery and WordNet and focuses on the disambiguation and completeness of the requirements and retrieving discovered services from UDDI registry. There are additional ontology-based research work such as Verlaine et al. [48], where the authors took the CORE Ontology (for Core Ontology for REquirements) [49] for requirement elicitation, and established a relationship with the concepts of WSMO [50].

Architectures are profitably used both in requirements analysis and design for new applications or business processes [51]. When EA work is implemented, it results several benefits such as reuse of different type of artifacts that provide a representation of the organization and guidelines for its development [52]. From an EA and requirements engineering perspective, some latest research work such as [53] and [14] proposed to identify some EA models by using different reverse engineering or mining techniques to discover artefacts to reuse. This allows, for instance, to facilitate the identification of stakeholders' concerns or build other missing models.

2.4. *Knowledge Management and Service Repositories for Service Reuse*

Research on repositories for an effective and useful management and discovery of services for service-oriented paradigm has recently earned significant impulse. In what follows, we list the works of the literature published between 2012 and 2020 regarding service discovery with consideration of our needs, Table 2 classifies the related service registry and discovery works according to the following criteria:

- C1) Organizational level: exploitation based on the identification of the stakeholders, business problems, goals and objectives of the targeted project.
- C2) Functional level: exploitation based on service interfaces, the business functions and related inputs and outputs.
- C3) Technical level: exploitation based on the identification of relevant technical requirement, interoperability requirement and technology constraints
- C4) Technology level: exploitation based on the identification of the platforms and infrastructure decomposition, determining the combinations of technology needed to achieve a particular technology stack.
- C5) Non-functional properties (QOS, Security...)
- C6) Exploitation based on a Requirements Engineering Process (it concerns all the mentioned levels)

Out of Table 2, we notice that several research works considered the functional level and QoS to manage service repository and matchmaking, but few of them considered the other levels such as the organizational level, the technical or technology level. We notice also that few research works considered the exploitation of the service registry in a software engineering cycle using a requirement engineering process to manage the user requirements for service discovery and matchmaking. Architecting actions

helps to manage the complexity of software by providing an abstraction of the system. Requirement engineering process drives the architecture actions, whereas decisions made in the architectural phase can affect the achievement of initial requirements and thus change them. We should go through these two fundamental activities namely requirement engineering and software architecting during the engineering process. These activities should evolve together to offer support to the developer or architect for formalizing the requirements and architectural artifacts to enable software and service discovery and reuse. There is, however, no structured solution (as depicted in Table 2) on how to perform the co-development of requirements and architecture actions to select the suitable software or services to reuse for the development of new business software.

Table 2. Service repository and discovery for reuse

Service Repository and Discovery	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**
Yu et al., 2012 [54], Haniewicz et al., 2012 [39], Hog et al., 2013 [55], Yoo et al., 2013 [56], Keppeler et al., 2014 [26], Narock et al., 2014 [34], Moradyan et al., 2015 [57],		+			+	
Seba et al., 2012 [58], Paliwal et al., 2012 [59], Xue et al., 2015 [60]		+				
Rathore et al., 2013 [61]					+	(+)
Li et al., 2013 [62]		+				(+)
Becha et al., 2014 [63]					+	(+)
Rodriguez-Garcia et al., 2014 [64], Kapitsaki et al., 2014 [42], Chiplunkar et al., 2014 [65]		+				
Matsuda et al., 2014 [66]		+				(+)
Alarcon et al., 2015 [43]		+	+			
Elshater et al., 2015 [67]		+		+		
Boissel et al., 2015 [68]		+				(+)
Khanfir et al., 2015 [35]	+	+			+	
Chhun et al., 2016 [31]		+			+	(+)
Purohit et al., 2016 [69]		+				
Zeshan et al., 2017 [70]		+		+	+	
Mu et al., 2018 [71]	+	+				(+)
Elgazzar et al., 2014 [72], Smiari et al., 2020 [7]		+		+	+	

2.5. *Scientific relevance and discussion*

From the state-of-the-art on service-oriented software reuse, we analyzed that currently ad-hoc methods are still used to identify the most suitable service-oriented software or artifacts to reuse, and a methodology or standardized process enabling this is still missing. Moreover, the description, the capability or qualification of these software are lacking wider view qualification taking into consideration the business, operational and technical views of the software and their related services. In addition, no solution has been provided to fit the requirement engineering along with the impact of architecture on requirements when dealing with the identification of the most suit-

able components and avoid the misevaluation during the selection phase. Therefore, enhance the capability description of the software and its related services in different levels of service description, along with its exploitation based on requirement engineering and architecting actions is a big challenge. This analysis highlighted the need for an Enterprise Architecture-based methodology for describing and classifying different artifacts produced during previous engineering lifecycles to be available as building blocks for reuse in future projects.

From the above analysis, we propose the following research directions: (i) Improve software and related service capability profile to bring value-in-use of the qualified feature for an organization that is interested into reuse; (ii) Shape a mechanism to identify the most suitable software with specific features or functionalities helping to overcome the use of ad-hoc methods; (iii) Improve the reuse of service-oriented solutions by considering a process which includes a requirement engineering and architecting actions for formalizing the requirements; (iv) Finally, ontologies enable to address the major challenges regarding software capability description by providing a standard vocabulary to prevent semantical problems and establish a common foundation for sharing enterprise architectural knowledge and perform software discovery, requirement consolidation, and reuse. Therefore, creating an ontology-based repository enables us to organize existing architecture and solution artifacts by considering internal solutions of companies enriched with external ones, along with artifacts that result from new business process development to be available as building blocks and facilitate the development of complex systems.

3. Enterprise Architecture Capability Profile Framework

Our goal is to maximize the reuse of an organization's internal and external software by improving the capability description, discovery, and sustainability of existing solutions, and by investigating the highest functional and non-functional compatibility of the desired functionalities and related constraints. For this purpose, we propose the EACP Framework which is based on The Open Group Architecture Framework (TOGAF) and its related Architecture Development Method (ADM). This proposed Framework offers higher-level of functional representations and covers the qualification and discovery of service-oriented software from the organizational, business, operational and technical aspects.

3.1. *Overview of the EACP Framework*

The EACP Framework is presented in Figure 1 and is composed by three main processes. The first process annotated with number (1) is called "Qualification Process". This latter takes as inputs a software and its related service endpoints. A wider view qualification is realized on the selected service-oriented software and creates a link between a feature (from a business point of view) and corresponding physical component (in this case the service endpoint). This process is composed of three activities as illustrated in Figure 1. The first activity called "Quality Check" where a quality assessment on the software and its related service endpoints is realized. Then for each service, an enterprise architecture-based qualification starts. Then, the activity called "Common Software Unit Requirement" guides the first qualification attempt based on an ISO 16100 Capability Profile specifically enhanced for service-oriented solutions. This latter specifies the functional properties, common software unit requirement, the

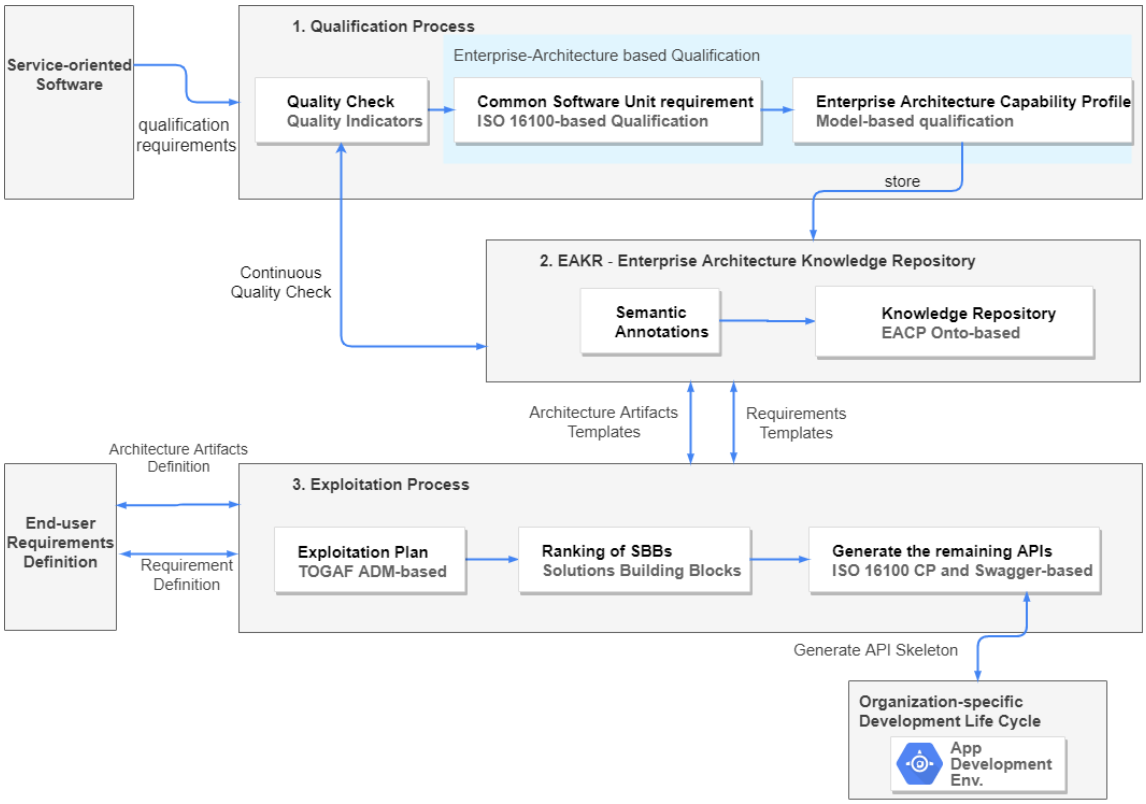

Figure 1. EACP Framework

activity name of the qualified functionality, needed parameters, and any functional dependency. This ISO-based profile assesses the qualification process before going to the next EA model-based qualification task.

Indeed, in the next task "Enterprise Architecture Capability Profile", a TOGAF-based model qualification is realized, and the output of this last task is characterized as EACP. The result of this qualification process offers a wider view of the software and its related exposed services. For each service, it results the organizational impact, details about the execution environment, non-functional properties, QOS attributes, and corresponding constraints. The meta-model of this qualification process is detailed in section 3.2.

The output of the qualification process is considered as an input to the EAKR process annotated with number (2). Each resulted EACP profile is annotated using semantic annotations related to a specific ontology which is detailed in section 4, and offers a standard vocabulary for the produced capability profile and a formal language to reason about software semantic annotation tags and infer new knowledge to answer to future business needs. This output is stored in a repository annotated in Figure 1 called EAKR gathering all software capabilities resulted from the qualification process. A continuous quality check is applied on already qualified solutions to bring up to date quality indicators. The design steps of the EAKR and examples are presented further in section 4.

To enable the discovery and reuse of already qualified solutions in the EAKR, the process annotated with number (3) and called "Exploitation Process" carry out the discovery and exploitation of service-oriented open source solutions and artifacts (such

as requirement templates produced during last exploitation instances) by considering a process which includes a requirement engineering and architecting actions for formalizing and consolidating the requirements of an end-user as depicted in Figure 1.

During the exploitation process, user requirements are expressed following a proposed template described later in section 3.4. The activity "Exploitation Plan" helps to guide the end-user to fetch and match his business requirements described in different levels following the ADM-based Method of TOGAF with qualified software components in the EAKR to consolidate the requirement with architecting actions and better evaluate the existing building blocks to reuse. Furthermore, after validation of each exploitation level, this exploitation activity saves the requirements gathered as artifacts in the EAKR to leverage these artifacts in next exploitation activity and serve as examples in the definition of the requirements.

The result of this exploitation plan is a set of Solution Building Blocks that correspond to the services that meet expressed needs. These SBBs serve as inputs to the next activity annotated by "Rankings of SBBs" in Figure 1. This activity ranks theSBBs based on a specific exploitation model that end-user specifies and based on a proposed template presented in section 3.4 where QOS parameter measured during the qualification process are used in this task for the ranking.

The resulted ranking enables to select the first SBB that corresponds best to the business function required by the end-user. However, for the business functions that don't correspond to any existing SBBs due to the one-one selection algorithm of the proposed methodology, we propose a tool that helps to generate the source-code skeleton to develop the required API and facilitate the development process. This activity is annotated as "Generate the remaining APIs" in Figure 1 and is based on the proposed ISO 16100 Capability Profile where the developer describes only the functionality and a project skeleton could be downloaded to develop the API. This tool is presented in previous work in [73].

At this stage, and as an output, the selected technical services (and the remaining APIs to develop) are orchestrated to generate an implemented BPMN, representing the prototype needed by the end-user and enables to evaluate the desired business application based on selected service-oriented open source software.

In what follows, we present in detail the meta-model of the EACP Framework, each process and related tasks, along with the proposed EA-based qualification models and requirement templates for the exploitation process.

3.2. *Meta-Model for Software Qualification and Exploitation for Reuse*

The proposed meta-model is depicted in Figure 2 and gathers functional and non-functional specifications; the organizational impact of a feature; and it links the features to their related physical components. The proposed meta-model as depicted in Figure 2 is composed of 6 packages:

(1) Organization package: Composed by the organizational unit, with its related business goals and objectives that guided the development of existing service-oriented software.
(2) Architecture Building Block (ABB) package: This entity is constructed according to the life-cycle creation of ABBs based on the Architecture Development Method (ADM) of TOGAF. ABBs are divided into 2 categories. The first level of an ABB which is initialized during phase B of the ADM, describes the business problem for which this component was developed, it's implementation specifi-

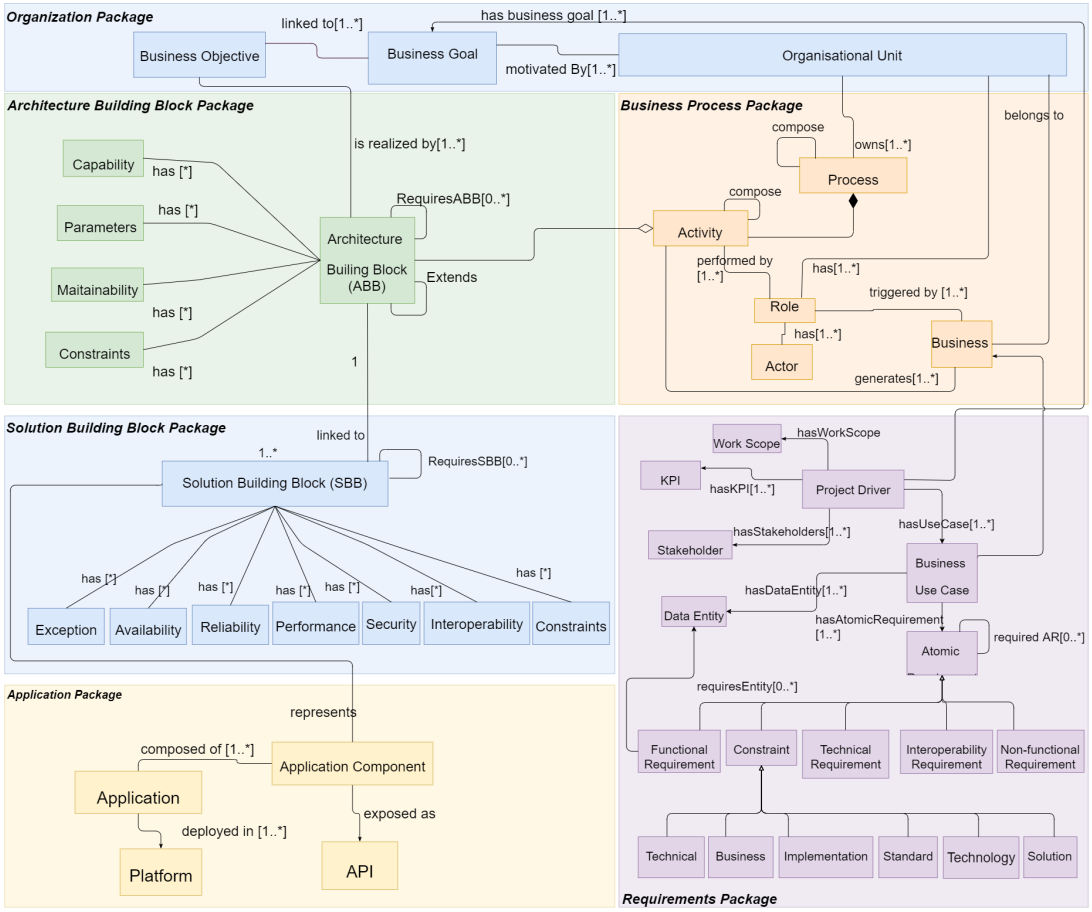

Figure 2. Proposed meta-model

cation, standards used, and the stakeholders. The second level of an ABB enriched in phase C, contains more details and concerns the operational vision of the component. It defines the business function of the ABB, its attributes and constraints, data and application interoperability requirements, and design-time quality attributes.

(3) Solution Building Block (SBB) package: SBBs represent the physical equivalent of ABBs and describe the component exposed by software. The SBB is linked to the exposed service or API for instance over the web in case of a REST-based application. This latter is defined by the URI, the HTTP method needed to get access to the resource, the related parameters and the serialization used in communication (for instance JSON). It defines run-time quality attributes (for instance performance and availability) which might be updated according to the defined frequency for each attribute. Other transversal attributes are defined such as the Security setup of the software exposing the service such as authentication or authorization parameters.

(4) Application package: Describes the technical requirements of the service-oriented software in general with its exposed components (for instance REST services). It describes also the execution environments on which the application is running (e.g Platform class).

(5) Business Process package: This package is used in the exploitation phase and rep-

resents the "to-be" business application to realize. It is composed of the activities that compose the new business application, the roles, and actors concerned by the activities and the event that triggers the process.

(6) Requirements package: This package is used in the exploitation phase and represents the requirements elicitation process, helping to guide the developer or the architect during the engineering lifecycle. The requirements are elicited in each phase of the ADM, going from the definition of project driver to the definition of the use cases and requirements in different levels (functional, non-functional, technical, interoperability requirements and constraints with its different level such as implementation, technical, business, standard, and technology constraints).

The resulted EACP profile instances are saved in the EAKR as an ontology instance, to discover and reuse when developing the business application based on specified user requirements. The related Ontology to EAKR repository is presented afterward in section 4.

3.3. *Qualification Process*

The qualification process is composed of three main activities as illustrated in Figure 1 and follows a bottom-up approach, enabling to qualify existing SBBs and then define the ABBs and motivation aspects reflecting the business goals and objectives targeted by the qualified software. To avoid severe unexpected consequences for organizations that aim to reuse service-oriented software in future business development due to quality metrics not in line with business requirements, the first activity concerns the quality assessment of the software in general and its related services in particular.

3.3.1. Quality Check

The overall quality and subsequent success of an application design depends on how it responds to a range of quality attributes such as security, reusability, and performance. In case of services or Web APIs, QOS parameters such as availability, response time, reliability, throughput and documentation, are among the most important ones [74]. However, the design of services is influenced by the environment, the context and other decisions made by service designers [75], and this may lead to violations of quality principles known as antipatterns [75]. Thus, to guide an organization in a future exploitation and reuse of any solution, whether it's an internal or external company's solution, we aligned different metrics from ISO 25010 [76] recommended technical indicators, extended with best practices from Microsoft® Application Architecture Guide [77] and TOGAF Technical Reference Model [78] which is universally applicable used to build any system architecture and, therefore, is considered later on in the exploitation phase. Obviously, the proposed meta-model may be enhanced with new indicators if necessary.

In the following, we present the categories of service quality indicators used in the construction of the meta-model in run-time: (i) Performance: Indicates system responsiveness to execute an action within a given time interval; (ii) Availability: It defines the percentage of time that the service is up and working; (iii) Reliability: It defines the ability of a system to remain operational over time; (iv) Security: It defines the ability of a system to prevent accidental or malicious actions outside of the designed usage and aims to protect assets and prevent unauthorized modification of

information.

The aforementioned attributes belong to runtime category and are linked to SBB level as depicted in Figure 1. Like any software system, services or Web APIs design should comply with a set of guiding and widely recognized design quality standards such as cohesion, coupling, modularity, and complexity ([79], [80], [75]). Design decisions and poorly planned changes may result into different anti-pattern instances such as god object web services and fine-grained web services which are the two most common antipatterns in web services [81]. God object web service aggregates too many methods into a single service and is not easily reusable due to its low cohesion. Regarding the too fine-grained Web service, it offers few operations and whose communications and maintenance outweighs its utility. Therefore, we selected three design-time metrics proposed in the literature for the selection of best candidate component to reuse ([79], [80], [75], [82]), and considered as best practices in ISO 25010 quality characteristics and Microsoft® Application Architecture Guide. They are linked to the ABB level and corresponds to the maintainability category. Maintainability defines the cohesion, coupling and complexity of the component.

The aforementioned attributes in design and run-time will serve to define the consumption model or the accepted thresholds for each quality indicator of the targeted system. It will enable also to rank the SBBs that offer the same functionality and choose the best candidate that has the highest compatibility level and that fits the specified conditions about non-functional requirements.

Once these indicators are measured, we go further in the qualification process. We present in the following the different qualification levels derived from the meta-model. These levels target the different criteria presented in Table 1. It enables at the end of this process to get a wider view qualification with a specific view about the functional specification, technical specification, organizational level, related QOS measured, the related technology and environment platform, the implementation specification along with the standard and constraint specifications if it exists.

3.3.2. Enterprise Architecture-based Qualification

Once the measurements are realized, a bottom-up qualification approach is carried out. The first activity concerns the retrieval of the common software unit requirements with the functional specification that concerns each technical service. This first qualification level is based on the ISO 16100 standard. It enables also to assess the qualification process before going to the next EA model-based qualification task.

Each exposed service is considered as software unit in terms of ISO 16100 specification. A software unit to be profiled must be analyzed and a template must be filled to build a profile. This profile describes the action performed by the exposed service and offers a functional specification as for instance input and output, data types, class name, method or function name. Furthermore, it describes the technical requirements of the application in general. Figure 3 describes the conceptual structure of the common part of an ISO 16100 Capability Profile where we can find for instance information about the vendor of the software, the version, the required computing facilities to run this software and pricing data if applicable.

Once the specific part of the profile is produced for each service, it is used in the next task by the EACP as depicted in Figure 4. EACP enables to:

- Gather the common software unit resulted from the ISO 16100 Capability Profile, with the technical specifications of the exposed component (in this case the

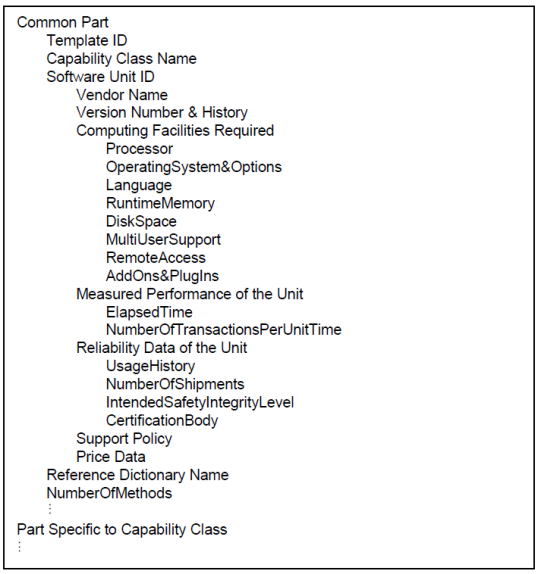

Figure 3. Common part of an ISO 16100 Capability Profile [83]

service). It describes also the execution environments on which the application is running. The related meta-model to this qualification level is depicted in Figure 2-Application Package.

- Link between the SBB and the corresponding physical component. This latter is related to the resource exposed over the web and accessed by a URI as it is the case of the service-oriented software. It defines also the related non-functional specification and QOS with all technical indicators measured during the process presented in section 3.3.1, along with the constraints and technologies considered during the development of the qualified software. The related meta-model to this qualification level is depicted Figure 2-Solution Building Block Package. A derived model of this package is depicted in Figure 5.
- Link the feature from the business point of view considered as ABB to the related SBB. There is a definition of the business problem for which this feature was developed, it's implementation specification, standards used, interoperability requirement and the stakeholders and constraints that should be respected when developing this component such as the solution of infrastructure constraints. In addition, dependencies between features and between physical components are also considered. The related meta-model to this qualification level is depicted in Figure 2-Architecture Building Block Package.
- Define the organizational impact of the exposed service to an organization, by describing the business goals and business objectives of the related feature and the organization using it in case the qualification concerns an internal solution of an enterprise. The related model to this qualification level is depicted in Figure 2-Organization Package.

Once all the EACP models are completed, it results a capability profile for each service, and all linked to specific service-oriented software that is ready to be discovered for reuse purpose. This capability profile enables to cover all the criteria defined Table 1 except for the semantic annotations and reasoning capabilities. These are presented in detail in section 4 and are considered as a basis for the design of the EAKR. But

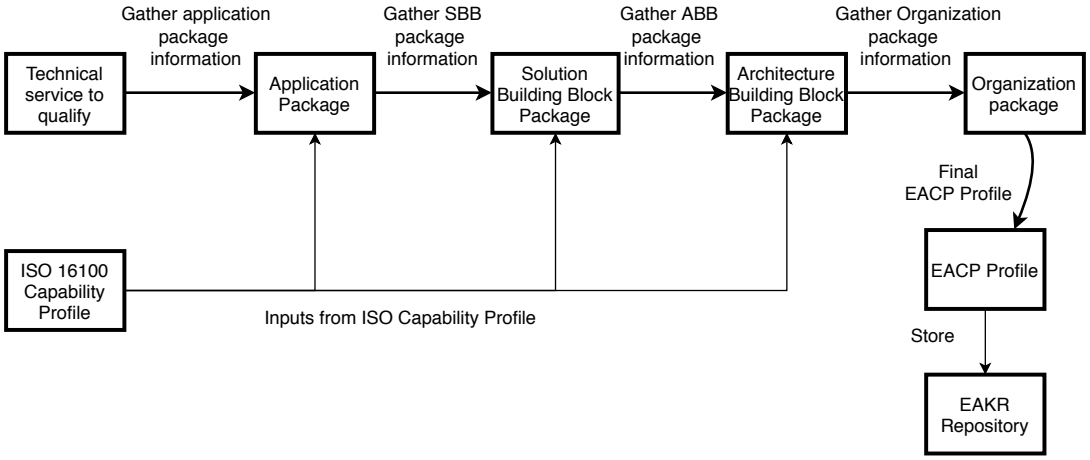

Figure 4. Qualification process to create the EACP Capability Profile

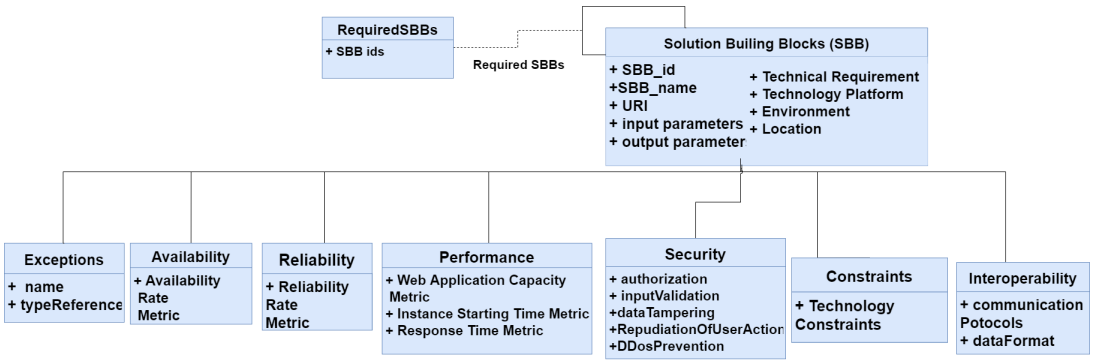

Figure 5. Proposed Model for SBBs

before going further in the design of this repository, we present in what follows the last process of the EACP Framework which is the exploitation process enabling the discovery and reuse of the qualified solutions based on the proposed EACP Capability Profile.

3.4. *Exploitation Process*

In order to reach the objective of designing a new business application based on qualified services of the EAKR, the developer or architect goes through a proposed requirement elicitation process. This latter is enrolled in an engineering cycle structured in several phases inspired by the TOGAF Framework and its ADM method. It concerns four phases starting by the architectural and requirement vision, going through the business architecture phase, data, application, and technology architecture phase, leading to the generation of an implemented BPMN which consumes the qualified services if a match confirmed.

Figure 6 refers to the task "Exploitation Plan" in the process number 3 of the EACP Framework (Figure 1) corresponding to the exploitation process. It depicts the proposed exploitation plan guiding gradually the architect or the developer from the analysis till the discovery of SBBs and the production of the targeted business application. In the following, we present in detail the actions to realize in each phase,

enriched with a concrete example to strengthen the understanding.

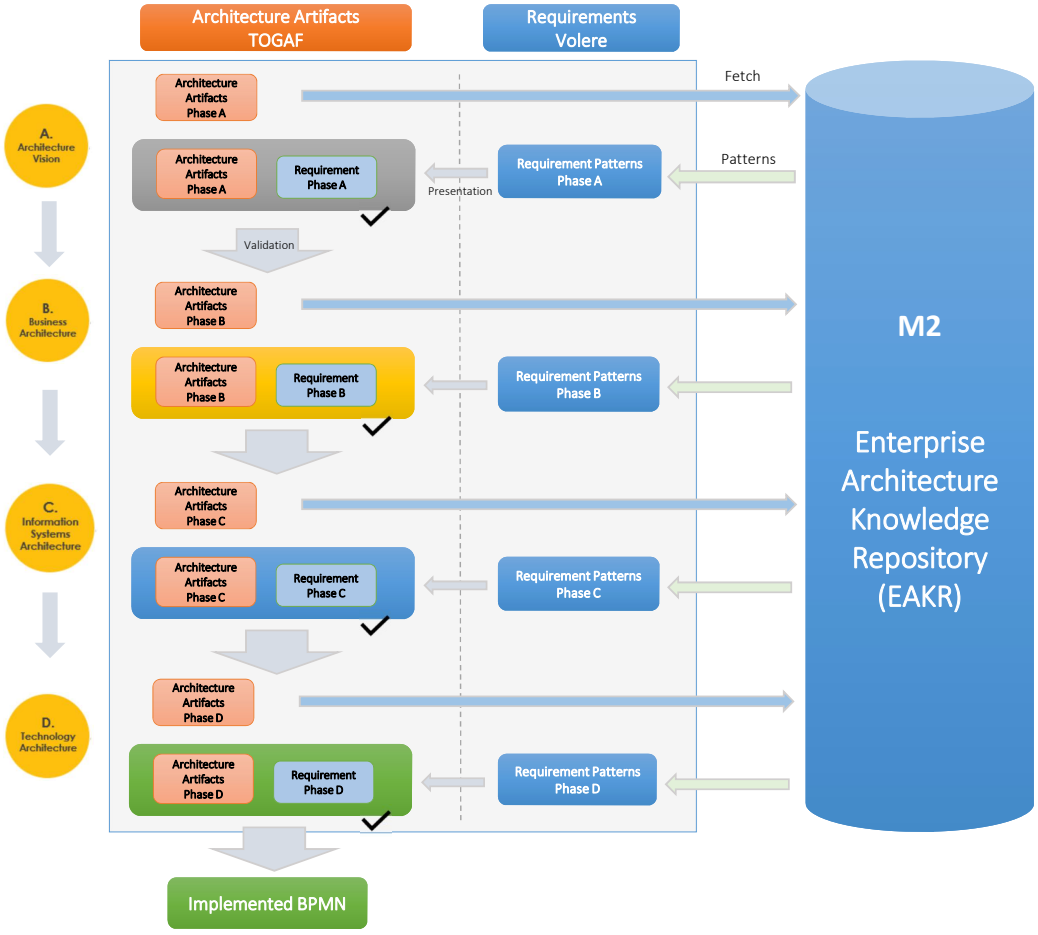

Figure 6. Exploitation Plan

3.4.1. Phase A: Architecture vision

The exploitation process starts by the requirement elicitation for Phase A (Architecture vision). The objective of this phase is to develop a high-level vision of the business value to be delivered as a result of the project. This phase is focused on gathering the business goals and related objectives of the targeted project. Other architecture artifacts are asked for the developer during the process to structure the project drivers, such as defining the organizational model of the company, the concerned stakeholders, and KPI helping to evaluate the targeted business application. Based on the architectural artifacts of phase A, which corresponds to the definition of the business goals and objectives from the developer, we fetch requirement patterns constructed during previous projects to guide and offer support to the developer for the upcoming requirement definition. If templates are found, they are presented to the developer as examples for the formalization of next inputs. The developer formalizes his requirements by defining, in complementary to the architecture artifacts, the project drivers such as the business actors, the client, and customer if applicable. These inputs help to consolidate the elicitation phase and redesign his requirements before going further

in the process. All the artifacts once validated are saved in the EAKR to be reused if needed in future exploitation process. An exploitation algorithm (Figure 7) is proposed for this phase. This latter uses business objectives and goals as filter lever, but it can be modified according to the preferences of the user if he doesn't wish to rely on these two inputs.

Data: *requirementPhaseA* : Architecture artifacts of phase A
Result: *projectDriversPhaseA* : A list of Project Driver ∈ EAKR with *projectDriversPhaseA.BusinessGoals* ⊆ *requirementPhaseA.BusinessGoals*

```
1  begin
2      projectDriversPhaseA ⟵ ∅;
3      for pdriver ∈ EAKR do
4          for bGoal ∈ pdriver.BusinessGoals do
5              if requirementPhaseA.BusinessGoals.find(bGoal) then
6                  for bObjective ∈ bGoal.BusinessObjectives do
7                      if ( bObjective ∈ requirementPhaseA.BusinessObjectives) & (pdriver ∉ projectDriversPhaseA ) then
8                          projectDriversPhaseA.add(pdriver) ;
9                      end
10                 end
11             end
12         end
13     end
14 end
```

Figure 7. Phase A: Requirements template selection

Figure 8 illustrates the models of architecture artifacts for phase A. The main inputs are the definition of business goals and objectives of the targeted system, along with stakeholders that define people who have an interest in the targeted system and whose inputs is needed to build the product.

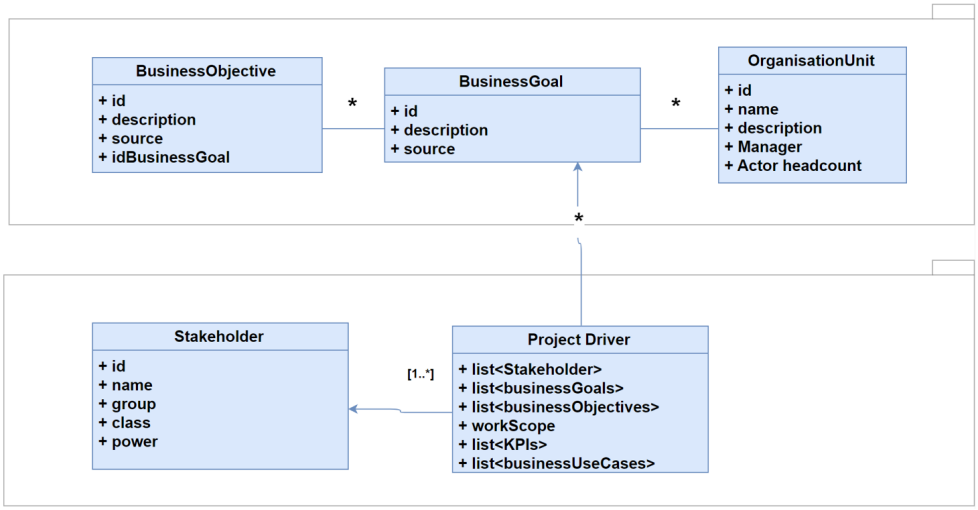

Figure 8. Models of Architecture artifacts for Phase A

3.4.2. Phase B: Business Architecture

The objective of this phase is to develop the target business architecture that describes how the enterprise needs to operate to achieve the business goals previously defined and responds to stakeholder concerns.

The most important architectural artifact in this phase is the high-level business scenario. This latter is designed using the BPMN modeling tool which is a standard for business process modeling. This first high-level modelization is designed using the

ArchiMate [2] core language which is recognized as a standard for EA modeling by the Open Group [81]. This high-level modelization helps to define the business-entity relationship to know which entities are needed for every business action or behavior.

This is designed based on the three aspects proposed by ArchiMate: (i) The Active Structure Aspect which represents the structural elements or the subject; (ii) The Behavior Aspect which represents the behavior performed by the active structure; (iii) The Passive Structure Aspect which represents the objects on which behavior is performed.

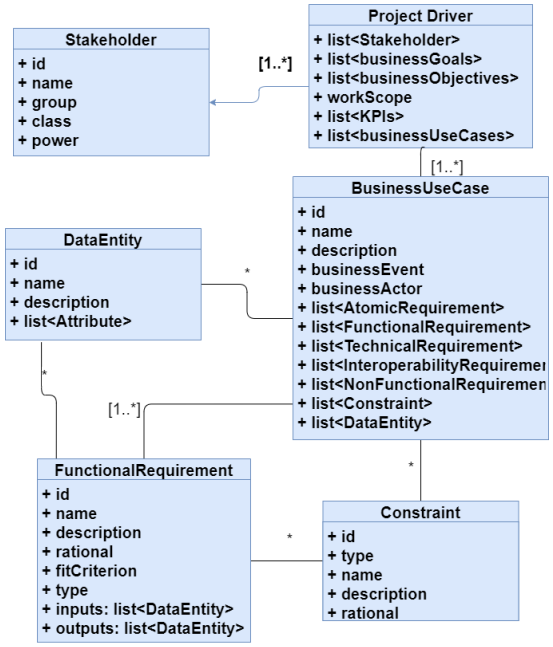

Figure 9. Models of Architecture artifacts for Phase B

This business model is enriched with other architectural artifacts such as the actor catalog updated with their related roles, and the definition of the architecture requirements specification which form a major component of an implementation contract and provides quantitative statements as required in ADM's Phase B outputs. It requires the definition of the implementation specifications to guide the development work, implementation standards in case the implementation should follow some specific standard. Figure 9 depicts the model of the different architecture artifacts managed during this phase.

Once these elements are defined, a request is sent to the EAKR to fetch requirement templates of phase B produced during last projects to be reused. The proposed algorithm for the requirement template selection is presented in Figure 10. The aim is to guide the developer to define the project constraints and the functional requirements. The resulted templates from the algorithm in Figure 10 present the scope of the existing projects related to the actual context, the business events connected to the actual business scenario, the use cases of the project or solution, and a set of functional requirements related to the selected use cases and their type, i.e service type component related to an SBB or user task activity if it is a user action. These resulted templates are presented to the developer to guide him during this phase B for

[2]https://www.archimatetool.com/

```
Data:
requirementPhaseB : Architecture artifacts of phase B
projectDrivers : list of project drivers ∈ EAKR
requirementPhaseB.BEM : Business Entity Modeling defining the needed use case and managed Data entity models
constraints : list of defined constraints during phase B (Solution, Implementation, Standard and Interoperability constraints)
Result: projectDriversPhaseB : A list of Project Driver ⊆ projectDrivers
1  begin
2      projectDriversPhaseB ⟵ ∅;
3      foreach pdriver ∈ projectDrivers do
4          foreach useCase ∈ pdriver do
5              foreach behavior ∈ requirementPhaseB.BEM.Behaviors do
6                  if (behavior = useCase ) & (behavior.dataEntities ⊆ useCase.dataEntities) then
7                      if (constraints != ∅) & (constraints ⊈ useCase.constraints) then
8                          go to next use case;
9                      else
10                         projectDriversPhaseB.add(pdriver);
11                     end
12                 end
13             end
14         end
15     end
16 end
```

Figure 10. Phase B: Requirements template selection

the consolidation of his requirements. It helps to offer support for defining and consolidating the use-cases and related functional specifications. The proposed template is inspired from the Atomic Requirement Template which is presented by Robertson et al. in [84]. This phase B ends with well formalized, testable and categorized as user or service task functional requirements.

3.4.3. Phase C: Information Systems Architecture

The objective of Phase C is to develop the target information system architecture. It involves a combination of data and application architecture. Therefore, this phase is composed of two sub-phases:

3.4.3.1. Data Architecture. This phase enhances the definition of the relation between data entities and targeted business functions previously defined in phase B. We have already depicted in Phase B the models that enable to link data entities associated to each business function. Then, the next needed action is to define the properties of each business entity involved in the business functions using relevant data models such as the Class Diagram in UML. Additional architecture requirements specifications are as-well formalized such as Data Interoperability and Technology Architecture Constraints. These constraints have the same description template as for requirements. The data interoperability requirement is needed to formalize specific needs for security policies as for example input validation or for data format and serialization. Regarding the technology architecture constraint, helps to identify constraints on the infrastructure about to be designed. During this phase, architecture artifacts and requirements are defined at the same time because we are reaching the low-level description regarding the business application to develop. Based on these inputs, a request is sent to the EAKR to fetch and map the ABBs and business functions using the defined data entities, and which are compliant with the constraints if defined. Figure 11 depicts the algorithm that aims to retrieve the ABBs and related data entities to provide support and guidance during this phase. Moreover, the re-

lated SBBs and their corresponding applications are gathered to highlight potential problems of integration.

```
   Data:
   requirementPhaseC : Architecture artifacts of phase C - Data Architecture
   projectDriversPhaseB : list of selected project drivers resulted from phase B
   requirementPhaseC.EBFM : Entity-Business Function Matrix
   constraints : list of defined constraints during phase C - Data Architecture (Technology constraint)
   Result: MAP(BF, ABBsPhaseC) : A MAP of Business Function and a list of Architecture Builiding Blocks ⊆ EAKR
1  begin
2      ABBsPhaseC ⟵ ∅;
3      foreach ABB ∈ EAKR do
4          foreach businessFunction ∈ requirementPhaseC.EBFM do
5              if (ABB.businessFunction = businessFunction ) & (ABB.parameters.inputs ⊆ businessFunction.entities ) then
6                  MAP.add(businessFunction, ABB);
7                  ABBsPhaseC.add(ABB);
8              end
9          end
10     end
11     foreach ABB ∈ ABBsPhaseC do
12         If (ABB.DataInteroperability! = requirementPhaseC.DataInteroperabilityReq )
13         //highlight potential problems of integration
14         requirementPhaseC.componentInteraction.add(ABB) ;
15     end
16 end
```

Figure 11. Phase C - Data Architecture: Requirements template selection Algorithm

3.4.3.2. Application Architecture. This phase concerns the Application Architecture artifacts and related requirements. The developer or architect is guided to define technical requirements based on the same template as for the atomic requirement . Technology or infrastructure constraints and application interoperability requirement are either defined in this phase. Those constraints are added to previous ones to fetch the SBBs and related applications. Figure 12 depicts the algorithm that aims to retrieve the application list according to the requirements and constraints defined in this phase. The resulted SBBs and related application offer the first overview of existing applications and related services to reuse. These solutions fit the requirements and constraints from a functional, technical and technology constraint side.

Up to this requirement level, a first version of the targeted business application based on BPMN 2.0 is defined. The user and service tasks are designed and a link between service tasks and a set of existing services is performed based on the elements defined during previous phases.

3.4.4. Phase D: Technology Architecture

The last phase D is about the technology architecture artifacts. The objective of this last phase is to define the basis of the implementation work. As part of phase D, the developer or architect needs to consider what relevant resources are available in EAKR to ensure that the target system will meet some or all the requirements and constraints. It is important to recognize that in practice it will be rarely possible to find and reuse components that reach 100% coverage of all defined requirements and constraints. During the previous phase C, technical and technological constraints are formalized. These latter are considered during this phase D when matching the final SBBs, enriched with non-functional properties that are defined based on the atomic requirement model enabling the definition of the Quality of Service needed from the existing services.

```
Data:
requirementPhaseC : Architecture artifacts of phase C - Application Architecture
requirementPhaseC.EBFM : Entity/Business-function matrix
MAP(BF, ABBsPhaseC) : A MAP of Business Function and a list of Architecture Builiding Blocks selected in Phase C - Data Architecture
constraints : list of defined constraints during phase C - Data Architecture (Technology constraint)
Result:
MAP(BF, ABBsPhaseC2) : A MAP of Business Function and a list of Architecture Builiding Blocks ⊆ ABBsPhaseC
requirementPhaseC.ApplicationList : List of applications that are required
1  begin
2      ABBsPhaseC2 ⟵ ∅;
3      for BF ∈ MAP(BF, ABBsPhaseC) do
4          for ABB ∈ BF.ABBsPhaseC do
5              if (requirementPhaseC.Constraints ∈ ABB.SBB.Constraints) & (requirementPhaseC.AppInteropRequirement =
                 ABB.AppInteropRequirement) & (requirementPhaseC.TechnicalRequirement = ABB.SBB.TechnicalRequirement ) then
6                  ABBsPhaseC2.add(ABB);
7                  requirementPhaseC.ApplicationList.add(ABB.SBB.applicationComponent.application);
8              end
9          end
10         MAP.update(BF, ABBsPhaseC2);
11     end
12     //Create BPMN Process based on Entity/Business-function matrix;
13     Init pool;
14     Init users;
15     foreach useCase ∈ requirementPhaseC.EBFM.behaviors do
16         Init start event;
17         foreach businessFunction ∈ useCase do
18             if (businessFunction.componentType = serviceTask ) then
19                 Generate service task;
20             else
21                 Generate user task;
22             end
23         end
24         End event;
25         Generate sequence flows;
26     end
27 end
```

Figure 12. Phase C - Application Architecture: Requirements template selection Algorithm

As a result, the retrieved SBBs, if they match, reflects strongly the defined requirements and constraints. This helps to implement the business process already produced during the last phase with the final SBBs, and related services with their service endpoints to support the business application. Figure 13 depicts the algorithm that aims to retrieve the list of SBBs according to the requirements defined in this phase.

3.4.5. SBBs Ranking and Selection phase

The template provided during phase D about non-functional requirements reflects the consumption model needed and is used to rank and select the Solution Building Blocks that have the highest compatibility level. The QOS parameters are calculated during the quality assessment as described in section 3.3.1. As illustrated in the algorithm of Figure 13, services are ranked according to the consumption model before selecting the final SBB and generate an implemented business process. We have introduced a priority parameter during the definition of the non-functional properties of phase D, to personalize the priority of each QoS during the ranking process. To facilitate the priority definition, we choose three priorities level. The value 1 refers to low priority, value 2 to medium and value 3 to high priority. In case no priority property is defined in the template, 1 is the default value assigned. There are two calculation methods as already defined in [38] for QOS ranking. For some QOS parameter, the highest the value is, the better the SBB is, for instance the availability rate metric. Whereas for some other QoS as for instance the response time metric, the lowest the value is, the better the SBB is. To calculate the ranking for each SBB, we consider that QoS_{upper} and QoS_{lower} are respectively the ranks regarding the two kinds of QOS parameters.

```
Data:
MAP(BF, ABBsPhaseC2) : A MAP of Business Function and a list of Architecture Builiding Blocks resulted from Phase C - Application Architecture
NFRequirementPhaseD : Non-functional requirement defined in Phase D - Technology Architecture
Result: MAP(BF, SBBsPhaseD) : A MAP of Business Function and a list of Solution Builiding Blocks
1  begin
2      SBBsPhaseD ⟵ ∅;
3      for BF ∈ MAP(BF, ABBsPhaseC2) do
4          for ABB ∈ BF.ABBsPhaseC2 do
5              for nfr ∈ NFRequirementPhaseD do
6                  if ( nfr.type ∈ ABB.SBB) & (ABB.SBB.checkNFR(nfr)) then
7                      SBBsPhaseD.add(ABB.SBB);
8                  end
9              end
10         end
11         RankSBBs(SBBsPhaseD);
12         MAP.add(BF, SBBsPhaseD);
13     end
14     UpdateBpmnWithSbbs(MAP);
15 end
```

Figure 13. Phase D - Technology Architecture: Requirements template selection algorithm

Let SBB_i be a solution building block and Q_j be a QOS parameter.

$$QoS(SBB_i, Q_j) = \begin{cases} QoS_{upper} & = & \frac{Val(SBB_i,Q_j)}{Max(Q_j)} * Priority \\ QoS_{lower} & = & (1 - \frac{Val(SBB_i,Q_j)}{Max(Q_j)}) * Priority \end{cases}$$

Where:

- Val is the value of the QOS parameter for a given SBB
- Max is the maximum value of the QOS parameter among all selected SBBs related to a Business Function
- And, $Priority$ is the priority previously assigned to the QoS parameter by the user during Phase D

Finally, the global score of an SBB is represented by $R(SBB_i)$ where :

$$R(SBB_i) = \sum_{j=0}^{\infty} QoS(SBB_i, Q_j)$$

The result of this ranking process is used during the generation of the implemented BPMN, where for each business function (only service tasks), we assign the first ranked SBB to the related task. In the next section, we present the development method of the EAKR and its related ontology based on the proposed meta-model.

4. Enterprise Architecture Knowledge Repository

We note that actually, there is no agreed methodology for the development of ontologies and no consensus on how ontologies should be evaluated. To develop our ontology, we considered the Uschold and King's methodology presented in [85]. In the following subsections, we present the different steps that we followed in our work.

4.1. *Design of the EACP Ontology*

First, we identified some state-of-the-art ontologies and selected those that cover our needs, and are relevant to our domain. We selected BFO which is a top-level ontology and four domain ontologies, namely OWL-S [86], TOGAF ontology [87], BPMN 2.0 Ontology [88] and IAO [89]. Third, we managed the selected foundational and domain ontologies, by integrating and extending in a coherent way the different ontologies into the targeted EACP ontology using the Protégé Ontology Editor. Finally, we evaluated the consistency and inferences of the resulted model using the Fact++ reasoner. In the following subsections, the *"Italic"* form is used to designate semantic classes and ontology relationships. Regarding the latter, labels such as BFO, OWL-S or TOGAF are used instead of the URI for the stake of legibility.

4.2. *Reuse of Existing Ontologies*

We used the following ontologies to standardize the representation of the information that are contained in the proposed meta-model. Indeed, no existing ontology can cover all the information of the meta-model. We selected the existing domain ontologies by considering the following aspects: (i) The free access and availability of the ontology on the Web; (ii) The proper definition of ontology classes and relationships to ensure appropriate reuse of the extracted entities; (iii) The coverage of the entities of the targeted domain to design a minimum set of terms; (iv) The stability of the selected ontology, so that future changes or updates do not affect the proposed model;

We used Protégé to analyze and select classes and relationships to be extracted. We adopted a modular architecture, consisting of the main ontology importing several modules. The main motivation for such a modular architecture was the ability to easily re-extract entities from existing ontologies. However, several situations must be considered, depending on the reused ontologies. The following subsection illustrates how we integrated TOGAF, OWL-S, BPMN and IAO ontology modules into a coherent semantic model, and how we used the imported OWL-DL descriptions to automatically query the resulted EA knowledge repository using a semantic reasoner.

4.3. *Construction Steps of the EACP ontology*

We depict in this section the construction steps of the EACP ontology from the meta-model presented with regards to the main steps of our ontology building approach.

4.3.1. Step 1: identification of existing ontologies

In the proposed semantic model, we described all entities of the meta-model. The resulted ontology contains 230 classes, 78 object properties, and 30 data properties.

The targeted modular EACP ontology defines its entities and relations under the BFO framework. This latter is designed to support information analysis, retrieval and integration from different semantic resources [90]. To map EACP entities to BFO, we have realized an alignment work based on the methodology described in [90].

To the best of our knowledge, BFO ontology has not yet been used for the integration of TOGAF enterprise architecture components although, some research works as for instance [91] and [92] has expressed the need to link the information system modeling to an upper-level ontology to facilitate the integration of existing various meta-models and resolve inconsistency problems. Moreover, we chose to reuse TOGAF 9 ontology for

three main reasons: first, TOGAF 9 ontology, developed in Knowledge-Projects [3], is the most suitable terminological model that represents meta-data of TOGAF 9 artifacts. Second, it has been used in other research works in [93] and [94], to semantically manage and share enterprise architecture information and third, TOGAF 9 ontology meets some of main modeling needs that are expressed in the meta-model as the description of entities that describe Business Architecture Component (e.g., actor, organizational unit, objective, goal, and event). In our work, we used the OWL formal language in the specification of the TBox and in the generation of RDF [95] triples.

4.3.2. Step 2: integration of existing ontologies into the BFO ontology (classes and relations)

To build our OWL ontology, we first analyzed the class hierarchy of our semantic modules (namely, TOGAF, BPMN, IAO and OWL-S). Then, we aligned existing ontologies' classes to the BFO class hierarchy. By doing this alignment work, we improved the definition of key concepts of the TOGAF ontology by making them more clear, for example the class *"TOGAF:core content"* regroups heterogeneous sub-classes as *"TOGAF:function"*, *"TOGAF:process"*, *"TOGAF:capability"*, *"TOGAF:process"*, *"TOGAF:role"* and *"TOGAF:organization unit"*. As we can notice, the meaning of the *"TOGAF:core content"* class as it is defined within the TOGAF ontology is confusing and ambiguous regarding the description of the kind of data that it covers i.e, does it refer to functions? roles? processes? dispositions? objects? or data items? In order to remove this confusion and inconsistency, we first defined the class *"TOGAF:content classification"* as a *"BFO:entity"* and then we defined the classes like *"TOGAF:function"*, *"TOGAF:role"*, *"TOGAF:capability"* as sub classes of *"TOGAF:core content"* and *"BFO:specifically dependent continuant"* given that they describe a continuant that inheres in or is borne by other entities. As consequence, the meanings of theses subclasses are more explicit and clear under the BFO framework.

4.3.3. Step 3: extension of existing ontologies

New 30 additional classes are inserted in the EACP ontology based on the proposed meta-model. For example, we introduced the *"EACP:ABB service category"* class as a direct subclass of *"IAO:data item"* and an indirect subclass of *"BFO:generically dependent continuant"*. Therefore, *"EACP:ABB service category"* is assumed to be a superconcept of the classes *"EACP:capability category"* or *"EACP:interoperability category"*.

We enriched the ontology with semantic axioms in order to define in a concise way classes and relations and to improve the ontology with reasoning capabilities.

In our work, we made some modeling decisions about the description of some unclear TOGAF aspects: we interrelate classes using a min number of relationships than those proposed in TOGAF ontology, this will improve the reasoning performance of our ontology. In TOGAF, there is a mix between class hierarchy and necessary conditions for example, the *"TOGAF:business architecture"* is defined as *"TOGAF:architecture"* and (*"BFO:hasContinuantPart"* some *"TOGAF:business architecture component*. We modified the some OWL restriction with a some necessary condition as explained above. This correction is valid for all the kind of business architecture component of the TOGAF ontology.

Finally, we completed the definition of our ontology with textual annotations that

[3] https://sites.google.com/site/ontologyprojects/home

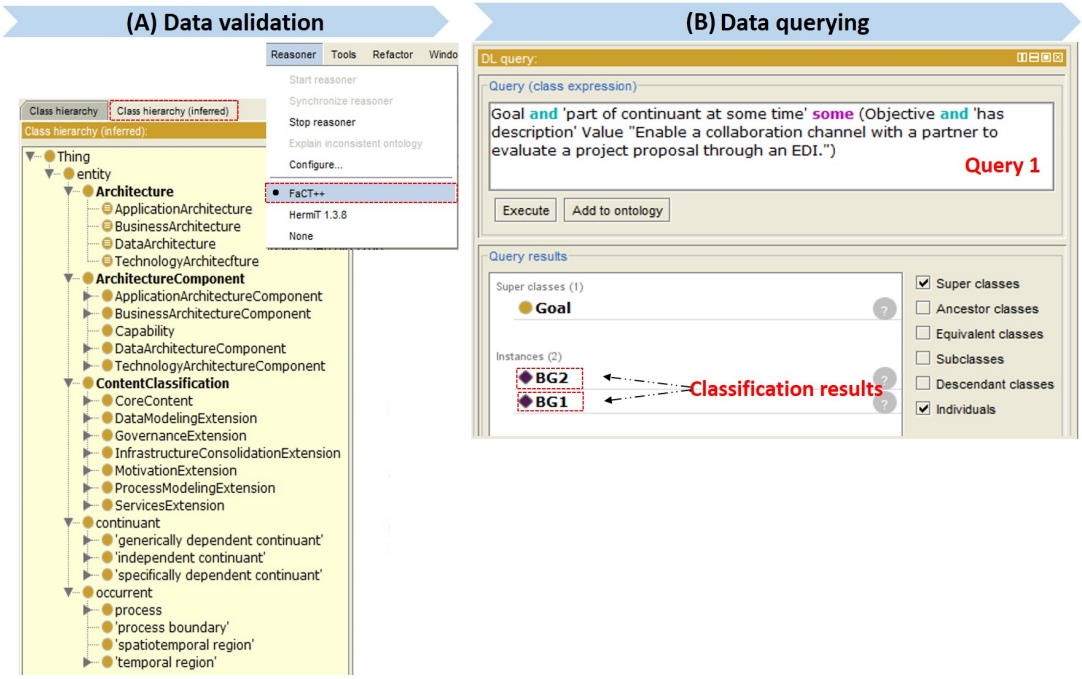

Figure 14. Data validation and querying result on the EAKR repository using the FACT++ reasoner, Protégé Ontology Editor

aim to give the users more details about our ontology components; in this annotation task we based our textual definitions on the documentation of TOGAF 9.

4.3.4. Step 4: evaluation of the consistency

To check the validity and the correctness of the designed ontology we used the Fact++ reasoner, the classification result is presented in Figure 14, no errors are found and all classes and relations are correctly classified.

5. Framework Implementation

We implement the exploitation process of the EACP Framework as a Web Application as depicted in Figure 15. EACP Framework's source code is available at Github repository [4]. The video of this technical presentation is available here [5] and the of the entire use case is available here [6]. We chose AngularJS [96] as a Web Framework that enables the development of single-page applications following the MVC pattern for the front-end environment, and NodeJS Framework [97] which is a popular platform for building server-side Web Applications written in Javascript. Regarding the EAKR repository, we deploy the EACP Ontology along with example of qualified open source solutions from vf-OS [7] and FITMAN project [8] in Apache Jena Fuseki [98].

[4] https://github.com/AbdBelf/EacpFramework
[5] http://bit.ly/eacpTechnicalP
[6] http://bit.ly/exploitP
[7] www.vf-OS.eu
[8] http://www.fiware4industry.com

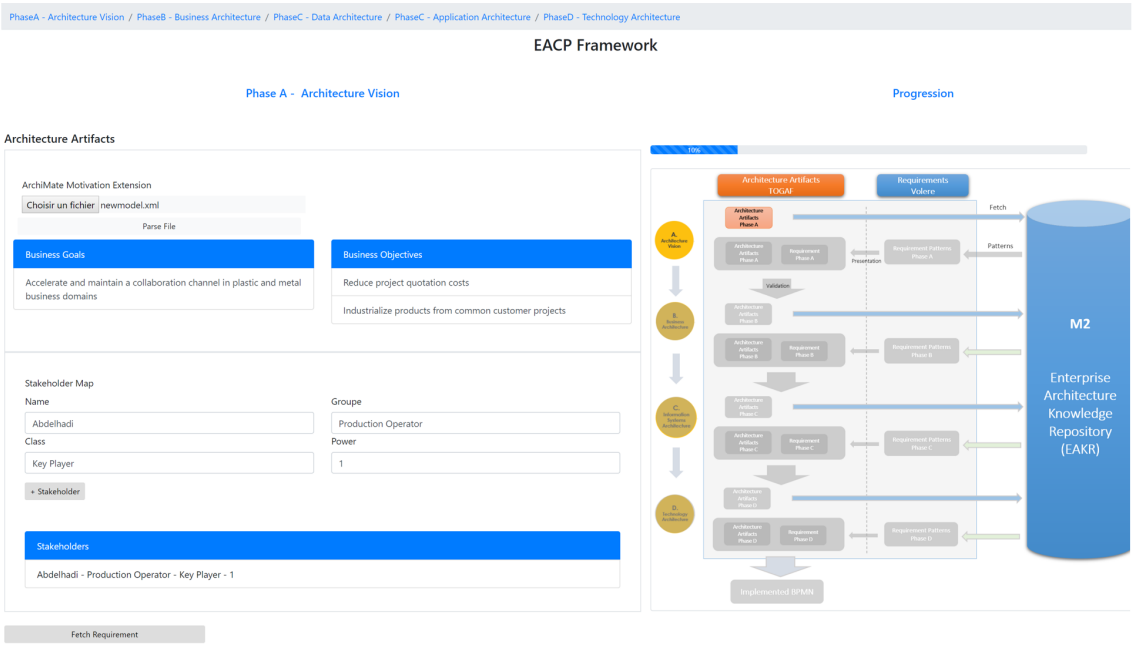

Figure 15. EACP Web Application - Phase A : Architecture Artifacts

In the next section, we present the use case on which we apply the EACP Framework, then we show in what way the developer can interact with the Framework in each phase, the inputs needed, how the information is presented and how the validation occurs.

5.1. *Use Case Scenario*

This use case scenario concerns two companies. The first company is specialized in plastic manufacturing, and the second in metal manufacturing. Both companies have significant expertise in engineering and transforming plastic and metal, respectively, parts using several technologies. Their key business issue is the difficulty to detect the most appropriate business collaboration opportunities among common customer projects. Starting from a CAD description of customer projects, the companies' consortium needs to quickly detect the set of projects that they are enabled to produce. The proposed use case aims to accelerate and maintain a collaboration channel in two complementary business domains. The objectives are to reduce project quotation costs and reduce the delay of customer quote treatment.

Currently, clients send product requests to one of the companies in the form of CAD/PDF files. Then, the chosen company decomposes all the project's features (e.g. parts, dimensions, type of surface, and type of raw material) to understand customer needs to verify the feasibility of the product, and to determine the relevance of the business opportunity in terms of ROI. After the decomposition and if there is a need for subcontracting (especially for multi-physical and complex products), the two companies will carry out a succession of negotiations, explore several ways to reach the client's requirements and submit their best offer to the client. For this purpose, we derive from this use case all needed requirements that we need to develop in the EACP Framework to discover existing services that allow to reduce cost and development

time. In what follows, we present how the developer goes through an architecture and requirement elicitation process to discover existing services and develop the needed business application. We present the inputs needed in each phase of the exploitation process and the screenshots of the prototype in each phase. The video of this use case scenario is available here [9].

5.2. *Phase A: Architecture Vision*

In this phase, one of the artifacts to provide is about the business goals and associated objectives of the targeted project. The offered design possibility is to upload the inputs designed in ArchiMate using the motivation extension. Figure 15 depicts an example from the proposed use case of the motivation diagram. An export in XML format is needed to import it to the EACP Web Application to parse it and retrieve the needed inputs for phase A. Figure 15 depicts the phase A of the Web application, The first column concerns the architecture and requirement elicitation process guiding the developer to consolidate and validate his requirement during this phase. The second column displays the actual phase state and the progression rate of the process. The developer can as well add other stakeholders not mentioned in the motivation diagram to be considered for the next actions.

Once the motivation diagram uploaded to the framework, a parsing of the XML source file is realized to retrieve the defined business goals and objectives. Based on these inputs, a request is sent to the EAKR (see the proposed algorithm in Figure 7) to fetch existing requirement templates guiding the developer during this requirement elicitation phase. Since it is the first instance of this process, we are not supposed to get any template. However, and for illustrative reasons, we defined one template that shares the same business goal to have an example of a template to reuse for defining and consolidating the required requirements for this phase. As we may notice in Figure 16, the requirements needed are the definition of the business context, the client and the customer of the system which is not applicable in this context, and the users that will interact with the targeted system. The retrieved templates are presented in the "EAKR Templates Requirement" side. The process progression column is updated, and the application now is waiting for the validation of the requirements to redirect the developer to phase B of the exploitation process.

5.3. *Phase B: Business Architecture*

Based on ArchiMate Business Entity Relationship diagram, the developer uploads the designed diagram to the architecture artifacts user interface. This latter is parsed to retrieve the actors, the business processes and related data entities involved in each business process or use case. These inputs are considered during the requirement pattern search in the EAKR repository and retrieved using the algorithm depicted in Figure 10.

Requirements specification of phase B concerns the functional requirements and project constraints. Based on the use case list and their related data entities, we fetch the previous project that has been saved to the EAKR based on a string similarity. These existing requirements help to offer support for defining and consolidating the use-cases and functional specifications close to the actual context, the business events connected to the actual business scenario and a set of functional requirements related

[9] http://bit.ly/exploitP

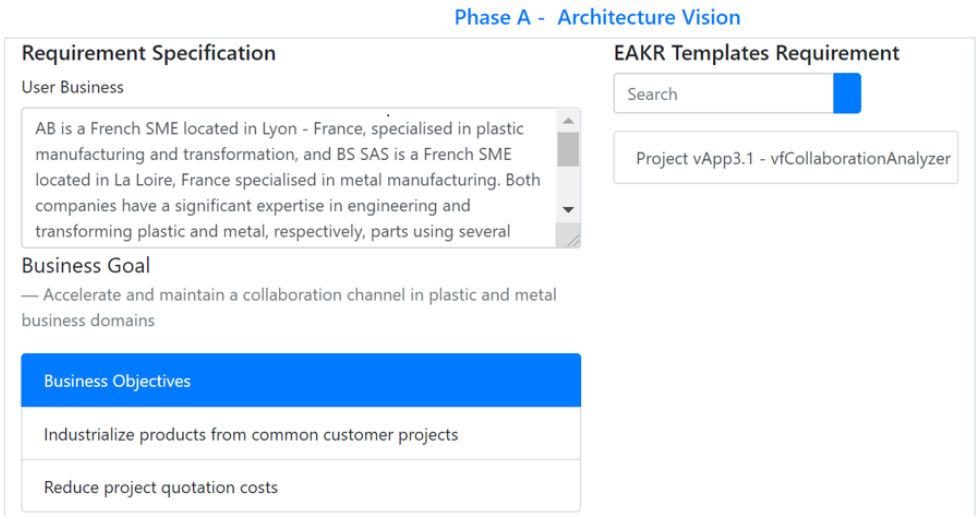

Figure 16. EACP Web Application - Extract of Phase A : Architecture Artifacts

to the selected use cases and their type (i.e service type component related to an SBB or user task activity if it is a user action). In the case of our business scenario, no template has been found but for illustrative reasons, we initialized requirement templates that correspond to the actual business scenario to be reused as depicted in Figure 17.

In this phase, there is a possibility to enrich the functional specifications by adding required constraints or architecture requirements specification such as the specification of implementation or the usage of a specific standard for the future development of the functional requirements. In the context of this proposed scenario, we link an implementation standard to the functional requirement "Visualize CAD file" as depicted in Figure 18.

5.4. *Phase C: Data Architecture*

Data architecture phase enhances the definition of the relation between data entities and targeted business functions previously defined in phase B. Then, the next needed action is to define the properties of each business data involved in the business functions using relevant data models such as the Class Diagram in UML that is serialized to retrieve the entities with their related attributes.

Based on these inputs, the application fetches and maps in the EAKR the business functions with the architecture building blocks using the defined data entities, and which respects the interoperability and infrastructure constraints as defined in algorithm of Figure 11. This latter matches between the defined functional requirements with business functions defined in ABB model. ABBs that corresponds to the conditions are selected with their related SBB and corresponding applications. The objective is to highlight potential problems of integration in case any selected ABB presents data interoperability constraint which is different from the defined constraint in this phase C. The results are presented in the proposed template of the EACP application as depicted in Figure 19. For instance, the qualified service from the EAKR matches with the functional requirement named "Visualize CAD file". It is selected among other qualified ABBs from FITMAN project that offer the same business func-

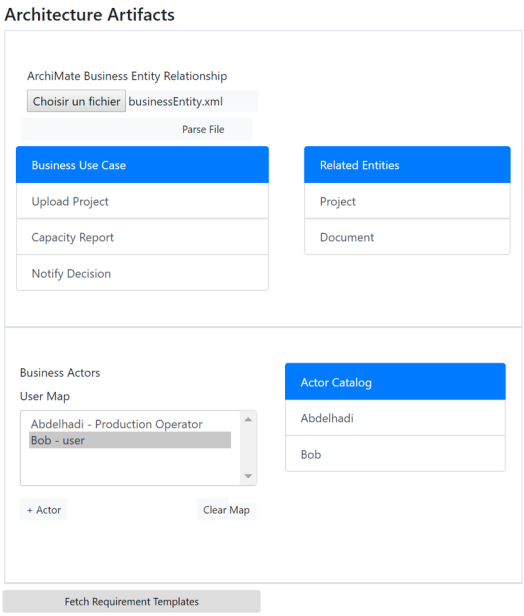

Figure 17. EACP Web Application - Extract of Phase B : Architecture Artifacts

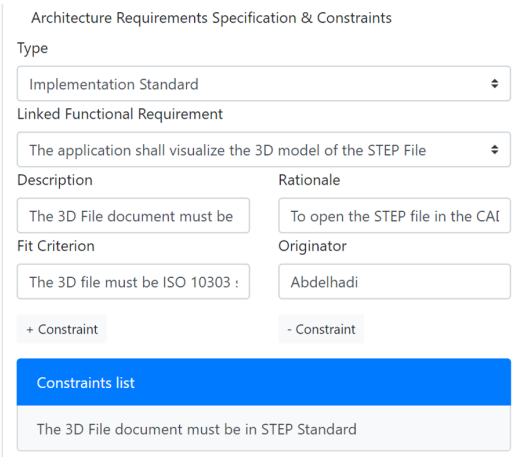

Figure 18. EACP Web Application - Phase B : Requirement Specification

tion, and that respects the implementation standard that was already defined in Phase B.

5.5. *Phase C: Application Architecture*

The developer is guided to define the technical requirement using the same template as for the functional requirements. Technology or infrastructure constraint is either defined. Those constraints are added to previous ones to select the SBBs as described in algorithm of Figure 12.

For instance, in this use case scenario, we define a technical requirement related to the targeted business function "Visualize CAD File". Indeed, we target an SBB which manages the CAD Objects with a specific file extension "STL extension", and that is based on the Javascript library Three.js. Then the action "Fetch ABBs" triggers the selection of the targeted ABBs and related SBBs in the EAKR that respect the defined technical requirement for each business function along with the technology constraints if defined. The result of this action is depicted in Figure 20.

To this level, these inputs enable to download a first version of the targeted business application based on BPMN 2.0 specification. Based on the functional requirements defined in phase B which are composed by user and service tasks, we generate an XML template.

5.6. *Phase D: Technology Architecture*

During the previous phase, technical and technological constraints are formalized. These latter are considered when matching the final SBBs, enriched in this phase with non-functional properties defining the Quality of Service needed from the existing services. This to consider what relevant resources are available in EAKR to ensure that the target system will meet the requirements and constraints. In the proposed

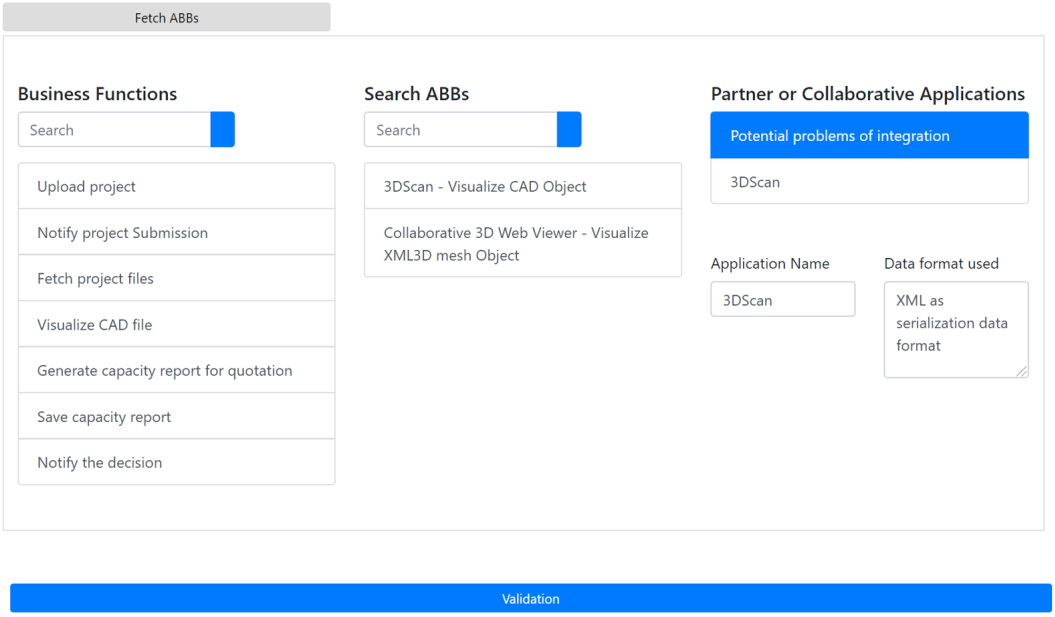

Figure 19. EACP Web Application - Phase C : Requirement Specification

use case scenario, we define an example for the consumption plan which is depicted in Figure 21 (NFR List). We set the average instance time metric as a non-functional requirement applicable for all the targeted services. After validation, SBBs are ranked based on the defined QOS threshold values defined in the consumption plan.

For each business function, we select the first SBB resulted from the ranking process as a building block to reuse for the implementation of the targeted business application. As a final result, we get a last version of an implemented BPMN with the related service endpoints of the solution building blocks.

6. Discussion

In this work, we propose an Enterprise Architecture Capability Profile specifically designed for service-oriented software enabling the qualification, the discovery, the reuse, and the sustainability for new business applications development. We demonstrate how the EACP Framework can assist developers or architects in the qualification process using the semantic Enterprise Architecture Knowledge Repository, based on a proposed meta-model inspired mainly from TOGAF and ISO 16100 Standard and formalized using semantic web techniques. This helps to offer a wider view qualification process that deals with the two perspectives of services which are the business perspective which brings value-in-use of the qualified feature for an organization that is interested into reuse, and the technical side along with a quality of service of the feature encapsulated by the software service. An exploitation methodology is defined in the proposed EACP Framework helping to overcome the use of ad-hoc methods to identify the most suitable components or artifacts to reuse. The proposed exploitation plan is designed based on the alignment of architecting actions with a requirement engineering process, and evolve together helping to investigate the highest functional compatibility of the desired functionalities and its related constraints.

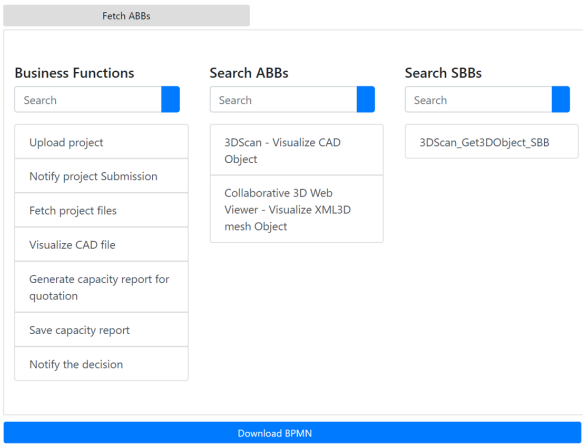

Figure 20. EACP Web Application - Phase C : Requirement Specification

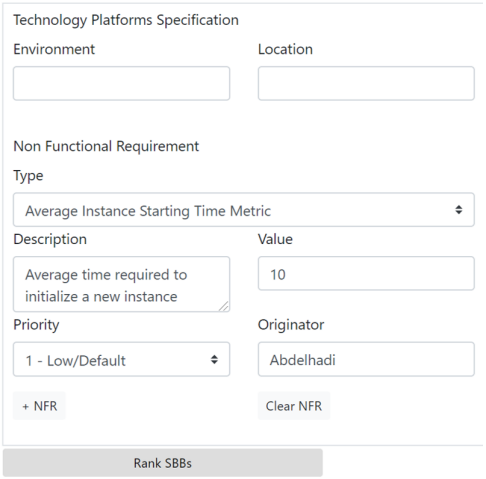

Figure 21. EACP Web Application - Phase D : Requirement Specification

The use of a semantic approach helped us in the formalization of the content of the proposed meta-model through the use of OWL language that explicitly specifies the meaning of EA meta-model entities and relations. Added to this, the proposed semantic model called EACP ontology supports as we demonstrate in our experimental work, advanced reasoning and querying requests on the EAKR that facilitates the comprehension and reuse of service-oriented software meta-data (capabilities, parameters, requirements, etc.). We note that the automatic content-based data management cannot be achieved in existing EA meta-models that do not use semantic markup. We adopted a semantic approach to describe EA information and we presented the different steps that we followed to design the EACP ontology.

Unlike existing EA ontologies mainly TOGAF first, the developed ontology is based on a foundational ontology namely BFO. Second, it is based on existing well-defined ontologies to cover all the meta-model components (functional, operational and technical). Third, the EACP ontology is modular, thus it can be easily extended by future users. Finally, the EACP ontology defines a data properties hierarchy to specify concrete values of concepts use cases.

The use of BFO has facilitated the integration of heterogeneous knowledge from different ontologies into a unique coherent semantic model. And the remove of concepts and relations ambiguity as for example the concept “TOGAF: core content“ by distinguishing between information data, qualities, functions, and processes. In most cases the alignment was not a trivial task for us for two main reasons: the complexity of the meta-model on one hand, and on another hand the ambiguity of TOGAF OWL concepts and object relations. The usage of an ontology for designing the model-based EAKR helps to achieve the targeted objectives and address the major challenges, namely create a software capability container that is modular, flexible, and extensible with a standard vocabulary to qualify existing solutions with semantic annotations. This helps either to capitalize on existing knowledge produced during the proposed engineering methodology and infer new knowledge to answer future business needs. One of the advantages of the proposed ontology is that it can serve alone without necessarily linking it to the proposed exploitation plan as a qualification and service description tool to describe the properties, design constraints, and capabilities of the service-oriented software.

As discussed in [99], architectural requirements can be significantly more important than their domain-specific equivalents. For instance, if we are designing a life-support machine, the availability or "up time" metric would be with a high importance. Regarding the proposed exploitation plan, it carries the validation of the requirements and drives the design of the foundations (ie, architecture) and the requirement definition of the business application we are building. This means at least, we offer the necessary structure for defining and validating architectural artifacts and requirement specifications, and at best, propose templates and artifacts of precedent projects or qualified solutions for recycling and reuse to meet the business need.

Regarding the exploitation process, as you may notice at run-time, the EACP Framework finds few results because no precedent project with its related requirements has been already introduced and capitalized. Also, it depends on the number of the qualified solutions and related services considered as architecture and solution building blocks in the EAKR. Continuous qualification is needed to maximize the exploitation and must be realized frequently to take full advantage of this proposed methodology.

Due to the modular, flexible and extensible properties of the proposed EACP-Ontology, several perspectives for the exploitation of the proposed EAKR are possible as for instance in case of no ABB or SBB is retrieved. Indeed, a connection with an external repository as "ProgrammableWeb.com" which has documented over 12.000 open web APIs and thousands of applications in its repository would be with a great value for companies. This helps to gather this open knowledge and exploit it in an efficient manner in the same way as we proposed in our exploitation plan.

The reusability potential of existing and proposed solutions is part of the technical and managerial contribution of software reuse to stimulate innovation and market penetration through shorter production cycles that promise strategic business advantages [100], in addition to several benefits that have been reported from successful reuse adoptions to reduce cost and accelerate development [101; 102], standardized architecture [101], and risk reduction [102] by means of known artefacts.

With a forward-looking perspective, next generation systems help the organizations to deliver outstanding value through a culture of innovation and speed [103]. Those systems are mass personalized, massively distributed, self-adapting system of systems, realized in an iterative and incremental development process to enable faster time to delivery and value realization [103]. This research work is a footstep for creating those next generation systems by enabling the discovery, design and delivery of new solutions that enables faster time to delivery and create alternatives.

7. Conclusion

In this work, we defined the EACP Framework for new business applications development based on the service reuse and orchestration. The Framework is based on a proposed meta-model enabling the design of an advanced software capability profile. The latter is designed based on an Enterprise Architecture Framework (TOGAF) and the best practices related to the implementation of ISO 16100 standard concepts. An enterprise architecture knowledge repository is proposed with a modular, flexible and extensible ontology (EACP-Onto). This resulted ontology is derived from the proposed models that offer a wider view qualification for service-oriented software covering business, technical and organizational aspects. Moreover, the proposed EACP Framework offers a methodology to guide the developers or architects during the qualification

process of existing solutions. An exploitation plan is proposed for designing new business application based on requirements engineering and software architecting actions to support requirement elicitation during the engineering life-cycle. Furthermore, the proposed approach helps to capitalize on artifacts produced during the engineering cycle of the projects and offers more visibility and sustainability to software for accelerating business application development. Finally, we validated the proposed approach with an implementation and an industrial use case.

Acknowledgment

This paper presents work developed in the scope of the project vf-OS. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 723710. The content of this paper does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in this paper lies entirely with the authors.

References

[1] G. Valenca, C. Alves, V. Alves, and N. Niu, "A systematic mapping study on business process variability," *International Journal of Computer Science & Information Technology*, vol. 5, no. 1, p. 1, 2013.

[2] G. S. Leal, W. Guédria, and H. Panetto, "Enterprise interoperability assessment: a requirements engineering approach," *International Journal of Computer Integrated Manufacturing*, vol. 33, no. 3, pp. 265–286, 2020.

[3] L. Da Xu, "Enterprise systems: state-of-the-art and future trends," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 630–640, 2011.

[4] M.-E. Iacob and H. Jonkers, "A model-driven perspective on the rule-based specification and analysis of service-based applications," *Enterprise information systems*, vol. 3, no. 3, pp. 279–298, 2009.

[5] S. Raemaekers, A. van Deursen, and J. Visser, "An analysis of dependence on third-party libraries in open source and proprietary systems," in *Sixth International Workshop on Software Quality and Maintainability, SQM*, vol. 12, pp. 64–67, 2012.

[6] L. Heinemann, F. Deissenboeck, M. Gleirscher, B. Hummel, and M. Irlbeck, "On the extent and nature of software reuse in open source java projects," in *International Conference on Software Reuse*, pp. 207–222, Springer, 2011.

[7] P. Smiari, S. Bibi, and D. Feitosa, "Examining the reuse potentials of iot application frameworks," *Journal of Systems and Software*, vol. 169, p. 110706, 2020.

[8] A. Mockus, "Large-scale code reuse in open source software," in *Emerging Trends in FLOSS Research and Development, 2007. FLOSS'07. First International Workshop on*, pp. 7–7, IEEE, 2007.

[9] M.-E. Paschali, A. Ampatzoglou, S. Bibi, A. Chatzigeorgiou, and I. Stamelos, "Reusability of open source software across domains: A case study," *Journal of Systems and Software*, vol. 134, pp. 211–227, 2017.

[10] G. Kakarontzas, P. Katsaros, and I. Stamelos, "Component certification as a prerequisite forwidespread oss reuse," *Electronic Communications of the EASST*, vol. 33, 2010.

[11] N. Niu, L. Da Xu, and Z. Bi, "Enterprise information systems architecture—analysis and evaluation," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2147–2154, 2013.

[12] D. Chen, G. Doumeingts, and F. Vernadat, "Architectures for enterprise integration and interoperability: Past, present and future," *Computers in industry*, vol. 59, no. 7, pp. 647–659, 2008.

[13] M. Snoeck, "Enterprise information systems engineering," *The MERODE Approach*, 2014.
[14] R. Perez-Castillo, F. Ruiz-Gonzalez, M. Genero, and M. Piattini, "A systematic mapping study on enterprise architecture mining," *Enterprise Information Systems*, vol. 13, no. 5, pp. 675–718, 2019.
[15] P. Saint-Louis and J. Lapalme, "An exploration of the many ways to approach the discipline of enterprise architecture," *International Journal of Engineering Business Management*, vol. 10, p. 1847979018807383, 2018.
[16] P. Saint-Louis, M. C. Morency, and J. Lapalme, "Examination of explicit definitions of enterprise architecture," *International Journal of Engineering Business Management*, vol. 11, p. 1847979019866337, 2019.
[17] B. Anthony Jnr, "Managing digital transformation of smart cities through enterprise architecture–a review and research agenda," *Enterprise Information Systems*, pp. 1–33, 2020.
[18] I. Arifin, "Design of architecture enterprise model information system academic and student administration bureau using togaf adm," 2019.
[19] Y. Putra and A. Hadiana, "Designing enterprise architecture for public health center based on togaf architecture development method," in *IOP Conference Series: Materials Science and Engineering*, vol. 879, p. 012163, IOP Publishing, 2020.
[20] R. Harrison, *TOGAF® 9 Certified Study Guide*. Van Haren, 2011.
[21] D. Quartel, M. W. Steen, and M. M. Lankhorst, "Application and project portfolio valuation using enterprise architecture and business requirements modelling," *Enterprise Information Systems*, vol. 6, no. 2, pp. 189–213, 2012.
[22] M. Matsuda, "Manufacturing software interoperability services which iso 16100 brings about," in *International IFIP Working Conference on Enterprise Interoperability*, pp. 60–70, Springer, 2012.
[23] A. Barros and D. Oberle, *Handbook of Service Description*. Springer US, 2012.
[24] S. Ghazouani and Y. Slimani, "A survey on cloud service description," *Journal of Network and Computer Applications*, vol. 91, pp. 61–74, 2017.
[25] H. Mezni, W. Chainbi, and K. Ghedira, "Aws-policy: an extension for autonomic web service description," *Procedia Computer Science*, vol. 10, pp. 915–920, 2012.
[26] Jonas, P. Brune, and H. Gewald, "A description and retrieval model for web services including extended semantic and commercial attributes," in *2014 IEEE 8th International Symposium on Service Oriented System Engineering*, pp. 258–265, IEEE, 2014.
[27] X. Zheng, Q. Wu, D. Ke, H. Li, and Y. Shi, "Social context enabled description model for web services," in *International Conference on Information Computing and Applications*, pp. 9–15, Springer, 2012.
[28] C. xiao Zhang, G. bing Zhang, S. xia Xing, and L. Xu, "An improved algorithm of fuzzy on the qos evaluation of web services based on owl-qos ontology," in *2013 International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013)*, Atlantis Press, 2013.
[29] M. Wei-bing, W. Wen-guang, Z. Yi-fan, and Y. Fa-yi, "Semantic web services description based on command and control interaction user context," in *2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference*, pp. 541–544, IEEE, 2014.
[30] D. Roman, J. Kopeckỳ, T. Vitvar, J. Domingue, and D. Fensel, "Wsmo-lite and hrests: Lightweight semantic annotations for web services and restful apis," *Journal of Web Semantics*, vol. 31, pp. 39–58, 2015.
[31] S. Chhun, N. Moalla, and Y. Ouzrout, "Qos ontology for service selection and reuse," *Journal of Intelligent Manufacturing*, vol. 27, no. 1, pp. 187–199, 2016.
[32] S. Ghazouani and Y. Slimani, "Towards a standardized cloud service description based on usdl," *Journal of Systems and Software*, vol. 132, pp. 1–20, 2017.
[33] C. Pedrinaci, J. Cardoso, and T. Leidig, "Linked usdl: a vocabulary for web-scale service trading," in *European Semantic Web Conference*, pp. 68–82, Springer, 2014.

[34] T. Narock, V. Yoon, and S. March, "A provenance-based approach to semantic web service description and discovery," *Decision Support Systems*, vol. 64, pp. 90–99, 2014.
[35] E. Khanfir, R. B. Djmeaa, and I. Amous, "Quality and context awareness intention web service ontology," in *2015 IEEE World Congress on Services*, pp. 121–125, IEEE, 2015.
[36] B. De, "Api management," in *API Management*, pp. 15–28, Springer, 2017.
[37] F. Khodadadi, A. V. Dastjerdi, and R. Buyya, "Simurgh: A framework for effective discovery, programming, and integration of services exposed in iot," in *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pp. 1–6, IEEE, 2015.
[38] H. Benfenatki, C. F. Da Silva, A.-N. Benharkat, P. Ghodous, and Z. Maamar, "Linked usdl extension for describing business services and users' requirements in a cloud context," *International Journal of Systems and Service-Oriented Engineering (IJSSOE)*, vol. 7, no. 3, pp. 15–31, 2017.
[39] K. Haniewicz, "Local controlled vocabulary for modern web service description," in *International Conference on Artificial Intelligence and Soft Computing*, pp. 639–646, Springer, 2012.
[40] R. Verborgh, T. Steiner, D. Van Deursen, J. De Roo, R. Van de Walle, and J. G. Vallés, "Capturing the functionality of web services with functional descriptions," *Multimedia tools and applications*, vol. 64, no. 2, pp. 365–387, 2013.
[41] B. C. Oliveira, A. Huf, I. L. Salvadori, and F. Siqueira, "Ontogenesis: an architecture for automatic semantic enhancement of data services," *International Journal of Web Information Systems*, vol. 15, no. 1, pp. 2–27, 2019.
[42] G. M. Kapitsaki, "Annotating web service sections with combined classification," in *2014 IEEE International Conference on Web Services*, pp. 622–629, IEEE, 2014.
[43] R. Alarcon, R. Saffie, N. Bravo, and J. Cabello, "Rest web service description for graph-based service discovery," in *International Conference on Web Engineering*, pp. 461–478, Springer, 2015.
[44] G. Beydoun, A. Hoffmann, R. V. Garcia, J. Shen, and A. Gill, "Towards an assessment framework of reuse: a knowledge-level analysis approach," *Complex & Intelligent Systems*, pp. 1–9, 2019.
[45] X. Xu, R. Liu, Z. Wang, Z. Tu, and H. Xu, "Re2sep: A two-phases pattern-based paradigm for software service engineering," in *2017 IEEE World Congress on Services (SERVICES)*, pp. 67–70, IEEE, 2017.
[46] H. Chen and K. He, "A method for service-oriented personalized requirements analysis," *Journal of Software Engineering and Applications*, vol. 4, no. 01, p. 59, 2011.
[47] K. Zachos, N. Maiden, X. Zhu, and S. Jones, "Discovering web services to specify more complete system requirements," in *International Conference on Advanced Information Systems Engineering*, pp. 142–157, Springer, 2007.
[48] B. Verlaine, I. Jureta, and S. Faulkner, "Towards conceptual foundations of requirements engineering for services," in *Proceedings of the Fifth IEEE International Conference on Research Challenges in Information Science (RCIS 2011), Gosier, Guadeloupe* (I. Computer, ed.), pp. 147–157, IEEE Computer society, 2011. Publication editors : IEEE Computer Society.
[49] I. J. Jureta, J. Mylopoulos, and S. Faulkner, "A core ontology for requirements," *Applied Ontology*, vol. 4, no. 3-4, pp. 169–244, 2009.
[50] D. Roman, U. Keller, H. Lausen, J. De Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web service modeling ontology," *Applied ontology*, vol. 1, no. 1, pp. 77–106, 2005.
[51] D. Quartel, W. Engelsman, H. Jonkers, and M. Van Sinderen, "A goal-oriented requirements modelling language for enterprise architecture," in *2009 IEEE International Enterprise Distributed Object Computing Conference*, pp. 3–13, IEEE, 2009.
[52] E. Niemi and S. Pekkola, "Using enterprise architecture artefacts in an organisation," *Enterprise information systems*, vol. 11, no. 3, pp. 313–338, 2017.
[53] P. Drews, I. Schirmer, B. Horlach, and C. Tekaat, "Bimodal enterprise architecture management: The emergence of a new eam function for a bizdevops-based fast it," in *2017*

IEEE 21st International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 57–64, IEEE, 2017.

[54] J. Yu, Q. Z. Sheng, J. Han, Y. Wu, and C. Liu, "A semantically enhanced service repository for user-centric service discovery and management," *Data & Knowledge Engineering*, vol. 72, pp. 202–218, 2012.

[55] C. E. Hog, R. B. Djemaa, and I. Amous, "Adaptable web service registry for publishing profile annotation description," in *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pp. 533–538, Dec 2013.

[56] H. Yoo, Y. Park, and T. Lee, "Ontology based keyword dictionary server for semantic service discovery," in *2013 IEEE Third International Conference on Consumer Electronics ¿ Berlin (ICCE-Berlin)*, pp. 295–298, Sep. 2013.

[57] K. Moradyan, O. Bushehrian, and R. Akbari, "A query ontology to facilitate web service discovery," in *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, pp. 202–206, Nov 2015.

[58] H. Seba, S. Lagraa, and H. Kheddouci, "Web service matchmaking by subgraph matching," in *Web Information Systems and Technologies* (J. Filipe and J. Cordeiro, eds.), (Berlin, Heidelberg), pp. 43–56, Springer Berlin Heidelberg, 2012.

[59] A. V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, and N. Adam, "Semantics-based automated service discovery," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 260–275, 2011.

[60] Y. Xue, C. Zhang, and Y. Ji, "Restful web service matching based on wadl," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 364–371, IEEE, 2015.

[61] M. Rathore and U. Suman, "An arsm approach using pcb-qos classification for web services: a multi-perspective view," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 165–171, IEEE, 2013.

[62] R. Li, K. He, and S. Wang, "An ontology-based process description and reasoning approach for service discovery," in *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pp. 320–325, IEEE, 2013.

[63] H. Becha and S. Sellami, "Prioritizing consumer-centric nfps in service selection," in *International Conference on Conceptual Modeling*, pp. 283–292, Springer, 2014.

[64] M. Á. Rodríguez-García, R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater, "Ontology-based annotation and retrieval of services in the cloud," *Knowledge-Based Systems*, vol. 56, pp. 15–25, 2014.

[65] N. N. Chiplunkar *et al.*, "Dynamic search and selection of web services," in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 1532–1536, IEEE, 2014.

[66] M. Matsuda, K. Kodama, S. Noguchi, S. Onishi, T. Asano, T. Horikita, and K. Komatsubara, "Configuration of a production control system through cooperation of software units using their capability profiles in the cloud environment," *Procedia CIRP*, vol. 17, pp. 416–421, 2014.

[67] Y. Elshater, K. Elgazzar, and P. Martin, "godiscovery: Web service discovery made efficient," in *2015 IEEE International Conference on Web Services*, pp. 711–716, IEEE, 2015.

[68] N. Boissel-Dallier, F. Benaben, J.-P. Lorré, and H. Pingaud, "Mediation information system engineering based on hybrid service composition mechanism," *Journal of Systems and Software*, vol. 108, pp. 39 – 59, 2015.

[69] L. Purohit and S. Kumar, "Web service selection using semantic matching," in *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, p. 16, ACM, 2016.

[70] F. Zeshan, R. Mohamad, M. N. Ahmad, S. A. Hussain, A. Ahmad, I. Raza, A. Mehmood, I. Ulhaq, A. Abdulgader, and I. Babar, "Ontology-based service discovery framework for dynamic environments," *IET Software*, vol. 11, no. 2, pp. 64–74, 2017.

[71] W. Mu, F. Benaben, and H. Pingaud, "An ontology-based collaborative business service selection: contributing to automatic building of collaborative business process," *Service Oriented Computing and Applications*, vol. 12, no. 1, pp. 59–72, 2018.
[72] K. Elgazzar, H. S. Hassanein, and P. Martin, "Daas: Cloud-based mobile web service discovery," *Pervasive and Mobile Computing*, vol. 13, pp. 67–84, 2014.
[73] A. Belfadel, J. Laval, C. B. Cherifi, and N. Moalla, "Toward service orchestration through software capability profile," in *Enterprise Interoperability VIII*, pp. 385–395, Springer, 2019.
[74] E. Al-Masri and Q. H. Mahmoud, "Qos-based discovery and ranking of web services," in *2007 16th international conference on computer communications and networks*, pp. 529–534, IEEE, 2007.
[75] A. Ouni, M. Kessentini, K. Inoue, and M. O. Cinnéide, "Search-based web service antipatterns detection," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 603–617, 2015.
[76] I. O. for Standardization, "Iso/iec 25010:2011 systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models," 2011.
[77] M. Patterns and P. Team, *Microsoft® Application Architecture Guide, 2nd Edition (Patterns and Practices)*. Microsoft Press, 2009.
[78] T. O. Group, *The Open Group Architecture Framework TOGAF^{TM} Version 9*. Basharat Hussain, 2009.
[79] C. Mateos, M. Crasso, J. M. Rodriguez, A. Zunino, and M. Campo, "Measuring the impact of the approach to migration in the quality of web service interfaces," *Enterprise Information Systems*, vol. 9, no. 1, pp. 58–85, 2015.
[80] F. Palma, N. Moha, G. Tremblay, and Y.-G. Guéhéneuc, "Specification and detection of soa antipatterns in web services," in *European Conference on Software Architecture*, pp. 58–73, Springer, 2014.
[81] A. Josey, M. Lankhorst, I. Band, H. Jonkers, and D. Quartel, "An introduction to the archimate® 3.0 specification," *White Paper from The Open Group*, 2016.
[82] J. M. França and M. S. Soares, "Soaqm: Quality model for soa applications based on iso 25010.," in *ICEIS (2)*, pp. 60–70, 2015.
[83] I. O. for Standardization, "Iso 16100-1:2009 industrial automation systems and integration – manufacturing software capability profiling for interoperability – part 1: Framework," 2009.
[84] S. Robertson and J. Robertson, *Mastering the requirements process: Getting requirements right (3rd Edition)*. Addison-wesley, 2012.
[85] M. Uschold and M. King, "Towards a methodology for building ontologies," 1995.
[86] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, *et al.*, "Bringing semantics to web services: The owl-s approach," in *International Workshop on Semantic Web Services and Web Process Composition*, pp. 26–42, Springer, 2004.
[87] A. Gerber, P. Kotzé, and A. Van der Merwe, "Towards the formalisation of the togaf content metamodel using ontologies," 2010.
[88] M. Rospocher, C. Ghidini, and L. Serafini, "An ontology for the business process modelling notation.," in *FOIS*, pp. 133–146, 2014.
[89] W. Ceusters, "An information artifact ontology perspective on data collections and associated representational artifacts.," in *MIE*, pp. 68–72, 2012.
[90] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology*. Mit Press, 2015.
[91] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with dolce," in *International Conference on Knowledge Engineering and Knowledge Management*, pp. 166–181, Springer, 2002.
[92] L. Peņicina, "Choosing a bpmn 2.0 compatible upper ontology," pp. 89–96, 2013.
[93] A. Czarnecki and C. Orłowski, "Ontology as a tool for the it management standards

support," in *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pp. 330–339, Springer, 2010.
[94] W. Chen, C. Hess, M. Langermeier, J. von Stülpnagel, and P. Diefenthaler, "Semantic enterprise architecture management.," in *ICEIS (3)*, pp. 318–325, 2013.
[95] O. Lassila, R. R. Swick, *et al.*, "Resource description framework (rdf) model and syntax specification," 1998.
[96] B. Green and S. Seshadri, *AngularJS.* " O'Reilly Media, Inc.", 2013.
[97] M. Cantelon, M. Harter, T. Holowaychuk, and N. Rajlich, "Node. js in action, greenwich, ct," 2013.
[98] A. Jena, "Fuseki: serving rdf data over http," 2014.
[99] P. Eeles, "Capturing architectural requirements," 2005. Accessed: 2020-06-10.
[100] V. M. Bauer, *Analysing and supporting software reuse in practice*. PhD thesis, Technische Universität München, 2016.
[101] O. P. N. Slyngstad, A. Gupta, R. Conradi, P. Mohagheghi, H. Rønneberg, and E. Landre, "An empirical study of developers views on software reuse in statoil asa," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, pp. 242–251, 2006.
[102] J. Varnell-Sarjeant, A. A. Andrews, and A. Stefik, "Comparing reuse strategies: An empirical evaluation of developer views," in *2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pp. 498–503, IEEE, 2014.
[103] A. Pendse and H. Amre, "Software 4.0: 'how'of building'next-gen'systems," in *Proceedings of the 11th Innovations in Software Engineering Conference*, pp. 1–2, 2018.

Glossary

ABB Architecture Building Block. 10–14, 19, 28, 29, 32
ADM Architecture Development Method. 8, 10, 15
API Application Programming Interface. 10–13

BFO Basic Formal Ontology. 23, 24, 31
BPMN Business Process Model Notation. 10, 15, 20, 22, 23, 29, 30

EACP Enterprise Architecture Capability Profile. 3, 8–10, 12–15, 23–26, 28, 30–32
EAKR Enterprise Architecture Knowledge Repository. 9, 10, 12, 14, 15, 17–20, 22, 25, 27–29, 31, 32

HTTP Hypertext Transfer Protocol. 11

IAO Information Artifact Ontology. 23
ICT Information and Communication Technology. 2

JSON JavaScript Object Notation. 11

MVC Model View Controller. 25

OWL-DL Ontology Web Language Description Logics. 23
OWL-S Web Ontology Language for Web Services. 5, 23

QOS Quality of Service. 4, 6, 9, 10, 12–14, 21, 22, 30

RDF Resource Description Framework. 24
REST Representational State Transfer. 11

SBB Solution Building Block. 10–15, 18, 20–22, 28–30, 32
SOA Service Oriented Architecture. 2

TOGAF The Open Group Architecture Framework. 4, 8–10, 12, 15, 23–25, 30

UDDI Universal Description Discovery and Integration. 6
UML Unified Modeling Language. 19, 28
URI Uniform Resource Identifier. 11, 14, 23

WSMO Web Service Modeling Ontology. 5, 6