Approach to the Design of a Microservice Architecture Based on Praxeme

Mihajasoa Léa Fanomezana
Laboratory for Mathematical and Computer Applied to the Development Systems (LIMAD)
University of Fianarantsoa, Madagascar
fmihajasoalea@gmail.com

Andrianjaka Miary Rapatsalahy
Laboratory for Mathematical and Computer Applied to the Development Systems (LIMAD)
University of Fianarantsoa, Madagascar
andrianjaka92@yahoo.fr

Nicolas Raft Razafindrakoto
Laboratory of Multidisciplinary Applied Research (LRAM)
University of Antananarivo, Madagascar
rnraft@gmail.com

Costin Bădică
Faculty of Automation, Computers and Electronics (ACE)
University of Craiova, Romania
costin.badica@edu.ucv.ro

***Abstract*— Frequent updates of business needs are one of the factors of the evolution of a company. These permanent changes require a large flexibility of the Information System (IS). SOA (Service Oriented Architecture) is an architecture that offers more scalability to an application by subdividing the monolithic block into independent services. Nevertheless, it is not sufficiently suitable in terms of accessibility of services and data. It is thus necessary to orient the design of the IS towards a new architecture called Microservice Architecture (MSA). The objective of this paper is to recommend a methodology to design MSA. Indeed, our approach is based on Praxeme which is an enterprise methodology appropriate to SOA. The result obtained from the approach proposed in this paper is a model allowing to automate the MSA design.**

Keywords—SOA, MSA, MDA, UML, Monolithic Architecture, software architecture design methodology, Praxeme, ReLEL

I. INTRODUCTION

Lately, organizations are facing a challenge of updates. The inability to govern the frequent changes in needs attributes to negative effects to the business. Current technologies are also evolving very fast and some are providing better qualities. Thus, the company must be able to adapt to new features and developments proposed. However, the update mechanism is long and complicated with a traditional application because the whole must be scaled. This structure is called "Monolithic Architecture"[1]. It is a less scalable method because the more the IS extends, the more interdependence occurs within the application. Furthermore, the adoption of cloud computing is sucking in researchers as well as practitioners lately [2]. Therefore, the design and implementation of an IS requires a new and more practical approach.

This is how the Service Oriented Architecture or SOA appeared in the early 2000s. It is an approach to software design and development that ensures the independence between the different software artifacts [3]. Its objective is to offer a flexible application by subdividing it into several independent services. SOA has specificities that are the interoperability of several applications or services, reusability and modularization [4]. However, it has limitations in terms of the communication bias of the Enterprise Services Bus (ESB) and the monolithism of its database. In other words, the deployment of all services must be reviewed if one of them is blocked. Moreover, the services in the bus may be inaccessible in case of ESB failure while an infrastructure such as ESB is obligatory in SOA [5]. (Oberhauser, R., and al., 2017) [6] also argued that SOA is a heavyweight architecture.

Then, a more advanced version of SOA called Microservice Architecture (MSA) came along. (Rademacher, F., and al., 2017) [7] claim that MSA is lighter and less complex than SOA due to its reduced service terminology. Each microservice operates in its own process. According to the article [8], microservices are described as independently developed and deployable elements.

This paper aims at proposing an approach for the automatic design of MSA. An analysis on enterprise methodologies has shown the adaptation of the Praxeme enterprise architecture methodology with SOA. In other terms, Praxeme is an enterprise methodology that adopts model-driven architecture (MDA) and SOA to organize all aspects of the enterprise [9]. Since MSA inherits the SOA principle of orienting the application into autonomous services, our approach thus relies on Praxeme for the design of MSA. We suggest an approach hypothesis based on model transformation rules. As a result, we were able to obtain a model that represents a speculation for the automatic design of MSA.

This paper is subdivided as follows, section II elaborates a study on the concepts of MSA and SOA as well as the comparison between the two architectures. Then, an analysis on the works of enterprise architecture methodologies is elucidated in section III. Subsequently, our proposed approach is presented in section IV which describes the Praxeme framework and illustrates a hypothesis on the design of MSA. Finally, Section V concludes the paper and discusses future work.

II. OVERVIEW OF SERVICE ORIENTED ARCHITECTURE AND MICROSERVICE ARCHITECTURE

A. *Service Oriented Architecture (SOA)*

In the past, software development was based on a monolithic architecture where the entire application was unified in a single code base. In fact, the maintenance and extension of the IS becomes very complex because its functionalities and components are interconnected and interdependent, hence the intervention of the SOA.

The Service Oriented Architecture or SOA is a structure that consists of facilitating the design and development of an IS by dividing it into several basic elements called "services". It is an approach that makes the application more flexible and more agile in the face of changing needs. It promotes modularization, reusability and autonomy of services with a system of weak coupling [10]. SOA also provides the cloud as an environment for development and deployment of the system

In addition, SOA provides a means that simplifies the communication between services and promotes the interoperability of multiple applications or services. To do this, services are provided and published by providers with a standardized style description in service registries that consumers can access and use. Thus, SOA consists of three actors, namely the main actors that are the service provider and the service consumers and the agencies that allow consumers to find services (Figure 1) [11].

The notion of the ESB or Enterprise Services Bus has been introduced into SOA. It is an IT tool that handles the communication between services and takes care of the messages and their transformations.

The main standards for web services in SOA are SOAP, WSDL, and UDDI [12]

- SOAP (Simple Object Access Protocol) describes a communication process between web services.
- WSDL (Web Services Description Language) is an XML language that describes the web service
- UDDI (Universal Description, Discovery, and Integration) is an element that is used to locate the web service sought on the network

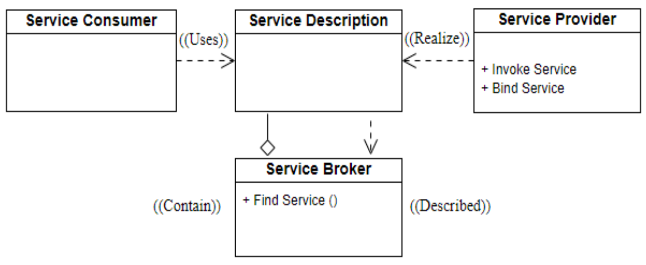

Fig. 1. Conceptual model of an SOA architectural style *[11]*

B. *Microservice Architecture (MSA)*

Microservice Architecture or MSA is an architectural style of designing and implementing an IS. MSA inherits the concept of SOA, which consists in dividing a system into smaller particles. These particles are called "Microservices".

This paper is based on two main motivations for choosing MSA over SOA.

- At the communication level, SOA has the ESB bus which seems to be heavy and very critical in case of failure

- The SOA database is still monolithic where all services have a common database

Indeed, MSA is a modern approach to develop an IS by breaking it down into reduced services that each have their own processes and communicate via lightweight mechanisms such as HTTP and REST API. (Fowler,M. and al., 2014) [13] defines MSA as a set of reduced services that are realized and deployed separately (figure 2). This architecture prioritizes software flexibility as well as service quality and security through the independence of services, testing and deployment of microservices. (Cojocaru, M. D., and al., 2019) [14] confirm that research and business sectors have recently become interested in MSA because of its various advantages. For example, powerful and modern platforms such as Netflix, Amazon and SoundCloud have turned to this famous approach [15]

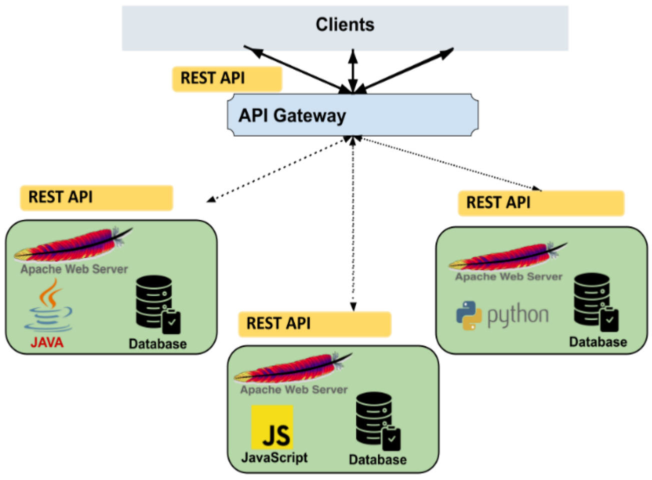

Fig. 2. Microservices Architecture

C. *Comparison Between SOA and MSA*

Recently, the research and industry field became aware of the complexity of maintaining monolithic applications and focused on SOA for IS flexibility. Then a newer and more refined version of SOA named "microservice" was becoming very apparent from the year 2014 [13]. Both architectures are based on services as building blocks for better application design and development. In other terms, the concept is defined by splitting the monolithic system into several autonomous services. Indeed, the services can be developed by different programming methodologies. On the other hand, MSA has smaller services compared to SOA [16]. And in terms of communication and protocols, MSA often proposes lighter mechanisms such as REST API while SOA generally uses standard protocols such as SOAP and ESB as a tool for exchange between services. The most obvious difference between MSA and SOA is in the database and the granularity of the services. Services in SOA have only one data store, whereas in MSA each microservice has its own database. In summary, (Fanomezana, M. L., and al., 2022) [2] illustrated Table I for a visibility of the distinction of the two architectures.

TABLE I. COMPARATIVE ANALYSIS OF MSA AND SOA [2]

	SOA (Service Oriented Architecture)	MSA (Microservice Architecture)
Definition [17]	Software architecture for the design and implementation of an IS based on the decomposition of the application into several independent services	Software architecture for the design and implementation of an IS based on the decomposition of the application into several autonomous microservices
Objective [17]	Overcome the problem of interdependence and make the IS more flexible to changing needs	Overcome the problem of interdependence and make the IS more flexible to changing needs
Sharing of resources between departments	Promotes the sharing of resources as much as possible [15]	Ensures departmental autonomy by sharing fewer resources [15]
Remote access protocols	Uses SOAP as standard protocol for remote access [12]	Uses lightweight protocols such as REST [17], [18]
Communication	The communication mechanism between services is done through the ESB	The communication mechanism between services is done through the API
Granularity	Services can be varied from fine-grained to large-grained services[7]	Service on a small scale [17]
Scope or coverage	Has an enterprise scope that quite often contains a set of application services, which are also constituted by several infrastructure services [15], [17]	Has an application scope where each microservice corresponds to a small application with its own hexagonal architecture [17]
Governance	Provides governance protocols common to all departments [19]	Provides decentralized governance [20]
Reusability	Promotes service reusability in an integrated service infrastructure [19]	Prefer to rebuild the code than to reuse it [16]
Data storage [16]	All services share the same data storage	Each microservice manages its own database
Interoperability [17]	Each service can operate on different technologies	Each service can operate on different technologies
Cloud-based	yes	yes

III. State of the Art on Enterprise Architecture Frameworks

Enterprise architecture is an approach that allows organizations to have a global vision of all its aspects and their relationships. It is a way to make the business areas, the automation aspects and the technological aspect collaborate to simplify the management of the IS. Otherwise stated, the role of this method is to ensure the alignment of IT sides, strategies and current business standards and also to design and develop required IS [2], [21]. However, enterprise architecture is usually applied with a methodological framework to facilitate its application at the organizational level. Indeed, among the existing methodological frameworks we will study the Zachman framework, the Open Group Architecture Framework (TOGAF), the Federal Enterprise Architecture Framework (FEAF) and the Praxeme methodology.

First, the Zachman framework was initiated by an American business and computer consultant named John A. Zachman in 1987 [22] then revised and extended in 1993 and 1999 [23].It is an enterprise methodological framework that classifies and structures descriptive representations of an enterprise in different dimensions. Its objective is to describe IS artifacts and ensure the implementation of standards for creating the information environment in an adequate way [24].

The article [24] mentioned that the implementation of enterprise architecture is not an easy task to perform. There are several challenges. Thus, the authors proposed an approach to facilitate the development of an enterprise architecture based on the business and technical perspectives of the Zachman framework. Next, an enterprise integration methodology using the Zachman framework was suggested after finding that enterprise integration increases an organization's enterprise architecture skills [25]. (Alwadain, A. S. A., and al., 2010) focused on the basic principle of the Zachman framework and analyzed various attempts to improve it to fit SOA and services. Their analysis led them to conclude that there is a lack of agreement on the SOA positions in the Zachman framework [21]. The paper [26] also aimed to present different models of SOA architecture and to demonstrate the place of SOA in the framework. In other words, the authors tried to introduce SOA into the Zachman framework. An approach to modeling SOA in the enterprise architecture framework was also discussed in the paper [27]. This discussion consists of extending the Zachman framework and allowing it to include service oriented artifacts exploiting notions of business and software service orientation within the enterprise.

From the studies of the Zachman framework, we see that it is a very popular and widely used methodology. (Benkamoun and al., 2014) similarly confirm this in his paper [28]. On the other hand, we do not see satisfactory and accurate results about the introduction of the SOA paradigm in the Zachman framework. The perspective of the paper [21] is based on an analysis that takes into account more current enterprise architecture frameworks like TOGAF. This means that it is a less modern Zachman framework compared to other architecture frameworks.

As for the TOGAF methodology, The Open Group Architecture Framework or TOGAF was proposed by The Open Group in 1990 as a strategy for the development of architecture in the context of enterprise architecture. TOGAF

1.0 is initially presented in reference to the technical architecture for information management at the US Department of Defense [29]. The version 8 of TOGAF was projected in 2004. Then the last version which is TOGAF 9 was launched in 2009 [2]. It consists of a framework called the Architecture Development Method (ADM). It is a large part of TOGAF that indicates how an enterprise architecture can meet specific business needs [30]. TOGAF is an enterprise architecture framework that has the objective of designing, planning and organizing the infrastructure of the IT system [31]. (Sofyana, L., and al., 2019) generated enterprise architecture planning in a college that has a vision and mission considering the development of IT technologies with TOGAF [30]. The latter showed that the approach led to an effective outcome. TOGAF was also exploited by [32] to design an architecture in a large-scale SOA-based research project. The paper concluded that combining the TOGAF framework with SOA increases the effectiveness of SOA such as reusability and flexibility of the enterprise IS. However, [32] produced more results at the research phase than in the domain of an enterprise. Then, a modeling approach aligned with BPM and SOA based on the TOGAF architecture framework was proposed by [33] to obtain a more agile architecture. On the other hand, the researchers in the paper [34] combined the TOGAF and SOA framework for service innovation in the general government office in Indonesia due to a problem of updating the integration process which is limited. The results showed that TOGAF will be the guide for the system's integrated business process with SOA as the technical approach.

TOGAF has advantages such as its good alignment between business and technology, its popularity and the existence of clear and detailed steps for building a business process and IS architecture [30], [33]. Nevertheless, The Open Group found the lack of SOA development support in TOGAF 9 and decided to make a consideration to fill this gap but TOGAF 9 does not yet include the results [35]. In addition, TOGAF also has limitations in terms of the lack of information on the maintenance of the framework, the lack of integration between the different proposed artifacts, and the exclusion of strategic aspects [2].

FEAF or Federal Enterprise Architecture Framework is one of the frameworks studied in this paper. The FEAF was developed by the US federal government to unite its agencies and functions under a common enterprise architecture. Its version 2 was released in May 2012 as part of an improvement in the deployment of IT services [36]. The Federal Enterprise Architecture (FEA) is built through a collection of reference models namely PRM, BRM, DRM, SRM, TRM [37]. This framework consists of defining the planning of the enterprise architecture and has the vision of simplifying and implementing common processes and information across federal agencies. The description of an enterprise architecture requires a methodological framework. This led (Mahdavifar and al., 2012) to propose a method for integrating concepts from the FEA framework and its Business Reference Model (BRM) and an International Software Testing Qualification Board (ISTQB) framework for the enterprise architecture testing process [37]. Then, (Defriani & Resmi, 2019) used FEAF for e-government architecture planning in Purwakarta districts in Indonesia. The study is to improve the quality of services in governance and resulted in an e-government architecture as well as a plan for the implementation of the e-government application in the concerned districts [36]. In 2021, the Regional Government Organization of Mataram City created integrated services through the implementation of the e-government system by focusing solely on simplifying the service process. The paper [38] used the Enterprise Architecture Planning (EAP) method for planning a roadmap to ensure integration and interoperability with other electronic systems. Both are then enhanced by the FEAF framework and the SOA software architecture.

FEAF has been found to have its strength in being able to describe and plan the enterprise architecture in a detailed and simplified manner. However, papers that exploit the combination of the FEA framework and SOA are very rare.

However, (Thierry and al., 2013) [39] found that these Enterprise Architecture frameworks of American origins are often inadequate, too heavy to design and less coherent.

- The Zachman framework shows its complexity through the 30 different aspects of the Enterprise

- The TOGAF framework does not take into account models and their transformation.

The article [39] has thus proposed an emergent approach which is Praxeme. The authors have demonstrated from an experimentation the reliability and the adequacy of Praxeme especially at the level of the transformation of the models and the design of the modeling until the operational stage.

Nevertheless, works such as [40], [9], [41] have noticed the absence of the model representing the intentional and semantic aspect of the company in Praxeme. Indeed, (Razafindramintsa and al., 2016) [41] presented a method that automatically derives the semantic aspect of Praxeme using the natural language model. The approach consists of transforming the eLEL (elaborate Lexicon extended language) requirement model into a business model at the Praxeme methodology level. (Rapatsalahy and Al., 2020, 2021) [9], [40], [42] extended the eLEL requirement model into ReLEL (Restructuring extended Lexical elaborate Language) and initialized it in Praxeme for automatic generation of the logical aspect of Praxeme as well as web services and then software components that represent the software aspect of Praxeme.

Praxeme takes the basic concepts of the methods of the last thirty years, i.e. the Zachman framework, Merise and other design methods and updates them. The articles [39] confirms that this methodology is open source and has more documentation. Indeed, Praxeme is a framework that appears to be one of the most recent, modern and advanced methodologies that have succeeded. Moreover, (Rapatsalahy and al., 2021) [9] states that it is an approach that takes into consideration the combination of SOA and MDA for the design and development of IS. However, (Fanomezana, M. L., 2022) confirm that Praxeme does not consist of a model for describing the intentional aspect of the enterprise [2]

IV. Proposed Approach

In this paper, we suggest an approach for the automatic generation of MSAs based on the Praxeme methodology. To do so, we will study the basic concept of Praxeme and develop a model that allows us to present our hypothesis of approach for the design of MSA. The MSA architecture is mainly composed of an orchestration container in which the microservice container is located. This container is in turn composed of microservices that each have their own database.

A. Concept of the Praxeme

Praxeme is an enterprise architecture methodology that covers all aspects of the organization's systems from strategy to deployment. It is a methodology of French origin coming from the words "praxis" (action) and "semeion" (meaning) which means "the meaning of action". It was mentioned in part "overview of the Service Oriented Architecture" which is in section II.A that SOA makes the design and development of an IS easier and ensures the flexibility of the application in the face of constant updates of the business needs. Thus, the design of SOA requires an adequate enterprise architecture methodological framework. Moreover, the Praxeme methodology suggests the best condition to be adapted to SOA in order to better take advantage of the privileges of SOA [42]. Praxeme also suggests UML as a modeling language for each aspect of the enterprise, MDA as an approach to automate the transition from one aspect to another, and SOA to manage changes in business requirements [2].

The Praxeme methodology proposes a framework for representing the reality of the enterprise that is composed of seven aspects (Figure 3). The design of SOA services is carried out at the level of the logical aspect, which is one of these aspects that elaborate Praxeme. In other words, Praxeme enables the IS to be organized by breaking it down into several components called "logical services". These represent the logical aspect of Praxeme, which is illustrated in Figure 4. The approach is based on the use of MDA with which rules are proposed to automatically derive SOA logical services from semantic or pragmatic models [42]

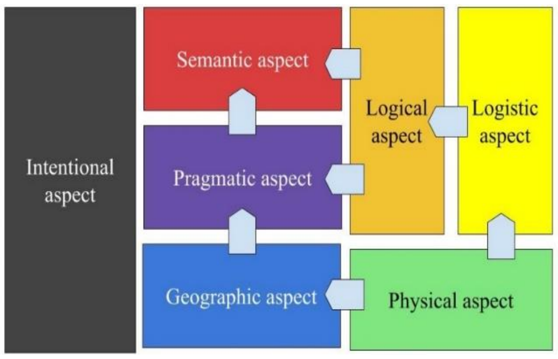

Fig. 3. The topology of the enterprise system

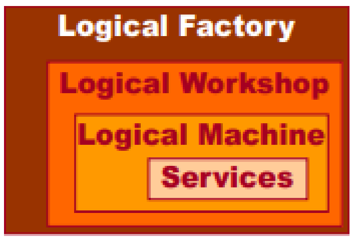

Fig. 4. Metaphorical terminology applied to the logical aspect

Figure 4 shows the logical aspect of Praxeme containing the logical factory which in turn is the logical machine in which the services are located. An automatic generation of SOA web services has already been realized by (Rapatsalahy and Al, 2021) [42]. The authors of the paper used the MDA approach to model the WSDL document from the Praxeme logic factory via derivation rules. The WSDL model is then translated into an XML file that describes the SOA web service [2]. However, (Fanomezana, M. L., and al., 2022) [2] asserts that the fragmentation of the logical workshop during logical modeling using Praxeme and SOA is very time consuming. It is the reason that we propose to automatically design MSA from the logical workshops of the Praxeme logical model.

According to the article [2], the idea of the term "Microservice" consists in fragmenting the application into the smallest possible entities. The idea of the article [2] lies in the exploitation of the logical machine which is the smallest component of the Praxeme logical model to design microservices. Indeed, the MSA components are modeled from the constituents of the logical aspect of Praxem. Therefore, the modeling of the orchestration container, the microservices container, the microservices is respectively accomplished from the logical factory, the logical workshop and the logical machine. In addition, [2] also proposes that the "data structure" component in the logical machine that is designed from the attributes of the semantic model models the database of each microservice.

B. Elaboration of the Model for the Design of the MSA

As our method is based on Praxeme, we rely on the MDA approach for model transformation in order to assemble the Praxeme aspects. Unlike the code-centric approach, design patterns do not change over time. MDA is thus an approach that is centered on models for the realization of IS. It is based on the notion of models, meta-models and model transformation [43].

The principle of MDA is based on four main elements which are the requirement model CIM or computation independent model, then the analysis and design model PIM or platform independent model, the code model at the software development phase PSM or platform specific model for the implementation of the system and finally the code (Figure 5) [44].

The transformation of the models can be divided into two categories

- M2M (Model to Model) transformation is the change from models to models

- M2T (Model to Text) transformation is the transformation of a model to the generation of a code or a file

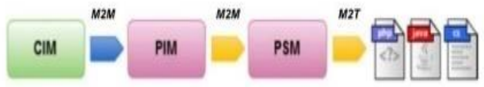

Fig. 5. The models involved in the MDA approach

For this purpose, we need to define a source model and a target model. Knowing that Praxeme does not have a model that represents its intentional aspect, we propose the requirement model ReLEL or Restructuring extended Lexical elaborate Language to describe this aspect. ReLEL will thus be considered as a source model while the model corresponding to each MSA component will be the target model.

The approach of our hypothesis is globalized in the steps shown in Figure 6. First, the semantic aspect is derived from the ReLEL requirement model which describes the intentional aspect of Praxeme. This step is followed by a modeling of the semantic aspect of Praxeme which is presented by UML diagrams. Then, the next phase consists in transforming the semantic model into a logical model via the M2M mechanism of MDA. After that, the logical model obtained is transformed into an MSA component model by the same transformation mechanism. Finally, the microservices components are automatically generated from the MSA component model via the M2T concept.

The ReLEL model, the semantic model, the logical model and the MSA component model each have their own metamodel descriptions.

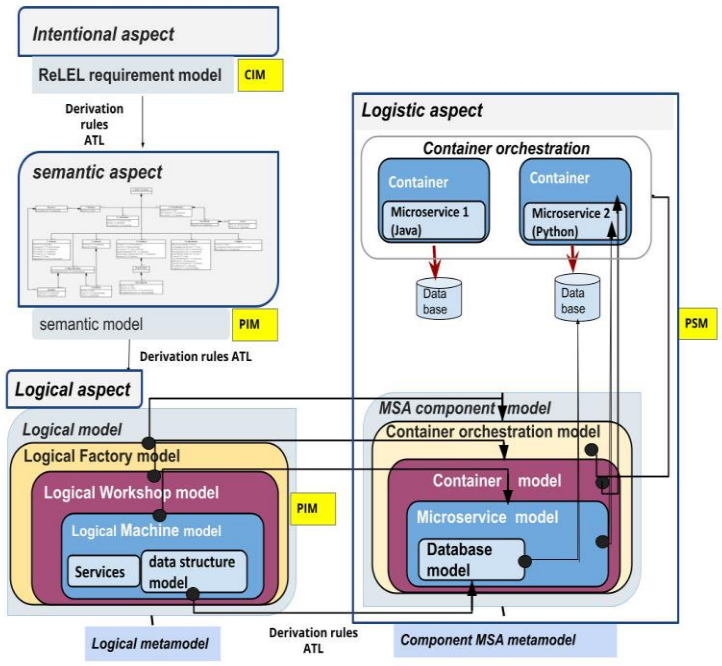

Fig. 6. Model of automating the generation of MSA components based on Praxeme

We have thus developed the following transformation rules describing the passage from one component to another

Rule 1 : The ReLEL requirement model is transformed into a semantic model

Rule 2 : The semantic model is transformed into a logical model.

Rule 3 : The logical factory model is transformed into a container orchestration model.

Rule 4 : The shop logic model is transformed into a container model

Rule 5 : The machine logic model is transformed into a microservice model.

Rule 6 : The data structure model is transformed into a database model.

Rule 7 : The container orchestration model is transformed into a container orchestration.

Rule 8 : The container microservice model is transformed into a container microservice

Rule 9 : The microservice model is transformed into a microservice.

Rule 10 : The database model is transformed into a database

We have therefore produced a final model that illustrates our hypothesis for the automatic generation of MSA components from the Praxeme methodology. We have as a source element of the approach the semantic model of Praxeme which represents its semantic aspect. Then, we obtained the MSA components as a result via derivation rules.

V. Conclusion and Future Work

In this paper, we have presented an approach to manage the complexities of the organization in the face of changing needs and technologies. SOA has been proposed to make an application agile and flexible by fragmenting it into autonomous services. Due to its limitations in terms of service and data accessibility, this paper suggests MSA as a more modern software architecture. It is an approach based on microservices that each have their own development and deployment processes [13]. Indeed, the objective of this paper is to suggest a methodology that automates the design of MSA. Therefore, the employment of an architecture framework appropriate to SOA for designing MSA is evident due to the similarity of the two software architectures in terms of the basic concept named service [2].

This paper is the extension of the article which concluded that Praxeme is the very well adapted enterprise architecture framework for the design of MSA [2].

The approach we adopt is thus the Praxeme methodology. We propose an elaborated process relying on its aspects. To do so, transformation rules have been elaborated to facilitate the passage from the semantic aspect of Praxeme to its logistic aspect. Our approach is composed of different steps, namely the derivation of the semantic aspect from the ReLEL requirement model which describes the intentional aspect of Praxeme, then the modeling of the semantic aspect of Praxeme followed by the transformation of the semantic model into the logical model, the design of the MSA component model from the logical model and finally the generation of the microservices components from the MSA component model. As a result, we obtained a model that represents the hypothesis on the automatic design of the MSA architecture.

The model we obtained shows the potency of our Praxeme approach that takes into account the design of the MSA up to the level of the data structure. Our hypothesis is also more coherent because all the constituents of the Praxeme logic model describe respectively the MSA components to be designed. In the future work, we plan to deepen the study on the MSA especially on its components to be generated and to experiment the suggested methodology.

References

[1] F. Ponce, G. Márquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A Rapid Review", in *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, 2019, pp. 1–7.

[2] M. L. Fanomezana, A. M. Rapatsalahy, N. R. Razafindrakoto, and C. Bădică, "Proposed Methodology for Designing a Microservice Architecture", in *2022 23rd International Carpathian Control Conference (ICCC)*, 2022, pp. 303–308.

[3] T. Zhang, S. Ying, S. Cao, and X. Jia, "A modeling framework for service-oriented architecture", in *2006 Sixth International Conference on Quality Software (QSIC'06)*, 2006, pp. 219–226.

[4] C. F. Fang and L. C. Sing, "Collaborative learning using service-oriented architecture: A framework design", *Knowledge-Based Systems*, vol. 22, no. 4, pp. 271–274, 2009.

[5] R. Berbner, T. Grollius, N. Repp, O. Heckmann, E. Ortner, and R. Steinmetz, "An approach for the management of service-oriented

architecture (soa) based application systems", *Enterprise modelling and information systems architectures*, 2005.

[6] R. Oberhauser and S. Stigler, "Microflows: enabling agile business process modeling to orchestrate semantically-annotated microservices", in *Seventh International Symposium on Business Modeling and Software Design (BMSD 2017), Volume 1*, 2017, pp. 19–28.

[7] F. Rademacher, S. Sachweh, and A. Zündorf, "Differences between model-driven development of service-oriented and microservice architecture", in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017, pp. 38–45.

[8] M. Grambow, L. Meusel, E. Wittern, and D. Bermbach, "Benchmarking microservice performance: a pattern-based approach", in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, New York, NY, USA: Association for Computing Machinery, 2020, pp. 232–241. Accessed: Jan. 09, 2022. [Online]. Available: https://doi.org/10.1145/3341105.3373875

[9] R. M. Andrianjaka, H. Razafimahatratra, M. Ilie, T. Mahatody, S. Ilie, and N. R. Razafindrakoto, "Derivation of Logical Aspects in Praxeme from ReLEL Models.", in *ENASE*, 2021, pp. 413–420.

[10] M. Kasparick *and al.*, "OR. NET: a service-oriented architecture for safe and dynamic medical device interoperability", *Biomedical Engineering/Biomedizinische Technik*, vol. 63, no. 1, pp. 11–30, 2018.

[11] M. Mohammadi and M. Mukhtar, "A review of SOA modeling approaches for enterprise information systems", *Procedia Technology*, vol. 11, pp. 794–800, 2013.

[12] M. K. Haki and M. W. Forte, "Service oriented enterprise architecture framework", in *2010 6th World Congress on Services*, 2010, pp. 391–398.

[13] M. Fowler and J. Lewis, "Microservices, 2014", *URL: http://martinfowler. com/articles/microservices. html*, vol. 1, no. 1, pp. 1–1, 2014.

[14] M.-D. Cojocaru, A. Uta, and A.-M. Oprescu, "Attributes assessing the quality of microservices automatically decomposed from monolithic applications", in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2019, pp. 84–93.

[15] H. Bloch *and al.*, "A microservice-based architecture approach for the automation of modular process plants", in *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)*, 2017, pp. 1–8.

[16] J. A. Bigheti, M. M. Fernandes, and E. P. Godoy, "Control as a service: a microservice approach to Industry 4.0", in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*, 2019, pp. 438–443.

[17] A. Andriyanto, R. Doss, and P. Yustianto, "Adopting SOA and Microservices for Inter-enterprise Architecture in SME Communities", in *2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE)*, Oct. 2019, vol. 6, pp. 282–287. doi: 10.1109/ICEEIE47180.2019.8981437.

[18] G. Buchgeher, R. Ramler, H. Stummer, and H. Kaufmann, "Adopting Microservices for Industrial Control Systems: A Five Step Migration Path", in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2021, pp. 1–8. doi: 10.1109/ETFA45728.2021.9613622.

[19] M. K. Haki and M. Wentl, "Service-oriented business-it alignment: a SOA governance model", 2010.

[20] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study", *Journal of Systems and Software*, vol. 150, pp. 77–97, 2019.

[21] A. S. A. Alwadain, A. Korthaus, E. Fielt, and M. Rosemann, "Integrating SOA into an enterprise architecture-a comparative analysis of alternative approaches", in *Proceedings of the 4th International Conference on Research and Practical Issues of Enterprise Information Systems*, 2010, pp. 1–15.

[22] J. A. Zachman, "A framework for information systems architecture", *IBM Systems Journal*, vol. 26, no. 3, pp. 276–292, 1987, doi: 10.1147/sj.263.0276.

[23] T. Iyamu, "Implementation of the enterprise architecture through the Zachman Framework", *Journal of Systems and Information Technology*, 2018.

[24] C. M. Pereira and P. Sousa, "A method to define an Enterprise Architecture using the Zachman Framework", in *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 1366–1371.

[25] J. Espadas, D. Romero, D. Concha, and A. Molina, "Using the zachman framework to achieve enterprise integration based-on business process driven modelling", in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2008, pp. 283–293.

[26] V. Barekat, E. B. Nejad, and S. E. Alavi, "Definition of zachman framework cells based on Service Oriented Architecture", *International Journal of Scientific and Research Publications*, vol. 3, no. 9, pp. 1–8, 2013.

[27] S. Khoshnevis, F. S. Aliee, and P. Jamshidi, "Model driven approach to Service Oriented Enterprise Architecture", in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*, 2009, pp. 279–286.

[28] N. Benkamoun, W. ElMaraghy, A.-L. Huyet, and K. Kouiss, "Architecture framework for manufacturing system design", *Procedia CIRP*, vol. 17, pp. 88–93, 2014.

[29] A. Setiawan and E. Yulianto, "E-government interoperability and integration architecture modeling using TOGAF framework based on Service Oriented Architecture", *The Asian Journal of Technology Management*, vol. 11, no. 1, pp. 26–45, 2018.

[30] L. Sofyana and A. R. Putera, "Business architecture planning with TOGAF framework", in *Journal of Physics: Conference Series*, 2019, vol. 1375, no. 1, p. 012056.

[31] I. Saepurrahman and I. D. Sumitra, "Designing Enterprise Architecture for Sports Information System Platform Using the Open Group Architecture Framework Architecture Development Method", in *IOP Conference Series: Materials Science and Engineering*, 2019, vol. 662, no. 4, p. 042013.

[32] A. Kabzeva, M. Niemann, P. Müller, and R. Steinmetz, "Applying TOGAF to Define and Govern a Service-oriented Architecture in a Large-scale Research Project.", in *AMCIS*, 2010, p. 356.

[33] F. Ni and R. Li, "TOGAF for Agile SOA Modelling", 2017.

[34] A. Hodijah, S. Sundari, and A. C. Nugraha, "Applying TOGAF for e-government implementation based on Service Oriented architecture methodology towards good government governance", in *Journal of Physics: Conference Series*, 2018, vol. 1013, no. 1, p. 012188.

[35] M. Postina, J. Trefke, and U. Steffens, "An ea-approach to develop soa viewpoints", in *2010 14th IEEE International Enterprise Distributed Object Computing Conference*, 2010, pp. 37–46.

[36] M. Defriani and M. G. Resmi, "E-government architectural planning using federal enterprise architecture framework in Purwakarta districts government", in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–9.

[37] H. Mahdavifar, R. Nassiri, and A. Bagheri, "A method to improve test process in federal enterprise architecture framework using istqb framework", *International Journal of Computer and Information Engineering*, vol. 6, no. 10, pp. 1199–1203, 2012.

[38] M. Tajuddin and A. B. Maulachela, "Integration and Interoperability of Electronic-Based Government System (SPBE) Roadmap using Federal Enterprise Architecture Framework (FEAF) Method", in *Seminar Nasional Sistem Informasi (SENASIF)*, 2021, vol. 5, pp. 2889–2901.

[39] Thierry Biard, Michel Bigand, and Jean-Pierre Bourey, "La méthode Praxeme : une nouvelle approche de l'Architecture d'Entreprise", 2013.

[40] R. M. Andrianjaka, H. Razafimahatratra, T. Mahatody, M. Ilie, S. Ilie, and R. N. Raft, "Automatic generation of software components of the Praxeme methodology from ReLEL", in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, 2020, pp. 843–849.

[41] J. L. Razafindramintsa, T. Mahatody, and J. P. Razafimandimby, "Elaborate Lexicon Extended Language with a lot of conceptual information", *arXiv preprint arXiv:1601.01517*, 2016.

[42] R. M. Andrianjaka, R. Hajarisena, I. Mihaela, M. Thomas, I. Sorin, and R. N. Raft, "Automatic generation of Web service for the Praxeme software aspect from the ReLEL requirements model", *Procedia Computer Science*, vol. 184, pp. 791–796, 2021.

[43] A. Elmounadi, N. Berbiche, and N. Sefiani, "Model Driven Architecture: Model Transformation Methods and Tools", *La cinquième édition des Journées Doctorales en Technologies de l'Information et de la Communication (JDTIC'13), Kenitra, Maroc*, 2013.

[44] X. Blanc and O. Salvatori, "*MDA in Action: Model-Driven Software Engineering"*. Editions Eyrolles, 2011.