Service Modeling and Architecting for Self-Adaptive IoT-Based Systems

Fatima Zohra Merabet[1], Djamel Benmerzoug[1]

[1]*LIRE Laboratory, University of Constantine2 Abdelhamid Mehri, Constantine, Algeria*

Abstract

In recent years, the Internet of Things (IoT) environment dynamics has increased, leading to an increase in the complexity of modern software systems. Therefore, the need to manage these complex systems comes with several challenges, and service-oriented systems (SoS) are a good option to realize this. Several research initiatives have addressed this vision. However, there is a clear need for flexibility and self-adaptation for service-based IoT applications to promote higher customization and adaptation to manage unpredictable changes and uncertain situations. In our research, we advocate the use of the SoS paradigm for building next-generation services using the IoT, and the main objective of our work is to ensure the self-adaptation of these complex systems. In this paper, we provide a particular perspective on the evolution of the field of self-adaptation with a new conception. By proposing a BPMN extension named BPMN4SAS (BPMN for the self-adaptive system), we stress the idea of adaptation in the design phase, starting from the first step in the BPM (Business Process Management) life cycle. The proposed extension is illustrated through an example dealing with accident management in a smart city system. The second contribution of this paper concerns the optimization phase, where we propose a global architecture to ensure the self-adaptation of the IoT services at run time. The proposed architecture aims to let the system collect data about the uncertainties during execution. The system uses this data to resolve uncertainties, be aware of itself, and adjust and satisfy the changing conditions, especially the quality-of-service (QoS) constraints.

Keywords

Services-Oriented Systems (SoS), Internet of Things (IoT), Quality of Services (QoS), Adaptation, Self-adaptive system (SaS)

1. Introduction

The Internet of Things (IoT) is one of the technologies that has fully changed human lifestyles. Nowadays, it is becoming more and more popular. Every day, uncounted connected objects are integrated into our environment. Their number is growing to such an extent that billions of objects are expected in the near future [1]. Indeed, all these connected physical objects will make it possible to provide new services through sensors (temperature, motion, pollution, etc.), actuators (remotely controllable devices acting on the environment), and other entities. They can be considered as new services in a service-oriented computing vision [2]. Also, the combination of these technologies with Cloud/Fog Computing and Artificial Intelligence approaches opens up new opportunities and possibilities for designing and deploying new complex and composite application services with high added value. These new application services are complex spaces characterized by the openness, heterogeneity, uncertainty, and dynamics of the entities that constitute them [3]. Thus, these characteristics raise challenges such as uncertainty management, event responsiveness, context-sensitivity, and self-adaptation to random changes that may take place in an IoT environment. Due to

RIF'20:: The 9th Seminary of Computer Science Research at Feminine, March 07th, 2022, Constantine 2-Abdelhamid Mehri University, Algeria
EMAIL: fatima.merabet@univ-constantine2.dz (F. Merabet); djamel.benmerzoug@univ-constantine2.dz (D. Benmerzoug)

this dynamism and all these environmental changes, the system's functioning will become more complex, and the quality of service will be uncertain.

One promising way to reduce these problems is self-adaptation. Self-adaptation has emerged as a methodology to enable software systems to autonomously respond to the changing environment to maintain the required QoS [4]. That means systems can detect changes and then act correctly while satisfying user goals and QoS. The question to be answered is how IoT systems respond to environmental changes and user goals during execution while keeping their behavior and overall functioning correct.

A self-adaptive system (SaS) is capable of automatically modifying itself in response to changes in its operating environment [5,6]. This modification is done by adjusting attributes or artifacts of the system in response to changes in the system itself or its environment. In recent years, SaS has seen increasing interest in different research areas, such as Pervasive Computing, Autonomic Computing [6], and Nature-Inspired (Organic) Computing [7].

This paper makes the following contributions according to the BPMN life cycle, as shown in Figure 1:

- A BPMN extension for Self-Adaptive Systems: The adaptation is guaranteed at the time of execution only if we reflect on it from the conceptual level. Our extension permits us to collect as much data as possible on the processes to simplify their adaptation during execution. The proposed extension is illustrated through an example dealing with accident management in a smart city system.
- A global architecture for dynamic adaptation of service-based IoT applications: The second contribution concerns the optimization phase, where we propose an architecture for dynamic adaptation of service-oriented systems inspired by self-adaptive systems. Our architecture can control and adapt composite services, taking into account the user context on the one hand and the variable preferences of users on the other.

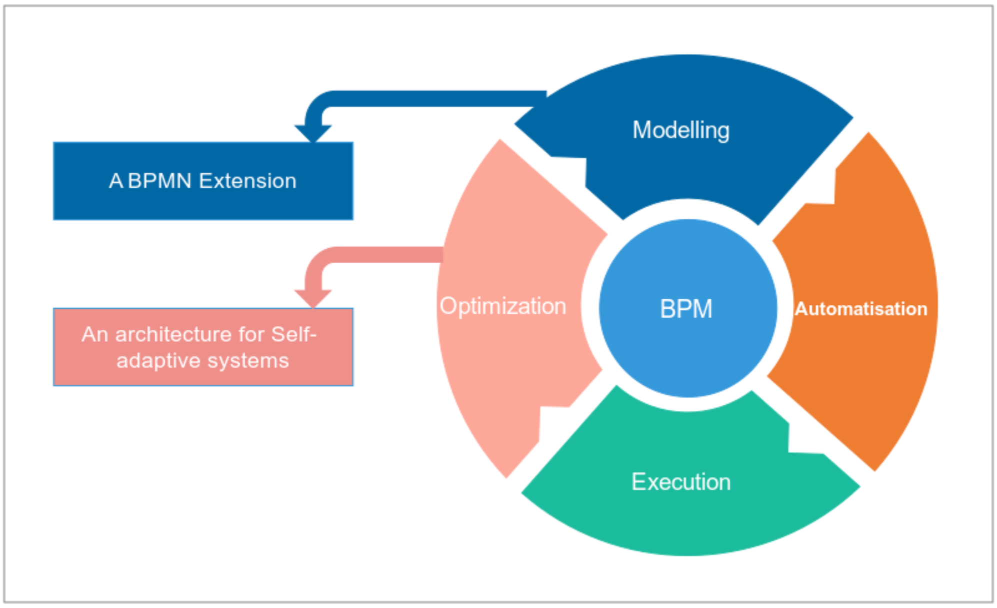

Figure 1: Our proposed contributions according to the BPM life cycle.

The remaining sections of this paper are structured as follows. Section 2 presents some related work. Section 3 discusses our first contribution, which is an extension of BPMN for self-adaptive systems. A global architecture that represents the second contribution is presented in section 4. Finally, Section 5 concludes this paper and presents future directions.

2. Related Work

Adaptation is an important requirement for many software systems, such as cloud-based, cyber-physical, and mobile systems. These are designed to satisfy different user requirements and expectations, especially the need to take advantage of service permanently. Such a system should be able to adapt its structure and/or behavior at runtime to cope with changes in its operating environment and user needs [8]. In the context of self-adaptation, several works are proposed to solve the problem of environmental change and user needs. Some of these works focus on self-adaptation in the design phase. Their main objective was to provide designers with the right tools they need to model all relevant concerns in the context [13].

Some authors introduce the notion of BPMN and the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) approach, such as in [10] and [11], where an Adaptation Engine (AE) that supports self-adaptation is introduced. More specifically, they describe how autonomous managers responsible for self-adaptation of process instances implement the MAPE control loop. However, their contributions are incomplete since the recommended solution deals only with the architecture of the AE, notably the internal architecture of the autonomic managers.

In [12], the authors propose a hybrid approach for the development of self-adaptive systems that uses both the local knowledge base in the vehicle and the global knowledge base provided via a Web service. The global knowledge base is shared and evolved by multiple vehicles through the Web service.

While in [9], the authors present a conceptual SaSs modeling approach consisting of a meta-model and a modeling process. The process defines how the meta-model can be instantiated from requirements specifications to create conceptual SaSs models.

Also, in [13], the authors present a new architecture that uses context-aware management to manage the auto-adapted system and uses the intermediate context model to redefine useful data exported from the BPMN tool at design time. Its goal is to provide a solution for using specific views at design time to simplify the expert's task. Moreover, it also opens the possibility of using many independent views to separate concerns and improve responsiveness in a dynamic context-aware system.

Process self-adaptation has been the focus of several contributions, such as [15,16,17,18,19,20, 23, 24]. Each of these papers recommends the use of the MAPE-K approach for monitoring activities (sub-processes or tasks) or processes. For example, [15] recommended a solution for managing process variants at design time and self-adjusting them at run-time. At design time, the solution makes it possible to model process variability by describing process variants.

Of all the previous works, some authors focused only on the self-adaptation technique like in [10,11,12,9,13]. However, to make an efficient self-adaptation, it is necessary to consider criteria defined at design time in the business process activities during its execution.

Other than that, other work, like in [14], proposed an extension to the BPMN language standard that allows specifying the requirements of business process (BP) activities in terms of security, compliance, cost, performance, and storage. However, they were not discussing the adaptation mechanism, although this is especially necessary for a dynamic environment like the IoT, where there are many changes to which the system must respond in a self-adaptive manner.

The adaptation is guaranteed at the time of execution only if we reflect on it from the conceptual level. Thus, in this paper, we have proposed two complementary contributions according to the BPM life cycle.

• The first one represents a BPMN extension named BPMN4SAS, whose goal is to collect as much data as possible on the processes to simplify their adaptation during execution. So, we think of adaptation at the conceptual level.

• The second one represents a global architecture of a self-adaptive system. This architecture can control and adapt composite services, considering the user context on the one hand and the variable preferences expressed by users on the other hand.

Our contributions will be discussed further in the following sections (see sections 3 and 4).

3. BPMN4SAS: A BPMN extension for Self-adaptive System

Through an illustrative example, our BPMN extension for self-adaptive systems (SaS) has been demonstrated, followed by a set of BPM tools that allow its implementation.

3.1. BPMN extension principle for Self-adaptive System

Modeling different criteria of such an IoT system is not a simple task because of several factors that will influence the overall behavior of these systems. Our BPMN4SAS extension is based on previously defined and perfectly measured criteria to optimize the overall performance of the system. BPMN4SAS explains the different criteria that must be taken into account to make the system adaptable to change since the IoT world is ubiquitous and requires self-adaptation to the context during execution. This extension was introduced to provide a standard visualization mechanism for designing, controlling, monitoring, and fluidly communicating ubiquitous systems. The extension mechanism offered by BPMN ensures the validity of the basic elements as well as the integrity of the concepts, adding to the specificity of the domain. This is an abstract structural representation that is used to define the different components of the BPMN 2.0 language. It is presented by a class diagram that has clear and well-defined semantics and takes into account the nature of the BPMN 2.0 language [22].

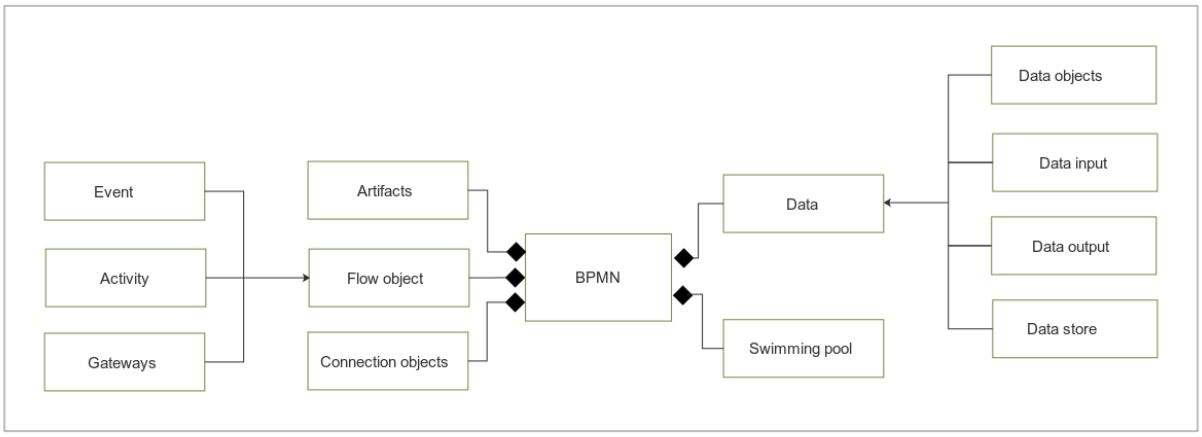

Figure 2: Meta model for structuring BPMN language elements

3.2. QoS criteria used in BPMN4SAS extension

In our BPMN4SAS extension, a business process is seen as a set of activities organized over time that produce a specific result. To simplify the adaptation of a process, our extension allows each process activity to explicitly represent the following criteria:

Table 1 QoS criteria used in the BPMN4SAS extension

ID	QoS criteria	Description	Unit
1	Response time	Specifies the estimated delay time required to send a request and receive a response.	Ms
2	Availability	The availability of a service is the probability that this service is active. It can be expressed by the ratio: active period/period total. The active period represents the time when the service is operational and responding to customer requests. The total period (active period + inactive period) is the time during which the customer wants the service to be active.	%
3	Reputation	Is a measure of the credibility of the service. It mainly depends on the direct or indirect end user's experience.	
4	Latency	Represents the time between sending and receiving the request.	Ms
5	Cost (or price)	Refers to the price that the customer must pay to the service provider for each invocation of the service. This parameter often occurs in commercial web services.	$

The QoS criteria are defined for all activities, and their values change according to the needs and priorities of each activity.

3.3. A Meta-Model of our extension

The BPMN specification model does not provide enough graphical notations for the representation of extensions. Accordingly, we completed the model by adding new graphical elements to the modeling tool as shown in figure 3. We have added new concepts related to our domain (QoS criteria are taken into consideration in our extension). The new BPMN extension must take into consideration the rules to make the complete design [22]:

• The basic flow elements (Activity, Event, and Gateway) must not be modified.
• It is not possible to add new flow elements.

Figure 3 shows the meta-model of our extension proposed by integrating the new criteria, which are performance, availability, security, cost, run-time, latency, and stability. All these criteria are linked to the entity: Activities

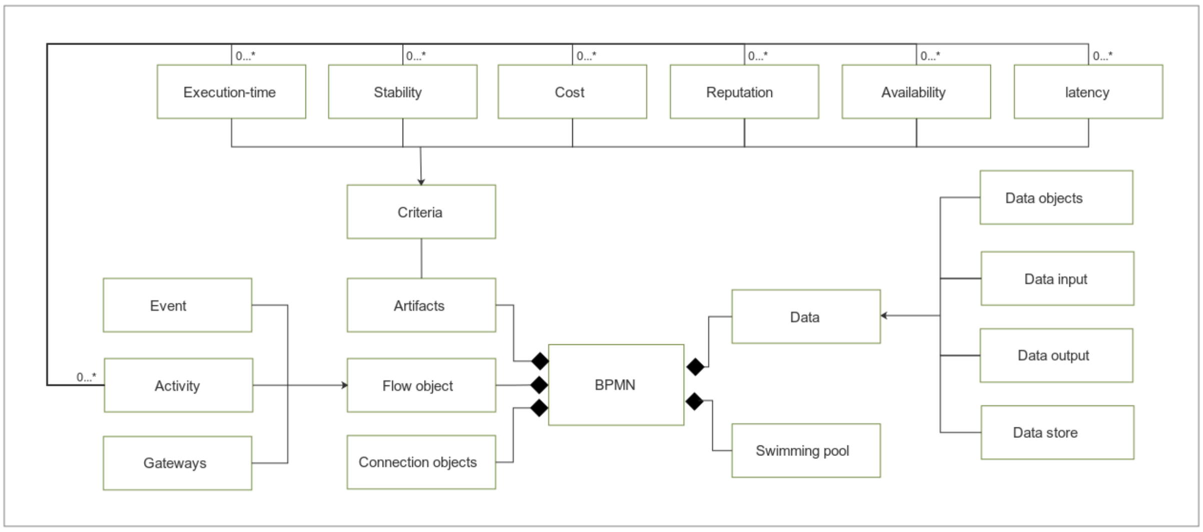

Figure 3: Meta-model for structuring BPMN4SAS extension elements

3.4. Illustrative example

In order to illustrate our extension, we will apply it to a real example from the IoT field; we chose accident management in a smart city system to explain it. We show the modeling of this example with the BPMN standard by specifying the different activities that must be carried out with the QoS constraints that must be respected in our proposed BPMN4SAS extension.

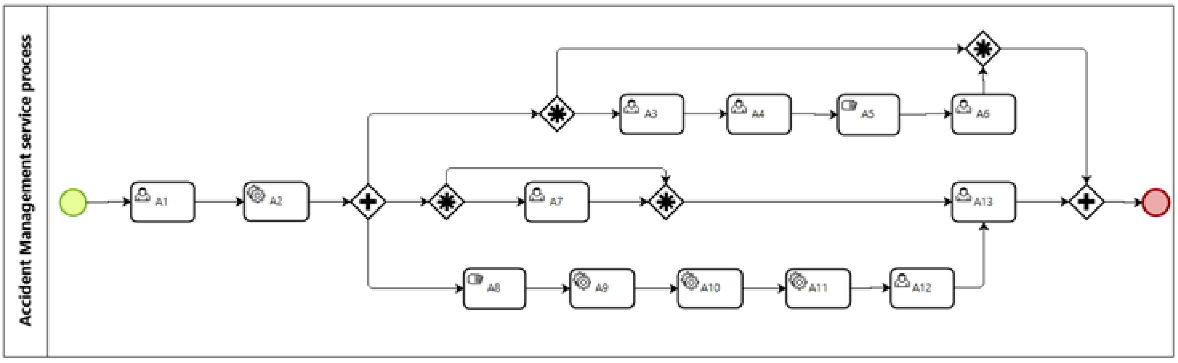

Figure 4: Accident management service process modeling with BPMN

Figure 4 shows the modeling process "Accident Management service process" with the BPMN standard: Here are the numbers and the description of the different activities in this example:

A1: Collect the photos of the accident.
A2: Analyze the recovered images.
A3: Transport the injured people.
A4: Make a preliminary diagnosis and send it to the medical establishment.
A5: Taking care of the wounded.
A6: Prepare a medical report and send it to the police automatically.
A7: Extinguish the fire.
A8: Involve law enforcement and investigation.
A9: Collect information from witnesses and all drivers in the accident area.
A10: Collect information about the road's state.
A11: Collect meteorological information.
A12: Establish an automatic accident report.
A13: Remove damaged cars.

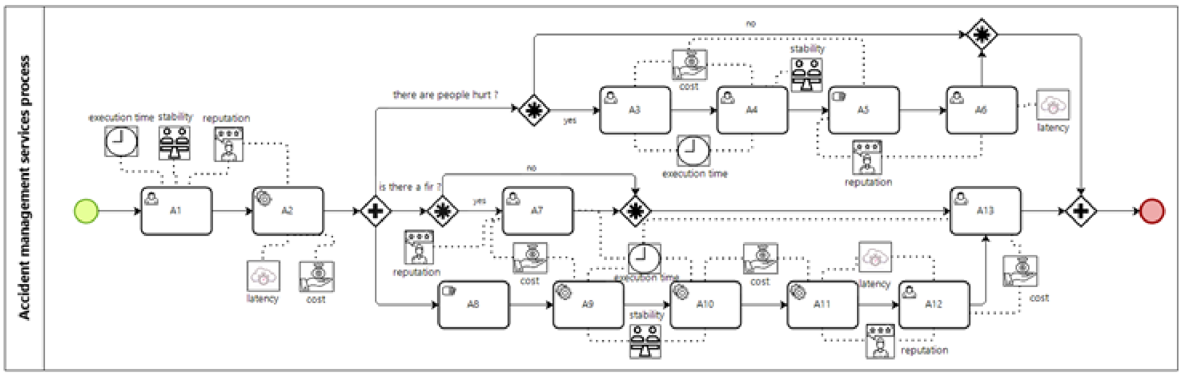

Figure 5: Accident management service process modeling with BPMN4SAS extension

In this example, availability applies to all services, so we have not included it in the example in Figure 5.

Each processing activity has its own specific QoS properties. For example, activity A1 requires a high-performance service with a good run-time to carry out the tasks under the right conditions. However, activity A3 requires a security criterion to ensure the confidentiality and protection of the users' rights to achieve a successful conclusion.

4. A global architecture for dynamic adaptation of service based IoT application

In this section, we present a general architecture with essential components designed to build a self-adaptive system with high performance.

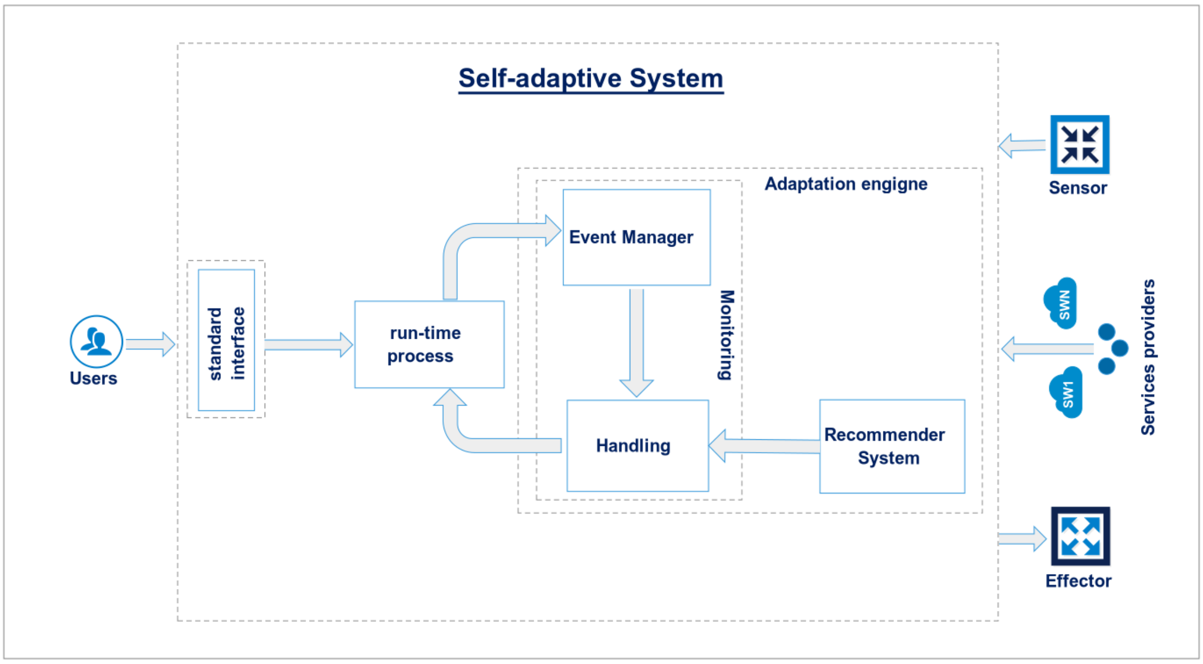

Figure 6: The proposed global architecture for self-adaptive service-oriented systems

4.1. Overview of the proposed architecture

The general architecture gives us a global vision of the organization, role, and function of the different modules in our system. It supports the self-adaptation of running micro services. First, we will define these different modules and describe the interactions between them. As shown in Figure 6, our self-adaptive system can be conceptually divided into two main modules: the runtime process (that represents the managed system) and the adaptation engine (that represents the system's manager). The structure of our architecture also includes a communication interface with users and services, sensors, and effectors.

Standard interface user: it is the interface that allows users to access the running process. It acts as an intermediary between them. Thanks to it, the users can invoke Web services according to the system's needs. It is the mediator between users and the system.

• Sensors: these are the terminals that provide services to users. Also, the system uses them to collect information about its context to make decisions about this information.

• Effectors: devices in which the system applies its execution results.

• Users: people who invoke internal and external services to use them in the runtime process activities.

• Service providers are people who offer services in addition to those available across a variety of devices.

• Run-time Process module: This module represents the run-time system managed by our adaptation engine module.

• Adaptation engine module: This module with the run-time process represents our self-adaptive system that provides a system that can repair itself if any execution problems occur, and complete its execution while respecting QoS requirements. It is structured into two sub-modules, which we will explain in the next sections (see section 4.1.1).

4.1.1. Adaptation engine module structure

In this section, we focus on the adaptation engine structure module, which is the kernel element in our architecture. It enables dynamic control and adaptation and contains two sub-modules.

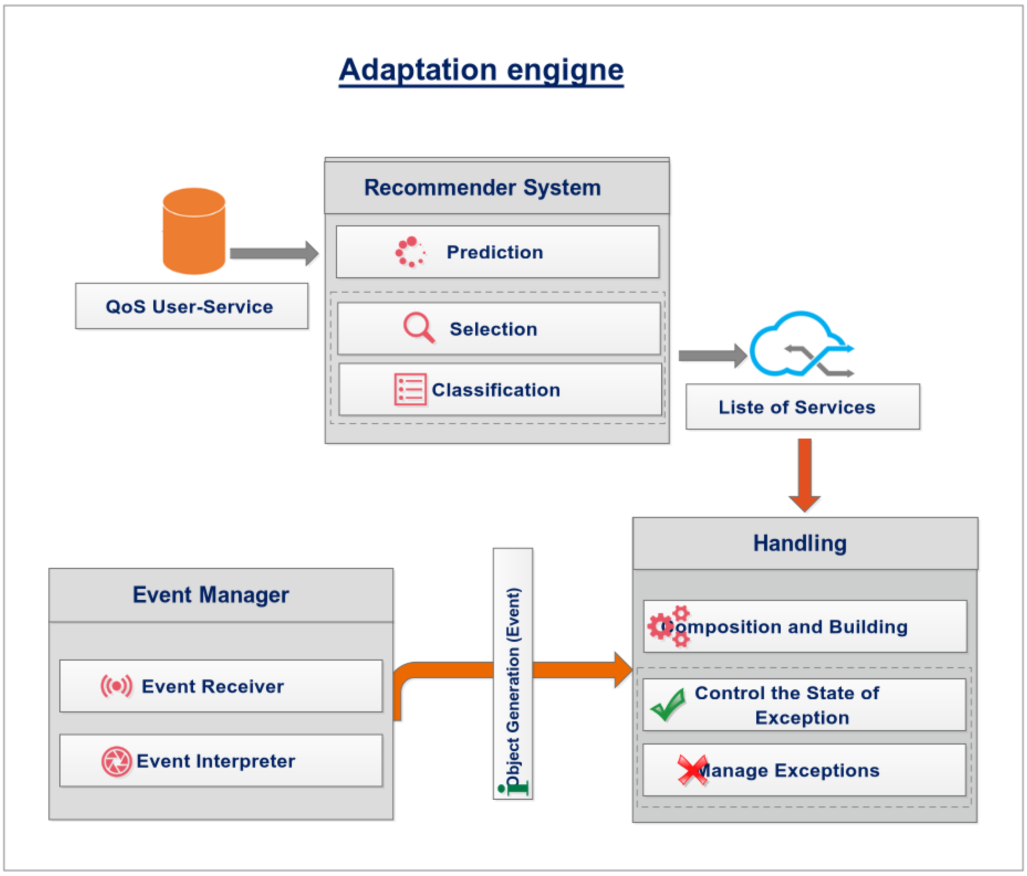

Figure 7: Structure of Adaptation engine module

As shown in figure 7, the "adaptation engine" module contains two sub-modules, and each module has several components, each of which has a specific function.

A. Monitoring: it contains components that lead to the monitoring of the system, so it acts as a controller of the dynamic adaptation of systems. So its primary goal is to collect services coming from different sensors and several providers. Then predict their quality of service to classify them into clusters. Hence, users can easily select the best services they need and then compose them to make a perfect system. It is a middleware between the runtime process and the recommended system. Its principal role is to monitor the runtime process to ensure its self-adaptation and then maintain the list of services recommended by the recommender system. If one of the currently working services is about to fail or its quality of service degrades, the monitor should re-compose it with the appropriate replacement service from the list. It contains two sub-modules that we will explain below.
Event Manager Module: this module acts as an event handler for events received from the running process. The received events are captured by the first component of the event handler (the event receiver). Then, after receiving the event, there is another "interpreter" component that interprets it to determine its nature. Finally, an event object is generated to determine its necessary information. This entity takes the directions to another module called the handling module.

Handling This module analyses the event objects generated by the previous module (Event Manager) to ensure the safe end of the various activities of the running process. Its role is to check if there is an overflow during the execution of the process. It has two steps:
• **Control the State of Exception:** this module examines the state of an event object received by the Handling module and supports the following decisions:

- If the captured object indicates in its state that there is good progress in the process, it is then a safe state (no exceptions, no errors...).
- If the captured object reports an error, in this case, it will be sent to the component that handles exceptions (Manage Exceptions).

• **Manage Exceptions:** it uses necessary functions and procedures to solve the system's problems during the execution step of the process. These models should always be considered as a guide during the execution step of the process.
• **Composition and Building:** this component uses the services recommended by the recommended system component to replace one service with another at run-time when a service fails or a new, more efficient one is available, while the goal is to build a functioning system and guarantee its correct evolution.

B. Recommender System: The increasing number of services on the Internet makes it hard for a user to select a suitable Web service from a large set of service candidates to use it in its self-adaptive system. Therefore, to solve this problem, we propose a novel recommendation system for Web services that helps users select the best one. Our recommender system first collects services from different sources, stores their QoS values in a historical dataset, and finally recommends lists of the best services to users. It is composed of three steps:
• **Prediction**: in a real scenario, the number of services invoked by a user is quite limited, which leads to a large number of missing QoS values for these services. So we need to predict these missing QoS values based on the historical QoS records of services invoked by users. This step aims to predict the missing QoS values of services to facilitate their selection and recommendation.
• **Selection**: after the prediction of QoS values, users aim to select the best services adequate to their needs. However, an inappropriate service selection may cause many problems (e.g., suit the performance of the process resulting). So, we propose in this step algorithms of selection to help users select services with optimal QoS values.
• **Ranking**: in this step, we assemble services from the previous step into lists, where each list contains services with the same QoS values. We classify them according to their QoS attribute values, taking into account the constraints and the degree of importance of each service quality criterion specified by our BPMN extension.

User-service QoS records represent the interaction between users and the services they invoke. It contains historical records of quality of service values.
List of Web services is the result of the recommended services. It represents the clusters of services selected and classified according to their QoS values, which are recommended for use in the composition and building steps.

5. Conclusion

In this paper, two complementary contributions have been proposed to enable the dynamic adaptation of systems in an IoT environment. These contributions were carried out in a service-oriented framework, following two essential aspects of the business process lifecycle. First, we presented a BPMN extension to manage adaptation from the earliest modeling phase, whose objective is to collect the maximum amount of data possible in the process and to simplify its adaptation during execution. In the second step, we presented a general architecture in the optimization phase to adapt applications during execution. This architecture has essential components that can be designed to build a modular self-adaptive system with high performance. Each module is based on several interconnected components that are involved in a collaborative process to achieve the specified objectives when detecting events in an IoT environment.

In future work, we aim to refine our current architecture by taking each component and interpreting it with machine learning techniques. By introducing the most important criteria presented in our BPMN extension, we aim to improve the prediction of QoS values by introducing the most important criteria presented in our BPMN extension, to facilitate the selection and classification of services into clusters for use by users. Also, we wish to apply our architecture to a real-life example of the Internet of Things field.

6. References

[1] G. Garzone, Approche de gestion orientée service pour l'Internet des objets (IoT) considérant la Qualité de Service (QoS), Ph.D. thesis, Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse), 2018.
[2] S. Cherrier, Y. M. Ghamri-Doudane, The "object-as-a-service" paradigm, in: 2014 Global Information Infrastructure and Networking Symposium (GIIS), IEEE, 2014, pp. 1–7.
[3] A. Yachir, Composition dynamique de services sensibles au contexte dans les systèmes intelligents ambiants, Ph.D. thesis, Université Paris-Est Créteil Val-de-Marne - Paris 12 - France, 2014.
[4] A. Elhabbash, M. Salama, R. Bahsoon, P. Tino, Self-awareness in software engineering: A systematic literature review, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 14 (2019) 1–42.
[5] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, A. L. Wolf, An architecture-based approach to self-adaptive software, IEEE Intelligent Systems and Their Applications 14 (1999) 54–62.
[6] J. O. Kephart, D. M. Chess, The vision of autonomic computing, Computer 36 (2003) 41–50.
[7] C. Müller-Schloer, H. Schmeck, T. Ungerer, Organic computing—a paradigm shift for complex systems, Springer Science & Business Media, 2011.
[8] S. Kallel, I. B. Rodruigez, K. Drira, Editorial adaptive and reconfigurable software systems and architectures, 2016.
[9] J. P. S. da Silva, M. Ecar, M. S. Pimenta, F. N. Kepler, G. T. Guedes, C. M. Betemps, Improving self-adaptive systems conceptual modeling, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, 2018, pp. 1292–1299.
[10] J. Oukharijane, I. B. Said, M. A. Chaabane, E. Andonoff, R. Bouaziz, A new adaptation engine for self-adaptation of bpmn processes instances (2019).
[11] J. Oukharijane, I. B. Said, M. A. Chaabane, E. Andonoff, R. Bouaziz, Towards a new adaptation engine for self-adaptation of bpmn processes instances, in: Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, SCITEPRESS-Science and Technology Publications, Lda, 2019, pp. 218–225.
[12] D. A. Kafaf, D.-K. Kim, A web service-based approach for developing self-adaptive systems, Computers & Electrical Engineering 63 (2017) 260–276.
[13] S. Lavirotte, G. Rey, N. Le Thanh, J.-Y. Tigli, et al., From bpmn to live application: How the context can drive an auto-adapted system, 2019.
[14] L. Fadila, Z. Karim, B. Djamel, Modelling business processes for outsourcing into the fog and cloud computing, Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain (2018).
[15] C. Ayora, V. Torres, V. Pelechano, G. H. Alférez, Applying cvl to business process variability management, in: Proceedings of the VARiability for You Workshop: Variability Modeling Made Useful for Everyone, 2012, pp. 26–31.
[16] K. Oliveira, J. Castro, E. Santos, R. Fidalgo, S. España, O. Pastor, A multi level approach to autonomic business process, in: 2012 26th Brazilian Symposium on Software Engineering, IEEE, 2012, pp. 91–100.
[17] K. Oliveira, J. Castro, S. España, O. Pastor, Multi-level autonomic business process management, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2013, pp. 184–198.
[18] S. K. V. Ferro, C. M. F. Rubira, An architecture for dynamic self-adaptation in workflows, in: Proceedings of the International Conference on Software Engineering Research and Practice (SERP), The Steering Committee of The World Congress in Computer Science, Computer . . . , 2015, p. 35.
[19] R. Seiger, S. Huber, P. Heisig, U. Assmann, Enabling self-adaptive workflows for cyberphysical systems, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 2016, pp. 3–17.
[20] R. Seiger, S. Huber, P. Heisig, U. Aßmann, Toward a framework for self-adaptive workflows in cyber-physical systems, Software & Systems Modeling 18 (2019) 1117–1134.

[21] M. Salehie, L. Tahvildari, Towards a goal-driven approach to action selection in self-adaptive software, Software: Practice and Experience 42 (2012) 211–233.
[22] S. A. White, C. Bock, BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Management Notation, Future Strategies Inc., 2011.
[23] Alfonso, Iv{\'a}n and Garc{\'e}s, Kelly and Castro, Harold and Cabot, Jordi, Self-adaptive architectures in IoT systems: a systematic literature review, Journal of Internet Services and Applications, Springer Open vol 12 (2021), nb 1, 1–28.
[24] Lee, Euijong and Seo, Young-Duk and Kim, Young-Gab, Self-adaptive framework based on MAPE loop for Internet of things, sensors: Multidisciplinary Digital Publishing Institute vol 19, nb 13, 2019, 2996.