# Requirements engineering and enterprise architecture-based software discovery and reuse

Abdelhadi Belfadel, Jannik Laval, Chantal Cherifi, Nejib Moalla

**Noname manuscript No.**
(will be inserted by the editor)

# Requirements Engineering and Enterprise Architecture-based Software Discovery and Reuse

**Abdelhadi Belfadel · Jannik Laval · Chantal Bonner Cherifi · Nejib Moalla**

Received: date / Accepted: date

**Abstract** Organizations' business processes need to be adapted in response to changing internal and external environments that are becoming increasingly complex. We target in this research work the exploitation of a software capability profile based on requirements engineering and enterprise architecture to respond to stakeholder requirements and efficiently reuse existing technical solutions. We provide an exploitation methodology based on the alignment of enterprise architecture actions with a requirement engineering process. These latter evolve together helping to investigate the highest compatibility of the desired functionalities and their related constraints. Our contribution aims to produce a ready-to-use application based on the defined requirements and the selected software capability profiles for accelerating business application development. Implementation and a case study are proposed to demonstrate the effectiveness of this approach.

**Keywords** Enterprise architecture · Capability profile · Requirements engineering · Service reuse · Software reuse

A. Belfadel
University Lumière Lyon 2, DISP Laboratory, Lyon, France
E-mail: Abdelhadi.Belfadel@gmail.com

J. Laval
University Lumière Lyon 2, DISP Laboratory, Lyon, France
E-mail: Jannik.Laval@univ-lyon2.fr

C. Bonner Cherifi
University Lumière Lyon 2, DISP Laboratory, Lyon, France
E-mail: Chantal.BonnerCherifi@univ-lyon2.fr

N. Moalla
University Lumière Lyon 2, DISP Laboratory, Lyon, France
E-mail: Nejib.Moalla@univ-lyon2.fr

## 1 Introduction

Modernizing or designing a new business process by reusing the functionality of existing software can be of great benefit to organizations. Leveraging previous developments and considering internal company solutions enriched with external ones, can help facilitate the development of complex systems at controlled costs while maintaining delivery times.

When developing software, the first thing to do is to understand and describe in a precise way the problem that the software must solve. Requirements for a targeted system describe what the system should do, what the services it should provide, and what quality or constraints it must have to make it attractive and acceptable to the owner [1]. These requirements reflect the needs of customers for a system. The process of analyzing, eliciting, and checking these services and constraints is called requirements engineering [2].

To maximize the reuse opportunities for companies, a component view with a concise evaluation model of software components (that describe in detail the capabilities of software) provides an overview of existing solutions and facilitates the discovery, selection, and decision to reuse [3]. This, combined with an organization of the different artifacts (an artifact is a more granular architectural work product that describes an architecture from a specific viewpoint. Examples include a class diagram, a server specification, a list of architectural requirements...) resulting from this evaluation model, aligned with a requirement engineering approach, aims to reduce the complexity when searching and selecting software components to reuse. In addition, focusing on service-oriented solutions, many opportunities for reuse of functionality will arise, resulting in more efficient use of existing resources.

To reduce the complexity of the description of software components that result from the evaluation model and to present the required detail of information at each level of its exploitation, enterprise architecture is of great value. Enterprise Architecture (EA) is the definition and representation of a high-level view of IT systems and enterprises' business processes. By considering an enterprise architecture-based approach, it is possible to organize the different artifacts in a way that enables an analysis of the reuse possibilities for an organization and ensure the feasibility of a targeted system or project. The insights or information provided by an enterprise architecture is needed, on the one hand, to determine, from a business perspective, the needs, and priorities for change [4] and, on the other hand, to organize the various components and technical artifacts and assess how an organization can exploit them.

With reference to this context, this research is an extended work of an already published work in [5]. The latter was focused on the design of a software capability profile implemented as a semantic model to gather description of the capabilities of existing solutions from several perspectives (organizational, business, technical and technological aspects). For the sake of readability, the proposed model and contributions published in [5] are presented in the following sections, as well as the new contributions related to the exploitation of the capability profiles through the alignment of a requirements engineering

approach and an enterprise architecture method. As discussed earlier, the objective of this research work is to leverage the model already published in [5] to address stakeholder requirements and efficiently reuse existing solutions, by investigating the highest functional and non-functional compatibility of existing software capability profiles with stakeholder's desired features to be implemented. The expected result from this work is an exploitation process of the software capability profiles, based on the Architecture Development Method from TOGAF [6], aligned to a requirements engineering approach implemented using Volere Specification [1]. However, the problem we faced is how to align requirements and architecture artifacts in an engineering cycle, to help in the refinement of requirements and select the best candidate components to serve as building blocks in a new system?

To respond to this 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 and present first the Enterprise Architecture Capability Profile (EACP) in section 3. Section 4 presents a concrete use case scenario on which this approach has been applied, and that serves as an example throughout the description of the exploitation process in section 5. Section 6 presents an implementation of the proposed approach. Section 7 discusses our work and finally a conclusion is drawn in Section 8.

## 2 Related Work

### 2.1 Requirements Engineering in Software Development Process and Service Reuse

Classical techniques for software solution specification are structured analysis and object-oriented techniques [7]. The view of requirement engineering as solution specification is taken by the IEEE 830 standard [8] and by other authors on requirements [1] [9]. In this view, and as mentioned by [7], a requirements specification consists of a specification of the context where the system operates, desired system functions, the semantic definition of these functions, and quality attributes of the functions.

Several research works and methods ([10], [11] or [12]) exist in the literature to enhance software requirement specifications and for feature selection. [1] propose Volere as a basis for a requirement specification. It is a result of many years of practice, consulting, and research in requirement engineering and business analysis. Volere provides template sections for each of the requirement types appropriate to today's software systems. [13] propose a paradigm for software service engineering to reuse services for developing new applications more rapidly with the aim to satisfy 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 implementation of the approach are proposed. [14] 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. [15] propose a service selection algorithm based on textual requirements 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 the UDDI registry. There is additional ontology-based research work such as [16], where the authors took the CORE Ontology (for Core Ontology for Requirements) [17] for requirement elicitation, and established a relationship with the concepts of Web Service Modeling Ontology (WSMO) [18].

### 2.2 Knowledge Management and Service Repositories for Service Reuse

Research on repositories for effective and useful management and discovery of services for service-oriented paradigm has recently earned significant impulse. Some specifications as UDDI [19] or ebXML Registry [20] has provided primary support to register, discover and integrate services. Due to limited capabilities offered by existing registry specifications for services discovery, some research works in the literature aim at improving service repositories with ontology-based discovery facilities. Later, semantic models have been proposed to enrich the service registry with semantic annotations, combined with matchmaking algorithms to match service capabilities.

Based on the systematic analysis of relevant research works regarding service discovery with consideration of our needs, Table 1 classifies the related service registry and discovery works published between 2002 and 2019 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 requirements, interoperability requirements, and technology constraints. C4) Technology level: exploitation based on the identification of the platforms and infrastructure. C5) Non-functional properties (QoS, Security...). C6) Exploitation based on a Requirements Engineering Process (it involves all the mentioned levels)

Out of Table 1, 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. Software architecture helps to manage the

complexity of software by providing an abstraction of the system. The requirements 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 1) 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.

| **Service Repository and Discovery** | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** |
|---|---|---|---|---|---|---|
| [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32] [33] | | + | | | + | |
| [15] | | + | | | + | + |
| [34] | | + | | | | |
| [35], [36], [37] | | + | | | | |
| [38], [39], [40] | | | | | + | (+) |
| [41], [42], [43] | | + | | | | |
| [44], [45] | | + | | | | (+) |
| [46] | | + | + | | | |
| [47] | | + | | + | | |
| [48] | | + | | | | (+) |
| [49] | + | + | | | + | |
| [50] | | + | | | + | (+) |
| [51] | | + | | | | |
| [52], [53] | | + | | + | + | |
| [54] | + | + | | | | (+) |

**Table 1** Service repository and discovery for reuse

## 2.3 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 this software are lacking wider view qualification taking into consideration the business, operational and technical views of the software and their related services [5]. In addition, no solution has been provided to fit the requirements engineering along with the impact of architecture on requirements when dealing with the identification of the most suitable components and avoiding the misevaluation during the selection phase. Therefore, enhancing 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 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 in 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 that includes requirements engineering and enterprise architecture for formalizing the requirements.

## 3 Qualification Model - Software Capability Profile

To bring the value-in-use of existing software and facilitate the discovery and reuse, we present in this section the meta-model of the Software Capability Profile (see Figure 1) and its related EACP Ontology that is presented in detail in [5] and depicted in Figure 2. The meta-model is inspired mainly from TOGAF [6], ISO 16100 [55], Microsoft Application Architecture Guide [56] and ISO 25010 [57]. It aims to gather functional and non-functional specifications; the organizational impact of an organizations' software; and it links the business services to their related physical components to offer a wider view qualification and improve the reuse when developing a new business application. The proposed meta-model is composed of 6 packages. i) Organization package: Composed by the organizational unit, with its related business goals and objectives that guided the development of existing software. ii) Architecture Building Blocks package: This entity is constructed according to the life-cycle creation of Architecture Building Blocks (ABBs) based on the ADM Method. An ABB describes the business problem for which this component was developed for, its implementation specification, standards used, the stakeholders concerned. It provides other details such as the operational vision of the component, the definition of the business function of the ABB, its attributes and constraints, data and application interoperability requirements and design-time quality attributes. iii) Solution Building Blocks package: SBBs represent the physical equivalent of ABBs and describe the components 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 and transversal quality attributes which might be updated according to the defined frequency for each attribute. iv) Application package: Describes the technical requirements of the service-oriented software in general with its exposed components (for instance REST services). It also describes the execution environments on which the application is running. v) Business Process

package: This package is used in the exploitation phase and represents the "to-be" business application to realize. vi) 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 life-cycle. 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.

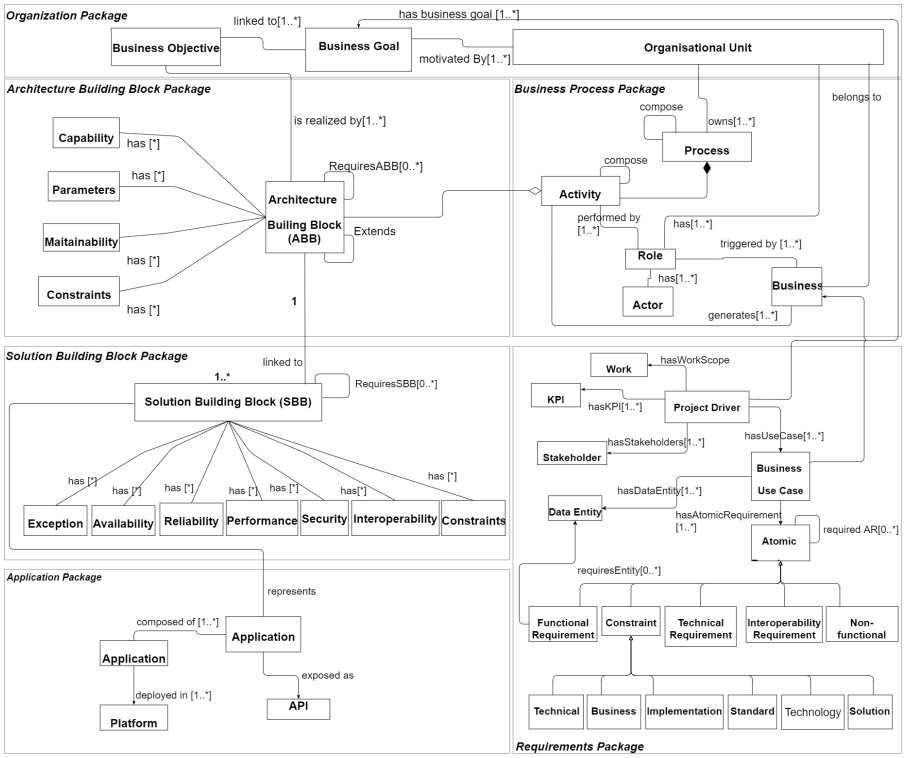



**Fig. 1** Proposed meta-model

The resulting EACP profile instances are saved in a semantic repository called in this context the Enterprise Architecture Knowledge Repository (EAKR repository) as an ontology instance. Regarding the design effort of the EACP Ontology, we identified from state-of-the-art solutions some existing ontologies to reuse. We have selected those that cover some of our needs and that are well-defined, consistent, and reused in other projects. We selected Basic Formal Ontology (BFO) which is a top-level ontology and four domain ontologies, namely Ontology Web Language for Services (OWL-S) [58], The Open Group Architecture Framework ontology (TOGAF-Ontology) [59], BPMN 2.0 Ontology [60] and Information Artifact Ontology (IAO) [61]. Third, we managed the selected foundational and domain ontologies, by integrating and extending in

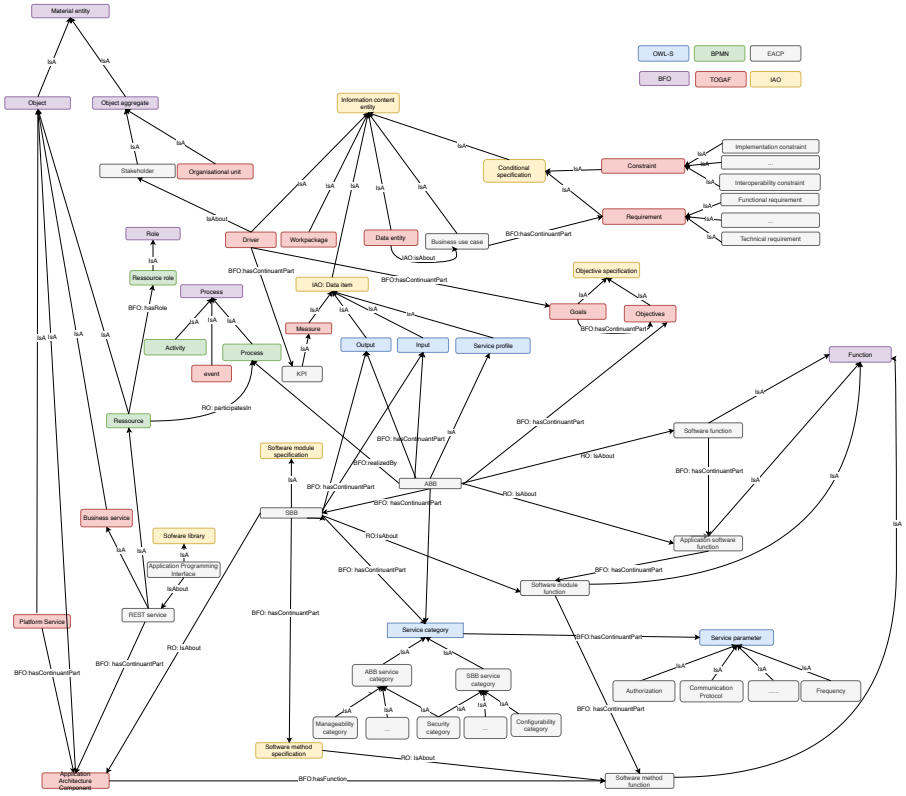



**Fig. 2** Basic pattern of the main classes and relationships in the EACP ontology

a coherent way the different ontologies into the targeted EACP ontology using the Protégé Ontology Editor. Finally, we evaluated the consistency of the resulted model using the Fact++ reasoner. An extract of the related Ontology to EAKR repository is presented in Figure 2.

The scope of this paper does not concern the presentation of the qualification model and the design of its ontology mentioned above and presented in [5], however, the elements presented above are enough for the understanding of the exploitation phase that will follow.

## 4 Use Case Scenario

This section presents the use case that will serve as an example for the description of the following exploitation process. This use case scenario involves 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

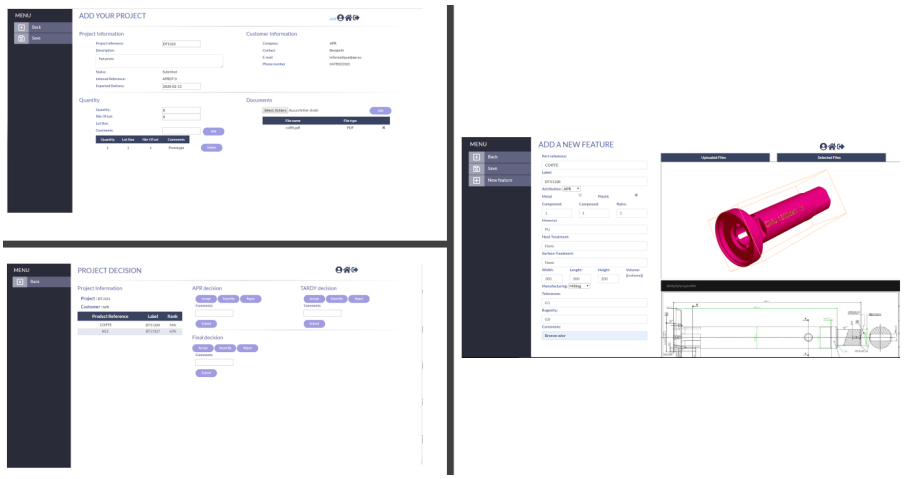



**Fig. 3** Screenshots of the targeted business application

most appropriate business collaboration opportunities among common customer projects. Starting from a Computer-Aided Design (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 return on investment. 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 discover existing technical services that allow reducing 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 an example of the inputs needed in each phase of the exploitation process. A screenshot of the developed prototype is presented in Figure 3.

## 5 Exploitation Process

To design a new business application reusing technical services from the EAKR, the architect or developer goes through a requirement elicitation process. This

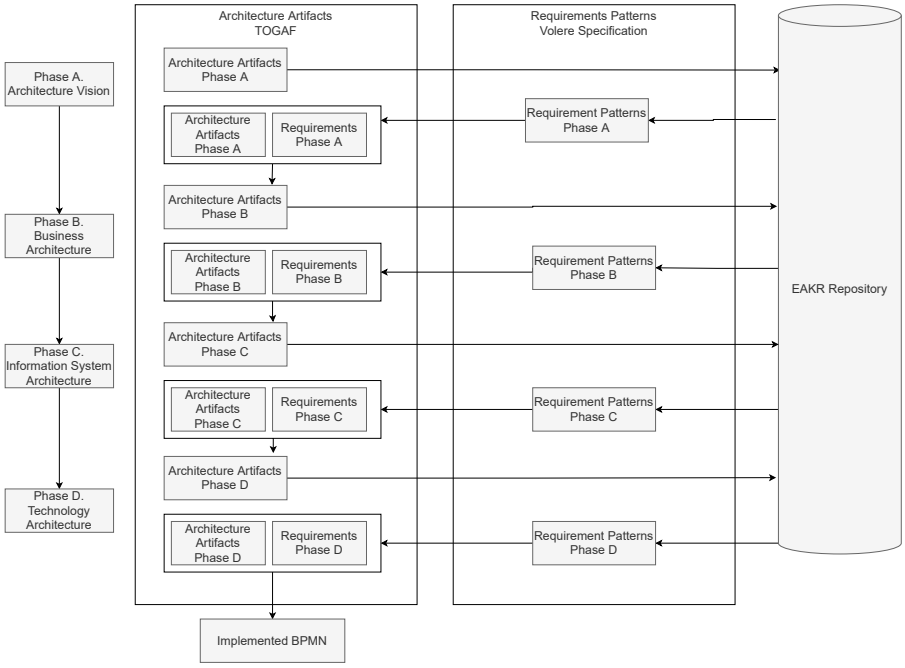



**Fig. 4** Exploitation Plan

process depicted in Figure 4 is structured in several phases starting with the architectural vision, going through the business architecture, data, application, and technology architecture phase, leading to the generation of an implemented BPMN which uses the qualified services if a match is confirmed. In the following sub-sections, we describe the actions to realize in each phase, enriched with a concrete example from the use case scenario to strengthen the understanding. Templates in JSON format are also proposed to formalize the architecture artifacts and requirement specifications in each phase.

## 5.1 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 mainly focused on gathering the business goals and related objectives of the targeted project. Other architecture artifacts are used to structure the project drivers, such as the identification of stakeholders, definition of the organizational model of the company, and KPIs enabling to evaluate the targeted business application. Based on the architectural artifacts of phase A (see Figure 5 for the exhaustive list - column ADM artifacts), we fetch requirement patterns (see Figure 5 for the exhaustive list from Volere specification - column Requirements) produced during previous projects to guide and support the developer for the

| Phase | ADM's artifacts | Requirement Specification |
|---|---|---|
| A - Architecture Vision | • Business goals<br>• Business objectives<br>• Organizational Model for Enterprise Architecture (to identify the organization units that will be affected by the architectural changes)<br>• Identify stakeholders with their concerns<br>• Defines Key Performance Indicators (KPIs) | • Project Drivers<br>○ The Purpose of the Project<br>▪ The User Business or Background of the Project Effort<br>▪ Goals of the Project<br>○ The Client, the Customer, and Other Stakeholders<br>▪ The Client<br>▪ The Customer<br>▪ Other Stakeholders (roles or types, names…)<br>○ Users of the Product<br>▪ The Hands-On Users of the Product (name/category, role, experience, technological experience…)<br>▪ Priorities Assigned to Users Key users, secondary users, unimportant users (to prioritize their requirements) |

**Fig. 5** Phase A: alignment of architecture and requirements artifacts

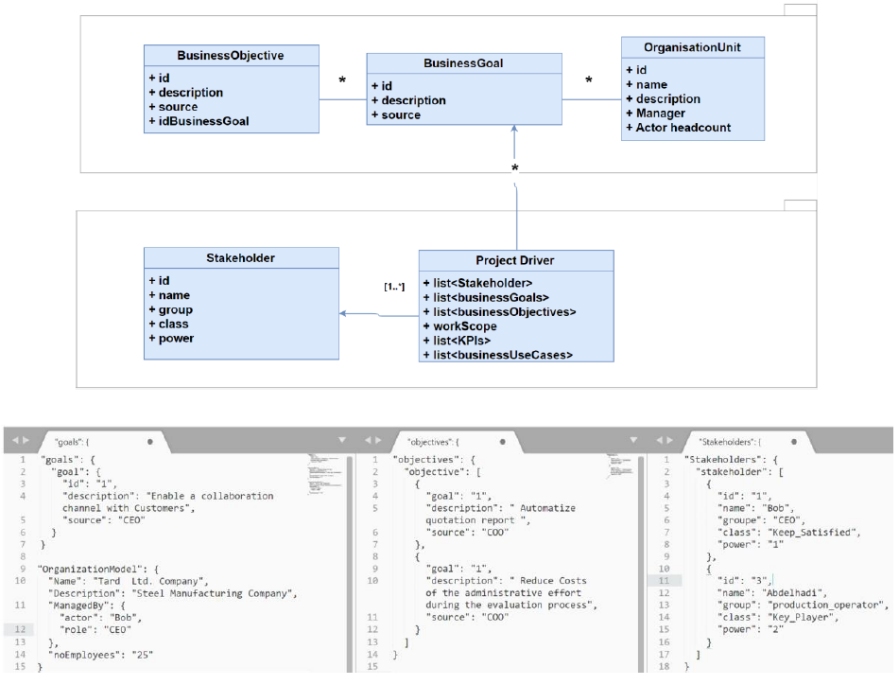



**Fig. 6** Example of architecture artifacts managed during phase A and related model

upcoming requirement definition. If a template is found, it is presented to the developer. In addition to the architecture artifacts, the developer formalizes his requirements by defining the project drivers such as the business actors, the client, and the 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 as requirements templates in the future exploitation process.

| Phase | ADM's artifacts | Requirement Specification |
|---|---|---|
| **B - Business Architecture** | • Architecture Definition Document<br>  ○ Business Architecture Components<br>  ▪ Define viewpoint for each stakeholder<br>  ▪ Define Actor/Role matrix<br>  ▪ Business Modeling and Activity Model: as BPMN or use-case diagram<br>  ▪ Identify which component is a function or service. Specify the SLAs for the services<br>  ○ Gap Analysis<br>  ▪ Identify building blocks to carry over to the target;<br>  ▪ Identify new required building blocks<br>  ○ Architecture Requirements Specification<br>  ▪ Implementation specifications<br>  ▪ Implementation standards<br>  ▪ Solution Constraints | • Project Constraints<br>  ○ Mandated Constraints<br>  ▪ Solution Constraints (design preferences, or only certain solutions may be acceptable)<br>  ▪ Functional Requirements<br>• The Scope of the Work<br>  ○ The Current Situation<br>  ○ Work Partitioning (A list showing all business events to which the work responds.)<br>• The Scope of the Product<br>  ○ Product Use Case (A use case diagram.)<br>• Functional and Data Requirements |

**Fig. 7** Phase B: alignment of architecture and requirements artifacts

Figure 6 illustrates an example from the use case with some proposed architecture artifacts and requirements 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 are needed to build the product.

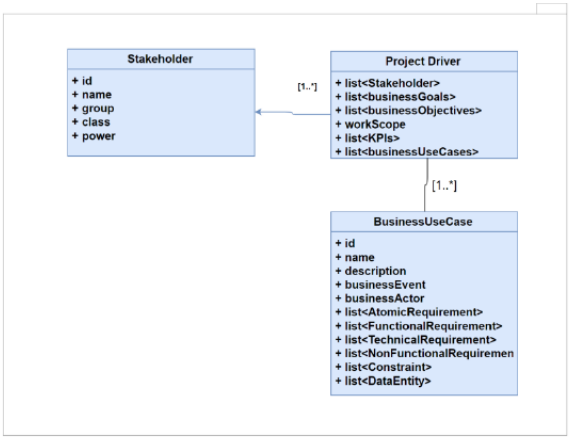
## 5.2 Phase B: Business Architecture

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. This phase focuses on the business side and supports the developer or architect to prepare all required inputs which are presented in Figure 7.

The most important architectural artifact in this phase is the high-level business scenario. This is designed using BPMN which is a standard language for business process modeling. This first high-level modeling is designed using ArchiMate [1] which is recognized as a standard for EA modeling by the Open Group [62] and that supports business process modeling. This high-level modeling helps to define the business-entity relationship to know which entities are needed for every business action or behavior.

This business model is enriched with other architectural artifacts (see Figure 7 for the exhaustive list - column ADM 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 Phase B outputs. It requires the definition of the implementation specifications to guide the development work, implementation standards in case the implementation

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




```
"businessActors": {
  "businessActor": {
    "name": "Abdelhadi",
    "description": "production_process_operator"
    "actorTask": "production_operator",
    "roles": "production_operator"
  }
}

"constraints": {
  "solutionConstraint": {
    "description": " Use internal LxP ERP
    database for production capacity evaluation",
    "severity": "1",
    "owner": "COO"
  }
}
```

```
"businessUseCases": {
    "businessUseCase": {
        id: 1,
        name: "upload project",
        description: "project details and
        documents using a form"
    },
    "businessUseCase": {
        id: 2,
        name: "Capacity report",
        description: "start the evaluation of
        the project"
    }
}
```

**Fig. 8** Example of architecture artifacts managed during phase B and related model

should follow some specific standard. Figure 8 depicts the model of some architecture artifacts managed during this phase and shows an example of the proposed template for some of these architecture artifacts.

Once these elements are defined, a request is sent to the EAKR to fetch requirement templates of phase B produced during the last projects to be reused. The aim is to guide the developer to define the project constraints and the functional requirements as depicted in Figure 7 and inspired from Volere specification (column Requirement). The resulted templates from the request (if any) present the scope of previous projects sharing the same context, the existing 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. Figure 9 depicts the model of the different artifacts that may result from the EAKR and shows an example of the proposed template for some of these artifacts.

These resulting templates are presented to the developer to guide him during this phase B for the consolidation and refinement of his requirements. It helps to offer support for defining and consolidating the use-cases and related

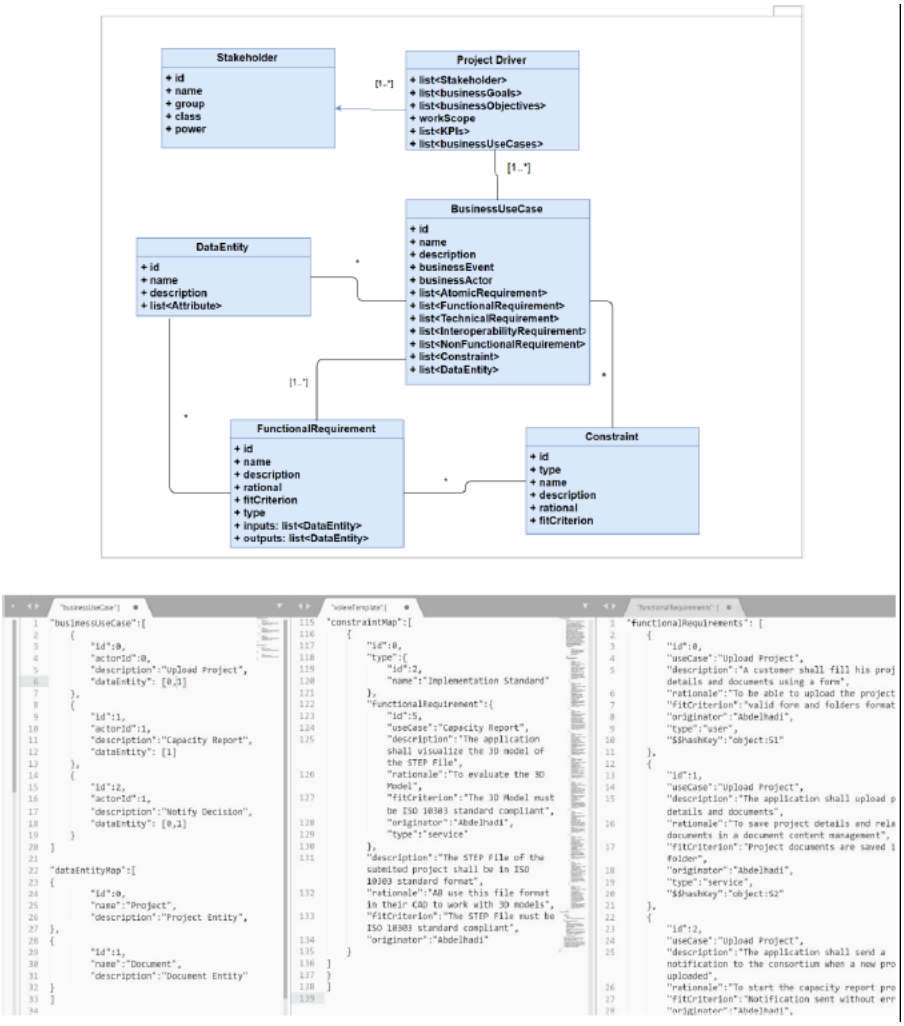



**Fig. 9** Example of requirement artifacts managed during phase B and related model

functional specifications. The proposed template is inspired by the Atomic Requirement Template which is proposed by [1] and depicted in Figure 10. This phase B ends with well formalized, testable, and categorized as user or service task functional requirements.

## 5.3 Phase C: Information Systems Architecture - Data Architecture

The objective of Phase C is to develop the targeted information system architecture. It involves a combination of data and application architecture. There-


**Fig. 10** Phase B: requirements template selection

fore, this phase is composed of two sub-phases, the Data and Application Architecture.

The Data architecture phase enhances the definition of the relationship between data entities and targeted business functions previously defined in phase B. We have already depicted in Figure 9 the models that enable to link data entities associated with each business function. Then, the needed action in this phase is to define the properties of each business entity involved in the business functions using relevant data models such as the Class Diagram in the Unified Modeling Language (UML). To this end, we propose templates based on the proposed models for the definition and formalization of the data entities involved in a business function. An example related to the use case is depicted in Figure 12. The left side of the figure depicts the definition of a class model, and the right-side links an entity with its attributes to a specific business function where the entity is used.

Additional architecture requirements specifications are as well formalized such as Data Interoperability or Technology Architecture Constraints (see Figure 11 for the exhaustive list - column ADM artifacts). 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

| Phase | ADM's artifacts | Requirement Specification |
|---|---|---|
| **Phase C – Information Systems - Data Architecture** | • Architecture Definition Document<br>  ○ Target Data Architecture<br>  ▪ Models for business data, logical data (existing relevant data models, such as the ARTS and POSC models.) corresponding to the selected viewpoints<br>  ▪ Data Entity/Business Function matrix<br>  ○ Architecture Requirements Specification<br>  ▪ Data interoperability requirement (e.g., XML schema, security policies)<br>  ▪ Constraints on the Technology Architecture about to be designed | • Project Constraints<br>  ○ Mandated Constraints<br>  ▪ Partner or Collaborative Applications (highlight potential problems of integration)<br>• Functional Requirements<br>  ○ Functional and Data Requirements<br>  ▪ Data Requirements (in form of class diagram) |

**Fig. 11** Phase C - Data Architecture: alignment of architecture and requirement artifacts

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, we fetch and map in the EAKR the ABBs and business functions using the defined data entities, and which are compliant with the constraints if defined (see Figure 13-left column for an ABB template example). The related SBBs and their corresponding technical components are gathered to highlight potential problems of integration and are presented in the proposed template depicted in Figure 13-right column. The related models of ABB, SBB and application package have already been described in section 3 and in [5].


**Fig. 12** Data Architecture artifacts example for Phase C

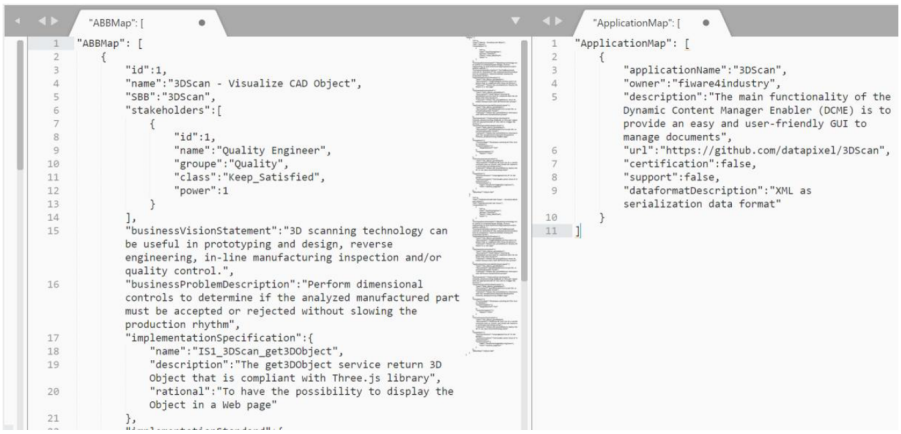



**Fig. 13** Requirement: Partner or Collaborative Applications (to highlight potential problems of integration)

## 5.4 Phase C: Information Systems Architecture - Application Architecture

This phase deals with the application architecture artifacts and the corresponding requirements. (see Figure 14 for the exhaustive list of the artifacts). The developer or architect is guided to define the technical requirements in the same template as for the atomic requirement (see Figure 15- left side for an example). Technology or infrastructure constraints and application interoperability requirements are either defined in this phase. Those constraints are added to previous ones to fetch for SBBs and related applications in the EAKR. The resulting SBBs and related application (middle and right side of Figure 15) offer the first overview of existing applications and related technical services (in the case of service-oriented solutions) 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 generated. The user and service tasks are generated and a link between service tasks and a set of existing services is performed based on the elements defined during previous phases (related to selected SBBs). This first solution reflects the prototype to realize, aiming to resolve and meet the business need expressed during this elicitation process.

## 5.5 Phase D: Technology Architecture

The last phase D is about the technology architecture artifacts (see Figure 16 for the exhaustive list of the 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 the EAKR repository to ensure that the target system will meet some

| Phase | ADM's artifacts | Requirement Specification |
|---|---|---|
| **Phase C - Information Systems - Application Architecture** | • List of applications or application components that are required<br>• Develop matrices across the architecture by relating applications to business services, business functions, data, processes<br>  ○ Role/Application matrix<br>  ○ Application/Function matrix<br>  ○ Data Entity/Business Function matrix<br>• Architecture Definition Document<br>  ○ Applications interoperability requirements<br>  ○ Constraints on the Technology Architecture about to be designed<br>  ○ Relevant technical requirements | • Project Constraints<br>  ○ Mandated Constraints<br>  ▪ Off-the-Shelf Software (OTS) that must be used to implement some of the requirements for the product |

**Fig. 14** Phase C - Application Architecture: alignment of architecture and requirement artifacts


**Fig. 15** Application architecture artifacts for Phase C

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 defining the Quality of Service needed from the existing services.

The model of non-functional requirement is based on the atomic requirement model. An example of the definition of this non-functional requirement is depicted in Figure 17. The resulted SBBs, if a match is confirmed, 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. The resulted SBBs are ranked as already defined by [63] for QoS ranking, reflecting the non-functional specifications before selecting the final SBB and generating an implemented business process. The result of this rank-

| Phase | ADM's artifacts | Requirement Specification |
|---|---|---|
| **Phase D - Information Systems - Technology Architecture** | • Architecture Definition Document<br>  ○ Technology platforms<br>  ○ Environments and Locations diagram<br>  ○ Application/Technology matrix | • Project Constraints<br>  ○ Mandated Constraints<br>  ▪ Implementation Environment of the System<br>• Off-the-Shelf Solutions<br>  ○ Ready-Made Products<br>  ○ Reusable Components<br>  ○ Products That Can Be Copied<br>• Non-functional Requirements<br>  ○ Performance Requirements<br>  ▪ Speed and Latency Requirements<br>  ▪ Precision or Accuracy Requirements<br>  ▪ Reliability and Availability Requirements<br>  ▪ Robustness or Fault-Tolerance Requirements<br>  ▪ Capacity Requirements<br>  ▪ Scalability or Extensibility Requirements<br>  ○ Operational and Environmental Requirements<br>  ▪ Requirements for Interfacing with Adjacent Systems (interface with partner applications and/or devices)<br>  ○ Security Requirements<br>  ▪ Access Requirements<br>  ▪ Integrity Requirements<br>  ▪ Privacy Requirements |

**Fig. 16** Phase D: Alignment of Architecture and Requirements artifacts


**Fig. 17** Phase D: Technology architecture artifacts template example

ing 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 (see Figure 18 for an example). In the next section, we present the implementation of a prototype of this exploitation process developed as a Web Application.

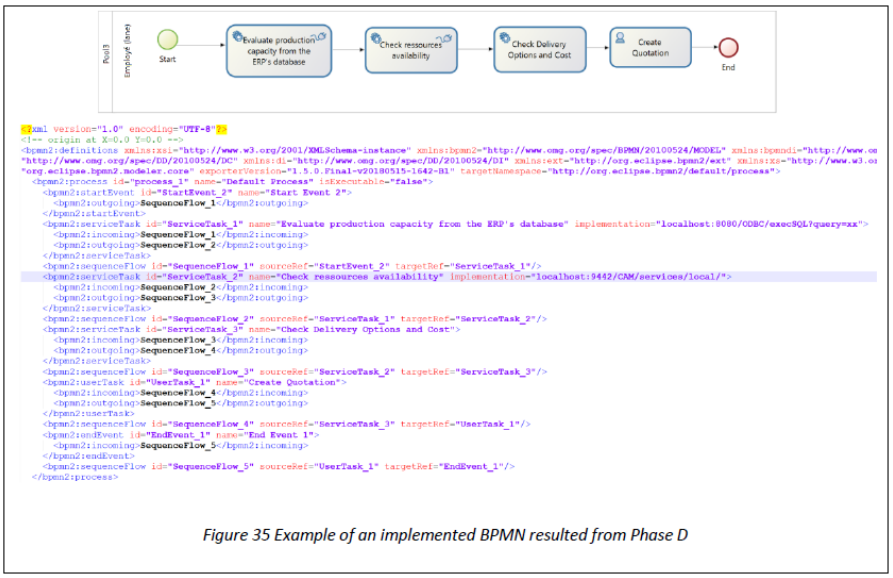



**Fig. 18** Example of an implemented BPMN resulted from Phase D

## 6 Framework Implementation

We implemented the exploitation process as a Web Application as depicted in Figure 19. The source code is available at Github repository [2]. The video of this technical presentation is available here [3] and the of the entire use case is available here [4]. We selected AngularJS [64] as a Web Framework that enables the development of single-page applications following the MVC (Model-View-Controller) pattern for the front-end environment, and NodeJS Framework [65] which is a popular platform for building server-side Web Applications written in Javascript. Regarding the EAKR repository, we deployed the EACP Ontology along with example of qualified open-source solutions from vf-OS [5] and FITMAN project [6] in Apache Jena Fuseki [66] (see Figure 20).

In the following subsections, we describe how a developer or architect can interact with this application in each phase, what inputs are required (based on the use case presented earlier in this paper), how information is presented, and how validation occurs.

Note that in the case where no artifact was found in the repository with a perfect match, we used string similarity based on Dice's coefficient. Several open-source JavaScript packages exist. We selected the string-similarity pack-

[2] https://github.com/AbdBelf/EacpFramework
[3] https://bul.univ-lyon2.fr/index.php/s/xsAMwEoYIbRYbLh
[4] https://bul.univ-lyon2.fr/index.php/s/EfoSLyZwkHYbT9t
[5] www.vf-OS.eu
[6] http://www.fiware4industry.com

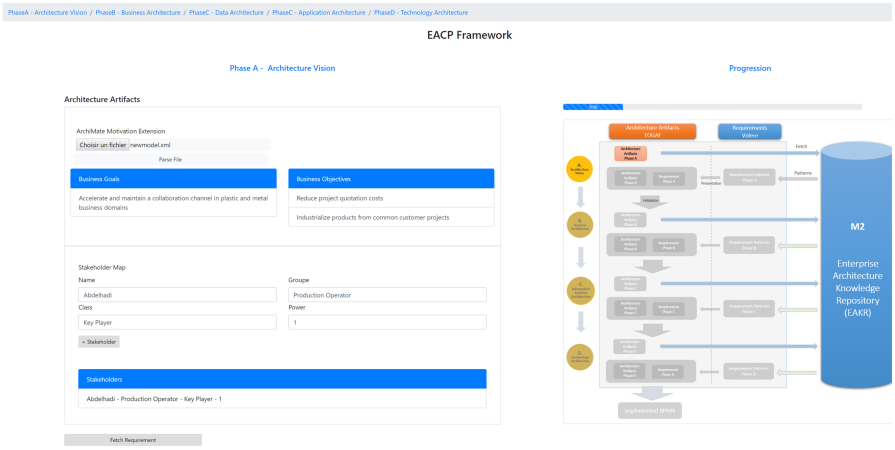



**Fig. 19** EACP Web Application - Phase A : Architecture Artifacts

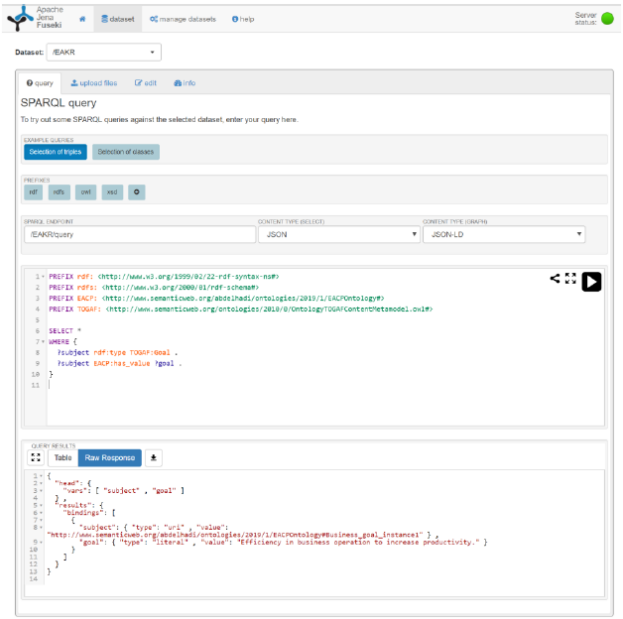



**Fig. 20** EAKR based on the EACP Ontology and Apache Jena SPARQL Endpoint

age that is publicly offered in GitHub repository [7]. We fixed the threshold to 90%, which could be modified to get flexible results.

[7] https://www.npmjs.com/package/string-similarity

### 6.1 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 19 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 19 depicts phase A of the Web application. The first column is the architecture and requirements elicitation process that guides the developer to consolidate and validate their requirements 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 is 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 an example of a SPARQL request in Figure 20) 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 19, 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 on 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.

### 6.2 Phase B: Business Architecture

Based on the 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 process depicted in section 5.

The requirements specification of phase B deals with the functional requirements and constraints of the project. 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

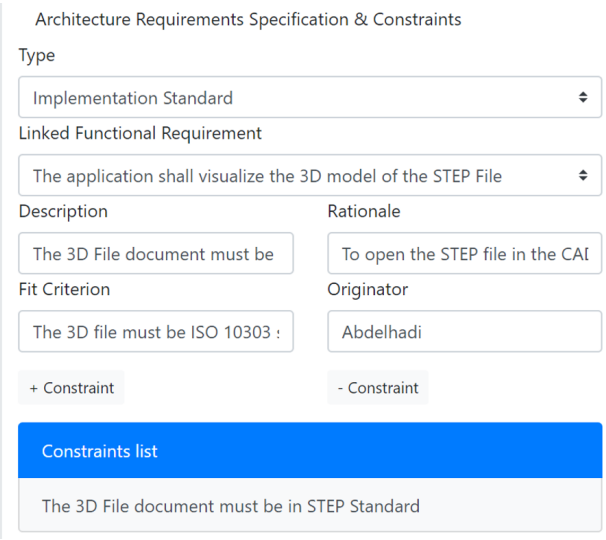



**Fig. 21** EACP Web Application - Phase B : Requirement Specification

business scenario 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). 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.

In this phase, there is a possibility to enrich the functional specifications by adding required constraints or architecture requirements specifications 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 21.

### 6.3 Phase C: Data and Application Architecture

The Data architecture phase enhances the definition of the relationship 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 Unified Modeling Language (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 section 5.3. This latter matches the defined functional requirements with business functions defined in the ABB model. ABBs that

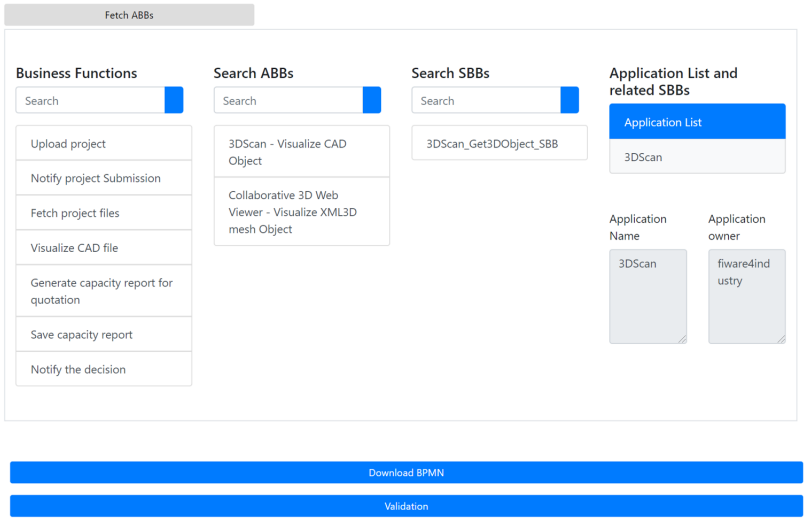



**Fig. 22** EACP Web Application - Phase C : Requirement Specification

correspond 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. Regarding the Application Architecture phase, 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 section 5. 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 22.

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 (see Figure 18).

### 6.4 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

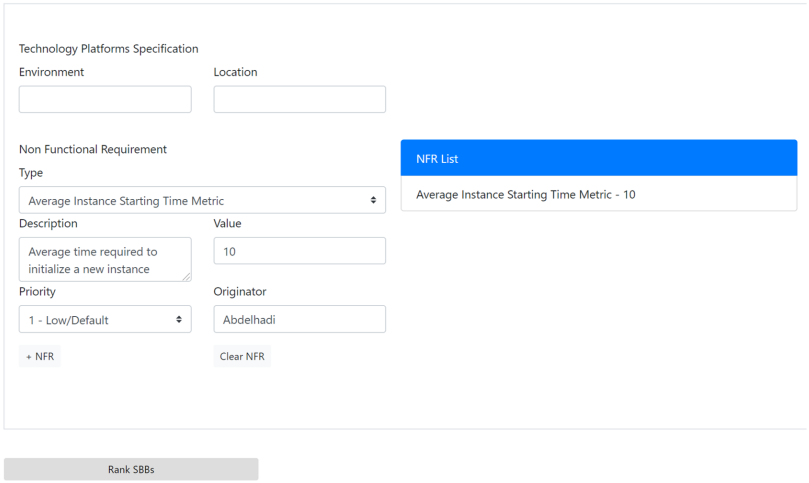



**Fig. 23** EACP Web Application - Phase D : Requirement Specification

needed from the existing services. This to consider what relevant resources are available in the EAKR repository to ensure that the target system will meet the requirements and constraints. In the proposed use case scenario, we define an example for the QoS which is depicted in Figure 23 (NFR List). We set the average instance time metric as a non-functional requirement applicable for all the targeted technical services. After validation, SBBs are ranked based on the defined QoS threshold values.

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.

## 7 Discussion

In this work, we propose an Enterprise Architecture Capability Profile specifically designed for service-oriented software enabling the qualification, the discovery, reuse, and sustainability for new business applications development. We demonstrate how the proposed approach 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 in reuse, and the technical side along with a quality of service of the feature encapsulated by the software service. An exploitation methodology is defined to overcome the use of ad-hoc methods to identify the most suitable compo-

nents or artifacts to reuse. The proposed solution 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 their related constraints.

As discussed in [67], on some projects, architectural requirements can be significantly more important than their domain-specific equivalents (as for instance, if we are designing a business application with a specific high availability as implementation constraint, the "up-time" metric would be with a high importance). Regarding the proposed exploitation methodology, it carries the validation of the requirements and drives the design of the foundations (i.e., 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 previous 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 process finds few results because no previous project with its related requirements has been already introduced and capitalized. Also, it depends on the number of qualified solutions and related services considered as architecture and solution building blocks in the EAKR Repository. Continuous qualification is needed to maximize the exploitation and must be realized frequently to take full advantage of this proposed methodology.

## 8 Conclusion

In this work, we defined the Enterprise Architecture Capability Profile that describes the business, operational and technical aspects for service-oriented software. It is designed based on an Enterprise Architecture Framework (TOGAF) and the best practices related to the implementation of ISO 16100 standard concepts. An exploitation methodology of the designed capability profile is proposed and based on the alignment of a requirements engineering process with the Architecture Development Method from TOGAF. These latter evolve together to investigate the highest functional and technical compatibility of the desired functionalities and related constraints, respond to end-user requirements, and efficiently reuse the qualified solutions. Finally, we provided an implementation with an industrial use case to demonstrate the effectiveness of this approach. Concepts presented in this research work have been implemented as open-source prototypes based on Node JS and Java platforms. These prototypes cover the entire exploitation process that leads to the targeted ready-to-use business application.

## 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 programme 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. Suzanne Robertson and James Robertson. *Mastering the requirements process: Getting requirements right (3rd Edition)*. Addison-wesley, 2012.
2. Ian Sommerville. Software engineering 9th edition. *ISBN-10137035152*, 2011.
3. Abdelhadi Belfadel, Emna Amdouni, Jannik Laval, Chantal Bonner Cherifi, and Nejib Moalla. Towards software reuse through an enterprise architecture-based software capability profile. *Enterprise Information Systems*, pages 1–42, 2020.
4. R Gosselt. A maturity model based roadmap for implementing togaf. In *17th Twente Student Conference on IT*, 2012.
5. Abdelhadi Belfadel, Jannik Laval, Chantal Bonner Cherifi, and Néjib Moalla. Semantic software capability profile based on enterprise architecture for software reuse. In *International Conference on Software and Software Reuse*, pages 3–18. Springer, 2020.
6. The Open Group. *The Open Group Architecture Framework TOGAF™ Version 9*. Basharat Hussain, 2009.
7. Roel J Wieringa. Requirements engineering: Problem analysis and solution specification. In *International Conference on Web Engineering*, pages 13–16. Springer, 2004.
8. S ANSI. Ieee. ieee guide to software requirements specifications, 1984.
9. MA Davis. Software requirements. *OBJECTS FUNCTIONS & STATUS*, 1993.
10. Syed Waqas Ali, Qazi Arbab Ahmed, and Imran Shafi. Process to enhance the quality of software requirement specification document. In *2018 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–7. IEEE, 2018.
11. Zahoor Ahmad, Musarrat Hussain, Abdur Rehman, Usman Qamar, and Muhammad Afzal. Impact minimization of requirements change in software project through requirements classification. In *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, page 15. ACM, 2015.
12. Abeer Abdulaziz Alsanad, Azeddine Chikh, and Abdulrahman Mirza. A domain ontology for software requirements change management in global software development environment. *IEEE Access*, 7:49352–49361, 2019.
13. Xiaofei Xu, Ruilin Liu, Zhongjie Wang, Zhiying Tu, and Hanchuan Xu. Re2sep: A two-phases pattern-based paradigm for software service engineering. In *2017 IEEE World Congress on Services (SERVICES)*, pages 67–70. IEEE, 2017.
14. Huafeng Chen and Keqing He. A method for service-oriented personalized requirements analysis. *Journal of Software Engineering and Applications*, 4(01):59, 2011.
15. Konstantinos Zachos, Neil Maiden, Xiaohong Zhu, and Sara Jones. Discovering web services to specify more complete system requirements. In *International Conference on Advanced Information Systems Engineering*, pages 142–157. Springer, 2007.
16. Bertrand Verlaine, Ivan Jureta, and Stephane Faulkner. Towards conceptual foundations of requirements engineering for services. In IEEE Computer, editor, *Proceedings of the Fifth IEEE International Conference on Research Challenges in Information Science (RCIS 2011), Gosier, Guadeloupe*, pages 147–157. IEEE Computer society, 2011. Publication editors : IEEE Computer Society.
17. Ivan J Jureta, John Mylopoulos, and Stéphane Faulkner. A core ontology for requirements. *Applied Ontology*, 4(3-4):169–244, 2009.
18. Dumitru Roman, Uwe Keller, Holger Lausen, Jos De Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied ontology*, 1(1):77–106, 2005.
19. Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93, 2002.

20. K Breininger, F Najmi, and N Stojanovic. The ebxml registry repository version 3.0. 1. *OASIS, Febuary*, 2007.
21. Massimo Paolucci, Takahiro Kawamura, Terry R Payne, and Katia Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347. Springer, 2002.
22. Jian Wu and Zhaohui Wu. Similarity-based web service matchmaking. In *2005 IEEE International Conference on Services Computing (SCC'05) Vol-1*, volume 1, pages 287–294. IEEE, 2005.
23. Marta Sabou and Jeff Pan. Towards semantically enhanced web service repositories. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):142–150, 2007.
24. Jian Yu, Quan Z Sheng, Jun Han, Yanbo Wu, and Chengfei Liu. A semantically enhanced service repository for user-centric service discovery and management. *Data & Knowledge Engineering*, 72:202–218, 2012.
25. Konstanty Haniewicz. Local controlled vocabulary for modern web service description. In *International Conference on Artificial Intelligence and Soft Computing*, pages 639–646. Springer, 2012.
26. 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*, pages 533–538, Dec 2013.
27. 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)*, pages 295–298, Sep. 2013.
28. [28] Jonas, Philipp Brune, and Heiko 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*, pages 258–265. IEEE, 2014.
29. Tom Narock, Victoria Yoon, and Sal March. A provenance-based approach to semantic web service description and discovery. *Decision Support Systems*, 64:90–99, 2014.
30. 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)*, pages 202–206, Nov 2015.
31. Sihem Ben Sassi. Towards a semantic search engine for open source software. In *International Conference on Software Reuse*, pages 300–314. Springer, 2016.
32. Kavitha Esther Rajakumari. Towards a novel conceptual framework for analyzing code clones to assist in software development and software reuse. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 105–111. IEEE, 2020.
33. Elena Goncharuk. A case study on pragmatic software reuse, 2021.
34. Matthias Loskyll, Jochen Schlick, Stefan Hodek, Lisa Ollinger, Tobias Gerber, and Bogdan Pîrvu. Semantic service discovery and orchestration for manufacturing processes. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8. IEEE, 2011.
35. Hamida Seba, Sofiane Lagraa, and Hamamache Kheddouci. Web service matchmaking by subgraph matching. In Joaquim Filipe and José Cordeiro, editors, *Web Information Systems and Technologies*, pages 43–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
36. Aabhas V Paliwal, Basit Shafiq, Jaideep Vaidya, Hui Xiong, and Nabil Adam. Semantics-based automated service discovery. *IEEE Transactions on Services Computing*, 5(2):260–275, 2011.
37. Yang Xue, Chunhong Zhang, and Yang Ji. Restful web service matching based on wadl. In *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 364–371. IEEE, 2015.
38. Maya Rathore and Ugrasen 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)*, pages 165–171. IEEE, 2013.
39. Hanane Becha and Sana Sellami. Prioritizing consumer-centric nfps in service selection. In *International Conference on Conceptual Modeling*, pages 283–292. Springer, 2014.

40. Amandeep Kaur Sandhu and Ranbir Singh Batth. Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm. *Software: Practice and Experience*, 51(4):735–747, 2021.
41. Miguel Ángel Rodríguez-García, Rafael Valencia-García, Francisco García-Sánchez, and J Javier Samper-Zapater. Ontology-based annotation and retrieval of services in the cloud. *Knowledge-Based Systems*, 56:15–25, 2014.
42. Georgia M Kapitsaki. Annotating web service sections with combined classification. In *2014 IEEE International Conference on Web Services*, pages 622–629. IEEE, 2014.
43. Niranjan N Chiplunkar et al. Dynamic search and selection of web services. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 1532–1536. IEEE, 2014.
44. Rong Li, Keqing He, and Sai 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*, pages 320–325. IEEE, 2013.
45. Michiko Matsuda, Kiminobu Kodama, Satoshi Noguchi, Sakuyuki Onishi, Toshikatsu Asano, Takuya Horikita, and Kousuke Komatsubara. Configuration of a production control system through cooperation of software units using their capability profiles in the cloud environment. *Procedia CIRP*, 17:416–421, 2014.
46. Rosa Alarcon, Rodrigo Saffie, Nikolas Bravo, and Javiera Cabello. Rest web service description for graph-based service discovery. In *International Conference on Web Engineering*, pages 461–478. Springer, 2015.
47. Yehia Elshater, Khalid Elgazzar, and Patrick Martin. godiscovery: Web service discovery made efficient. In *2015 IEEE International Conference on Web Services*, pages 711–716. IEEE, 2015.
48. Nicolas Boissel-Dallier, Fréderick Benaben, Jean-Pierre Lorré, and Hervé Pingaud. Mediation information system engineering based on hybrid service composition mechanism. *Journal of Systems and Software*, 108:39 – 59, 2015.
49. Emna Khanfir, Raoudha Ben Djmeaa, and Ikram Amous. Quality and context awareness intention web service ontology. In *2015 IEEE World Congress on Services*, pages 121–125. IEEE, 2015.
50. Sophea Chhun, Néjib Moalla, and Yacine Ouzrout. Qos ontology for service selection and reuse. *Journal of Intelligent Manufacturing*, 27(1):187–199, 2016.
51. Lalit Purohit and Sandeep Kumar. Web service selection using semantic matching. In *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, page 16. ACM, 2016.
52. Khalid Elgazzar, Hossam S Hassanein, and Patrick Martin. Daas: Cloud-based mobile web service discovery. *Pervasive and Mobile Computing*, 13:67–84, 2014.
53. Furkh Zeshan, Radziah Mohamad, Mohammad Nazir Ahmad, Syed Asad Hussain, Adnan Ahmad, Imran Raza, Abid Mehmood, Ikram Ulhaq, Arafat Abdulgader, and Imran Babar. Ontology-based service discovery framework for dynamic environments. *IET Software*, 11(2):64–74, 2017.
54. Wenxin Mu, Frederick Benaben, and Herve Pingaud. An ontology-based collaborative business service selection: contributing to automatic building of collaborative business process. *Service Oriented Computing and Applications*, 12(1):59–72, 2018.
55. Iso 16100-1:2009 industrial automation systems and integration – manufacturing software capability profiling for interoperability – part 1: Framework, 2009.
56. Microsoft Patterns and Practices Team. *Microsoft® Application Architecture Guide, 2nd Edition (Patterns and Practices)*. Microsoft Press, 2009.
57. Iso/iec 25010:2011 systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models, 2011.
58. David Martin, Massimo Paolucci, Sheila McIlraith, Mark Burstein, Drew McDermott, Deborah McGuinness, Bijan Parsia, Terry Payne, Marta Sabou, Monika Solanki, et al. Bringing semantics to web services: The owl-s approach. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 26–42. Springer, 2004.
59. Aurona Gerber, Paula Kotzé, and Alta Van der Merwe. Towards the formalisation of the togaf content metamodel using ontologies. 2010.
60. Marco Rospocher, Chiara Ghidini, and Luciano Serafini. An ontology for the business process modelling notation. In *FOIS*, pages 133–146, 2014.

61. Werner Ceusters. An information artifact ontology perspective on data collections and associated representational artifacts. In *MIE*, pages 68–72, 2012.
62. Andrew Josey, Marc Lankhorst, Iver Band, Henk Jonkers, and Dick Quartel. An introduction to the archimate® 3.0 specification. *White Paper from The Open Group*, 2016.
63. Hind Benfenatki, Catarina Ferreira Da Silva, Aïcha-Nabila Benharkat, Parisa Ghodous, and Zakaria Maamar. Linked usdl extension for describing business services and users' requirements in a cloud context. *International Journal of Systems and Service-Oriented Engineering (IJSSOE)*, 7(3):15–31, 2017.
64. Brad Green and Shyam Seshadri. *AngularJS*. " O'Reilly Media, Inc.", 2013.
65. M Cantelon, M Harter, TJ Holowaychuk, and N Rajlich. Node. js in action, greenwich, ct, 2013.
66. Apache Jena. Fuseki: serving rdf data over http, 2014.
67. Peter Eeles. Capturing architectural requirements, 2005. Accessed: 2020-05-10.