Deep Attention on Measurable and Behavioral-driven Complete Service Composition Design Process

Ilyass El Kassmi[1], Radia Belkeziz[2], Zahi Jarir[3]
LISI Laboratory, Computer Science Department
Faculty of Sciences, Cadi Ayyad University, Marrakech, Morocco

***Abstract*—The web service technology has still proved its effectiveness in the digital revolution we are facing. This success unfortunately raises more and more complex obstacles, particularly related to the service composition. The integration of Non-Functional Requirements (NFRs) in each step of service composition process, starting with abstract service composition specification to the generation of the verified and concrete composed services, represents one of them. Furthermore, this complexity remains more difficult when NFRs are addressed in both quantifiable (i.e. Quality of Service) and behavioral aspects. Despite the relevant contributions present in the literature, this challenge still remains an open issue when considering NFRs modeling, publishing, integrating with each other, and handling conflicts and dependencies in the whole composition's lifecycle. As a consequence, we suggest this contribution that aims to propose an approach showing how to weave efficiently required NFRs with functional requirements in a complete lifecycle composition supporting specification, formalization, model checking verification and integration steps of desired concrete composite service. Patient Health Records in Regional and University Health Centers in Morocco is used as a case study to experiment our approach.**

Keywords*—Non-Functional requirements composition; behavioral non-functional requirements; quantifiable non-functional requirements; model checking; web service composition formalization

I. Introduction

In the digital revolution we are facing, Web Service (WS) technology still proved its effectiveness. This technology is widely used to build highly advanced applications that support digital transformation, including artificial intelligence, big data, the Internet of Things, cloud computing, and other emerging technologies. Web Services are defined as loosely-coupled, distributed processes that communicate over a network to perform a specific task and to facilitate interoperability among heterogeneous systems. They can be autonomously developed, decentralized and independently deployable, built and integrated by composition processes to fulfill complex requirements. These requirements, known in software engineering by properties or concerns, are classified mainly into two main classes: Functional requirements (FRs) and Non-Functional Requirements (NFRs). Functional Requirements specify what business-related goals the service composition should achieve. Whereas Non-Functional Requirements define how these services are supposed to fulfill their goals in term of performance and other quality constraints, mainly known as quantifiable QoS properties (e.g. availability, reliability, etc.).

In the software engineering literature, specifically in the service-oriented architectures (SOA), different definitions and classifications of NFRs can be found [1]. We can notice that most of contributions addressing NFRs are focusing on QoS attributes. These attributes describe mainly quality aspects of published services such as availability, cost, response time, reliability, performance, etc. An interesting work classified and analyzed each of 530 studied attributes extracted from 11 industrial requirements specification [2]. The aim of this work is to determine if the NFRs can be really considered as non-functional requirements, or simply be approached as behavioral aspects that can be treated in the same way as the functional requirements. Until now, less efforts are deployed to address unquantifiable requirements in web service composition process. In fact, providing a complete service composition process that details NFR quality-oriented and behavioral integration from specification, modeling, and verification to the composition is always a fastidious task and still an open issue. This difficulty comes from the fact that the web service composition is closely linked to other challenges such as discovery and selection of the most appropriate services, implementing FRs or NFRs, the verification of feature interactions between the non-functional properties of a specific functional service, etc.

Before integrating NFRs with each concerned FR, there is a clear need to specify and formalize them correctly. Some interesting surveys outlined the most used formalization method including Automata, Process Algebra, Petri Nets, etc. [3]. Once NFRs are formalized, some algorithms and techniques are then required to combine them seamlessly with associated FRs to avoid any feature interaction.

To enable our approach to meet the majority of needs in terms of modeling and implementing a complete service composition design process, we are convinced that the use of automata is a better method due to the advantages and the simplicity they offer and also satisfactory results obtained during our previous contributions [4][5]. Therefore, in this contribution we propose an automata modelling approach for Functional and Non-Functional Requirements aimed at providing expert users with increased flexibility to design and integrate numerous complex behavioral NFRs, such as security attributes (e.g. authentication, access-control methods, encryption, etc.), to others custom business-related behavioral NFRs. A varied choice of QoS oriented properties is also integrated in our approach to help selecting the optimal service composition based on attributed weights for each property.

In this perspective, we suggest in this paper a contribution having the advantage of:

- Handling quantifiable and behavioral NFRs using automata-based modeling.
- Publishing, discovering and selecting services implementing behavioral NFR.
- Providing support for composing NFRs with FRs.
- Performing a QoS-driven selection for quantifiable NFRs to generate the best matching service composition.
- Proposing a model checking verification to validate the proposed composition.

The remainder of the paper is organized as follows. Section 2 exposes some interesting contributions tackling service composition integration with NFRs. In Section 3 we present an analysis of QoS-oriented NFR integration to the service composition. In Section 4 we project our contribution to integrate behavioral NFRs to the composition process. In Section 5 we present a novel approach handling integration of both quantifiable and behavioral NFRs to the service composition, whereas Section 6 presents a case study to demonstrate the behavior of this approach. Finally, we conclude by summarizing suggested approach and highlighting future works and upcoming perspectives on Section 7.

II. Related Works and Motivation

The service composition is a widely explored topic. A considerable amount of literature has been published tackling different aspects, problems and perspectives from design time to execution including and not limited to modeling, formalization, discovery, NFR integration, selection, optimization, verification and code generation. In this section we present some interesting contributions addressing this challenge, their limits and similarities with our approach.

In order to conduct and classify the main contributions and provide their motivations in more details, we define a list of research guidelines (RG) as follows:

- RG1 – Are both FR and NFR modeling included in the service composition process?
- RG2 – Does the service composition integrate quantifiable NFRs?
- RG3 – Does the service composition integrate behavioral NFRs?
- RG4 – Is there any validation of the overall behavior of the composed service?
- RG5 – Does the service composition process allow the service publication and discovery based on behavioral NFRs?
- RG6 – Does the service composition provide the optimal service selection based on quantifiable NFRs?
- RG7 – Does the service composition support multiple behavioral NFRs integration applied to the same autonomous service?

Chen et al. proposed in [6] an approach allowing to compose services addressing QoS attributes and dependencies. This work consists of performing a goal softening to reduce the candidate using Pareto techniques combined with Vector Ordinal Optimization in order to find Pareto Optimal Solutions, by considering multiple QoS dependencies criteria to prune uninteresting candidates. Deng et al. proposed in [7] a Correlation-Aware Service Pruning method that improves the QoS of the generated sequential service composition by taking QoS correlations into account in the service selection process. This proposed method is based on a preprocessing algorithm for candidate services to remove irrelevant services. Then a service selection with correlation in adjacent or not adjacent tasks is performed step by step for each task in the service plan to compose the optimal composite services and prune services that are concluded not optimal. In [8], authors proposed a contribution performing exploration of cloud services and returning the optimal solution based on QoS parameters using Eagle Strategy with Whale Optimization Algorithm (ESWOA). According to presented experimentation, the proposed approach got better results compared to other optimization algorithms such as Genetic Algorithm (GA), Hybrid Genetic Algorithm (HGA), Whale Optimization Algorithm (WOA). Y. Liang et al. proposed in [9] a QoS-aware automatic service composition based on QoS correlations between services. They proposed a preprocessing algorithm to address the available services on the pool and generate a service dependency graph. The experimental results are compared to the approach in [10] proposed by Feng et al., which used a method that dynamically refines the composed workflow considering the QoS dependencies, user-provided constraints and QoS constraints. These two approaches offer significant improvements in performance dealing with QoS dependencies. The work in [11] presented by Jatoth et al. proposed a MapReduce-based Evolutionary Algorithm with Guided Mutation MR-EA/G in order to compose Big services with better performance, considering five QoS attributes: price, throughput, availability, reliability and response-time. Jin et al. proposed in [12] a service description modeling associated with a service correlation mapping allowing to get the QoS values of described services automatically. They highlighted the result of comparing results obtained by their proposed approach for candidate service search for the selected QoS parameters: time, cost, availability and reliability against the traditional Genetic Algorithms. Liang et al. proposed another approach in [13] which aims to handle QoS inter-service correlation using Double Information based Cooperative Coevolutionary Algorithm. They use Potter's cooperative coevolutionary framework and provide both local and global knowledge for the dynamic service selection optimization. Wang et al. proposed a Q-Graphplan approach in [14] to solve the QoS-aware automatic service composition problem with multiple QoS criteria constraints. The optimal solution is extracted from the path generation graph using a backward A* algorithm with the heuristics of the planning graph. The experiment is conducted according to six QoS criteria (response time, price, latency, availability, successful rate, and reliability).

As presented above most cited contributions tackling the integration of NFRs into the service composition focus only on measurable QoS NFRs. Behavioral NFRs are not widely explored, and are commonly restricted to specific security attributes. Also, a verification phase to validate the conformance of constructed composition is often omitted.

Since our objective in this article is to focus on both measurable and behavioral NFRs in a complete service composition process, the rest of this section will be dedicated to present some interesting and similar contributions addressing the same objective.

Lu et al. proposed a model-checking based approach in [15] to verify the satisfaction of behavior-aware privacy requirements in services composition. They used extended interface automata for modeling BPEL process, including a support for privacy semantics. The proposed approach consists on extracting Linear Temporal Logic (LTL) specification from behavioral constraints, but limited to privacy requirements. These specifications are transformed to Promela description in order to allow a model-checking based verification using SPIN. Dou et al. presented in [16] an enhanced version of their proposed method implementing k-means algorithm to ensure privacy-aware cross-cloud service composition based on QoS history records. Souri et al. proposed in [17] a formal verification approach to tackle cloud service composition problem in the multi-cloud environment in order to decrease the number of cloud providers and obtain optimal results according to QoS parameters. The presented approach proposed a behavioral modeling using a Multi-Labeled Transition Systems (MLTS)-based model checking and Pi-Calculus-based process algebra methods for monitoring functional and non-functional requirements.

Most of proposed approaches are focusing their contribution on adding a specific security layer to the composite service, and consequently ensuring the satisfaction of some security attributes additionally to commonly explored QoS properties. In other hand, Brucker et al. proposed in [18] a framework for modeling, validating and composing secure services. The approach uses a BPMN based modeling to design the user's functional need and implement the desired security properties based on ConSpec formalization. The overall framework allows different actors to collaborate starting from requirements definition, modeling, security planning, security validation then generating the secure service composition. The framework supports three non-functional properties which are encryption, cost and availability in order to rank discovered services based on attributed weights.

Table I presents a summary of the above cited contributions according to raised research guidelines. We notice that the focus is mainly conducted to the integration of quantifiable quality-oriented NFRs. Behavioral NFRs (e.g. security attributes) are either neglected, or conducted separately for each security property (e.g. privacy, integrity, encryption, etc.).

This survey incorporated our previous contribution to the proposed classification which consisted of a verification module to validate the correctness of the composition of the designed service. This prompted us to improve the validation of the service composition obtained and to enrich it with appropriate formalization and algorithms, taking into account the specifications of quality and behavioral NFR integrations. This survey incorporated our previous contribution to the proposed classification which consisted of a verification module to validate the correctness of the composition of the designed service. This prompted us to improve the validation of the service composition obtained and to enrich it with appropriate formalization and algorithms, taking into account the specifications of quality and behavioral NFR integrations. Thus, the integration of both behavioral NFRs and quantifiable quality-oriented NFRs in more fine-grained analysis is still an open challenging issue. This motivated us to suggest an approach to tackle the issue of integrating both behavioral and quality-oriented NFRs in the service composition context. Another motivation comes from the work of Rai and Gangadharan that presented a survey consisting on classifying approaches tackling the model checking based verification of web service composition [19]. This survey incorporated our previous contribution to the proposed classification which consisted of a verification module to validate the correctness of the composition of the designed service. This prompted us to improve the validation of the service composition obtained and to enrich it with appropriate formalization and algorithms, taking into account the specifications of quality and behavioral NFR integrations.

TABLE I. CATEGORIZATION OF CITED CONTRIBUTIONS ACCORDING TO RESEARCH GUIDELINES

	[6] Chen et al.	[7] Deng et al.	[8] Gavvala et al.	[9] Liang et al.	[10] Feng et al.	[11] Jatoth et al.	[12] Jin et al.	[13] Liang and Du	[14] Wang et al.	[15] Lu et al.	[16] Dou et al.	[18] Brucker et al.
RG1	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RG2	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓
RG3	-	-	-	-	-	-	-	-	-	✓	✓	✓
RG4	-	-	✓	-	-	-	-	✓	-	✓	-	✓
RG5	-	-	-	-	-	-	-	-	-	-	-	✓
RG6	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓	✓
RG7	-	-	-	-	-	-	-	-	-	-	-	-

III. Deep Analysis on Service-Oriented Behavioral Non-Functional Requirements

The literature reveals an increasing attention to quality properties when dealing with NFRs in web service context. However, there are some quality properties that cannot be quantifiable using QoS metrics, e.g. security-oriented properties. Additionally, these properties are not fulfilled with a common behavior, but instead, it may differ from a use case to another. These properties are denoted as "Behavioral NFRs". Behavioral NFRs are defined as rules, policies or restrictions applied to an abstract service. They aim to integrate specific behaviors before or after the execution of the services they are associated to. Behavioral NFRs integration change depending on the use case. For instance, authentication and access control attributes can be implemented using different methods and schemes, depending on the user's perspectives and goals. Consequently, unlike quantifiable quality attributes behavioral NFRs integration need a complete understanding of the context, and require a detailed modeling to express in an accurate way the behavioral interactions with collaborative services.

In the literature, studies have suggested different methods to tackle modeling and formalization of NFRs in the context of web service composition such as Process Algebra, Finite State Automata and Petri Nets [20]. Other contributions opted for BPMN as a modeling method for aspect-oriented service composition [21] due to its exhaustivity and expressiveness. In our approach and in order to help ensuring a rigorous composition fulfilling the designer's requirements, we aim to use a Finite State Automata (FSA) based modeling. Using FSA allows us to extend its formalization to meet our requirements and to proceed to a model checking phase to verify the correctness of designed models according to user's properties. In the service composition context, FSA allows a rich description of services and their interactions. The modeling phase consists on describing three different sets of requirements: (1) Functional requirements, which are the main business-oriented goals required by the end user. They are translated into an abstract functional automaton (AFA) defining the main functional process describing the interactions between contributing abstract services, (2) Behavioral NFRs representing the desired constraints, policies or restrictions applied to contributing services, and (3) Measurable quality-oriented NFRs dealing with QoS preferences to fit, in order to build the optimal composition. Designing all these NFRs together produces an Abstract Service Composition Automaton (ASCA), which groups all behavioral and quality-oriented scopes applied to the primary AFA.

Definition 1: An Abstract Service (AS) is a service mold, allowing to group a set of desired functionalities (goals) as functional queries. These functional goals need to be fulfilled by some potentially adapted concrete services.

Definition 2: Abstract Functional Automaton (AFA) is a septuple AFA = (S, s0, Sf, T, RF, RB, RQ), where:

- S is a set of states, $s_0 \in S$ is the initial state, $S_f \subseteq S$ is a set of final states.
- T is a set of transitions where $S \times T \times S$ is the transition relation, graphically denoted as $s^{src} \rightarrow^{t} s^{tar}$, which means that the transition t changes the state from the source AS state s^{src} to the target AS state s^{tar}.
- R_F, R_B and R_Q express respectively the sets of Functional Requirements $fr_{j,}$ Behavioral Requirements and Quality Requirements associated to abstract services. To get the set of functional, behavioral NFRs and quality NFRs for a defined abstract service we use respectively the functions *functionalReq(as), behavioralReq(as)* and *qualityReq(as)* as follow:

functionalReq $(s_i) = \{fr_{i1}, \ldots, fr_{in} \mid fr \in Q, s_i \in S \text{ and } n \geq 1\}$.

behavioralReq$(as_i)=\{br_i \mid br \in R_B \text{ and } s_i \in S\}$.

qualityReq$(as_i)=\{qr_{i1}, \ldots, qr_{im} \mid qr \in R_Q, s_i \in S \text{ and } m \geq 1\}$.

In order to complete the modeling of the AFA, the designer proceeds to describe the NFR preferences. We use scope notations to associate NF attributes to a service or a subset of services. Behavioral NFRs are integrated to the AFA using behavioral scopes, illustrated using dotted lines surrounding the service subset.

Since behavioral NFRs constitute an additional restriction over the functional automaton, they are dealt with as a particular workflow having its own description and modeled separately as Behavioral Requirement Automata (BRA). This workflow illustrates the desired NFR behavior with respect to the same automata formalization. Each behavioral NFR is indexed using a behavioral signature [22]. A behavioral signature is a regular expression notation describing the translation of the associated BRA in summarized and verbally understandable way. BRAs are published in a Non-Functional Registry represented by a database indexed using the behavioral signatures. This registry groups all the BRAs with their associated URIs corresponding to associated published concrete services. In Fig. 1, "UHCAuthentication. UHCAccessControl" is an example of a behavioral signature outlining the behavioral scope associated to the abstract service AS2'. This regular expression is summarizing the desired behavior including both Authentication and Access Control. Each behavioral scope in the modeling phase corresponds to an atomic or composite service that needs to be integrated to the current functional composition.

Definition 3: Behavioral Requirement Automaton (BRA) is a quadruple BRA = (S, s0, Sf, T, BS), where:

- S is a set of states, $s_0 \in S$ is the initial state, $S_f \subseteq S$ is a set of final states.
- T is a set of transitions where $S \times T \times S$ is the transition relation, graphically denoted as $s^{src} \rightarrow^{t} s^{tar}$, which means that the transition t changes the state from the source AS state s^{src} to the target AS state s^{tar}.

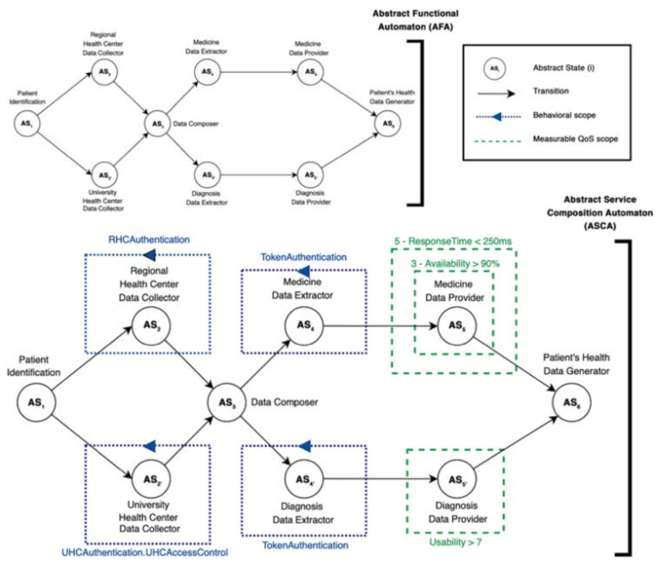


Fig. 1. An Abstract Service Composition Automaton based on the Illustrative Scenario.

In our approach, we distinguish between two types of behavioral NFRs: the pre-execution and the post-execution behavioral requirements. The pre-execution behavioral requirement is illustrated by an arrow in backward direction over the behavioral scope. It aims to be handled before the execution of the associated concrete service, whereas the post-execution behavioral requirement, illustrated by an arrow in forward direction over the behavioral scope, defines the required behavioral process to be performed after the execution of the associated concrete service. For both cases, we proceed to an automata composition allowing to merge the BRAs with the AFA. This composition process for each case is defined below.

Definition 4: Prior Execution Behavioral Automata Composition is the merging operation between the Abstract Functional Automaton AFA = (S, s0, Sf, T, RF, RB, RQ)AFA and a Behavioral Requirement Automaton BRA = (S, s0, Sf, T)BRA to fulfill the prior execution behavioral requirement associated to the state sb (i.e. having a pre-execution behavioral scope). The product is a Composition Automaton CA = (S, s0, Sf, T)CA, a quadruple described as following:

- S_{CA} is a set of states, such that $S_{CA} = S_{AFA} \cup S_{BRA}$ and $S_f \subseteq S_{CA}$.
- $s_0 \in S_{CA}$, and $s_{0(CA)} = s_{0(AFA)}$ when $s_b \neq s_{0(AFA)}$
- T_{CA} is a set of transitions where $T_{CA} = T_{AFA} \cup T_{BRA} \cup T_{AFA\rightarrow BRA} \cup T_{BRA\rightarrow AFA}$ such that:
- $T_{AFA\rightarrow BRA}$ is a set of transitions where $t_1 \in S_{AFA} \times T_{AFA\rightarrow BRA} \times S_{BRA}$ is the transition relation from the state s_{b-1} directly before s_b with the initial state $s_{0(BRA)}$, graphically denoted as $s_{b-1} \rightarrow^{t1} s_{0(BRA)}$.
- $T_{BRA\rightarrow AFA}$ is a set of transitions where $t_2 \in S_{BRA} \times T_{BRA\rightarrow AFA} \times S_{AFA}$ is the transition relation from the final states $s_{fi(BRA)}$ with the scoped state s_b, graphically denoted as $s_{fi(BRA)} \rightarrow^{t2} s_b$ such that $s_{fi} \in S_{f(BRA)}$ and $i \geq 1$.

Definition 5: Post Execution Behavioral Automata Composition is the merging operation between the Abstract Functional Automaton AFA = (S, s0, Sf, T, RF, RB, RQ)AFA and a Behavioral Requirement Automaton BRA = (S, s0, Sf, T)BRA to fulfill the post execution behavioral requirement associated to the state sb (i.e. having a post-execution behavioral scope). The product is a Composition Automaton CA = (S, s0, Sf, T)CA, a quadruple described as following:

- S_{CA} is a set of states, such that $S_{CA} = S_{AFA} \cup S_{BRA}$,
- $s_{0(CA)} = s_{0(AFA)}$, and $S_f \subseteq S_{CA}$.
- T_{CA} is a set of transitions where $T_{CA} = T_{AFA} \cup T_{BRA} \cup T_{AFA\rightarrow BRA} \cup T_{BRA\rightarrow AFA}$ such that:
- $T_{AFA\rightarrow BRA}$ is a set of transitions where $t_1 \in S_{AFA} \times T_{AFA\rightarrow BRA} \times S_{BRA}$ is the transition relation from the scoped state s_b with the initial state $s_{0(BRA)}$, graphically denoted as $s_b \rightarrow^{t1} s_{0(BRA)}$.
- $T_{BRA\rightarrow AFA}$ is a set of transitions where $t_2 \in S_{BRA} \times T_{BRA\rightarrow AFA} \times S_{AFA}$ is the transition relation from the final states $s_{fi(BRA)}$ with the state s_{b+1} directly after the scoped state s_b, graphically denoted as $s_{fi(BRA)} \rightarrow^{t2} s_{b+1}$ such that $s_{fi} \in S_{f(BRA)}$ and $i \geq 1$.

IV. Deep Analysis on Quality-of-Service Non-Functional Requirements in the Service Composition

In the literature, Non-Functional Requirements can be defined and classified in various ways depending on the context of use. Chung and do Prado Leite [23] presented different representations and classifications of NFRs. FURPS+ model is an example of classifications for software quality attributes, which illustrates a software quality tree and aims to address concerns for key types of NFRs and importantly possible correlations among them. Another model is proposed by the international standard for the evaluation of software quality ISO/IEC [24] which is a quality-oriented scheme. Its revised version in 2011 [25] proposed two main models: (1) software product quality model that groups attributes such as reliability, performance, operability, security, maintainability, etc., and (2) Quality in use model, defined using three main attributes: a) Usability in use describing the effectiveness, efficiency and satisfaction in use, b) Flexibility in use dealing with the context conformity, and c) Safety for operator, public and environment. In other hand, Galster and Bucherer [26] proposed a service-oriented taxonomy to classify NFRs. They introduced the quantifiability factor allowing to define how each attribute can be measured. This classification consists on dividing NFRs into three main classes: a) Process requirements, which are properties dealing with service design, discovery, composition and runtime, b) Service requirements, centered on the service and can be derived directly from user needs, and c) External requirements, defining the external economic or legal constraints on the development or deployment process. Authors in [27] proposed a literature review highlighting the most frequently used NFRs in service-oriented context, such as performance, reliability, usability, security, and maintainability. It aims to propose a classification based on definitions, typologies, types of systems and application domains of NFRs. Another contribution [28] proposes a detailed review classifying the NFR approaches according to different criteria, then providing a qualitative analysis of their scopes and characteristics. This work focuses on the three main classes of NFR approaches which are the Goal-oriented approaches, the Aspect-oriented approaches and the Pattern-based approaches. The work on NFRs integration to service composition has been the subject of various contributions. However, most of proposed works surrounding this topic are limited to quantifiable quality-oriented NFRs, commonly known as Quality-of-Service (QoS) attributes.

In our approach, each quality attribute is defined by a set of metrics (cf. Table II). The designer selects the appropriate quality metrics to apply to a set of abstract service. The measurement correlation expresses whether the best results are associated to higher metric value (Positive: +), or lower metric value otherwise (Negative: -).

In order to select the best matching quality-aware concrete service for a specific abstract service, we apply a weight coefficient to each desired quality metric. This coefficient helps to select the most appropriate services according to user's preferences, when dealing with multiple quality conditions associated to a common set of abstract services. The Web Service Popularity Score [29] (WSPS) was previously introduced to compute the quality measures by introducing a more appropriate and decisive factor to distinguish functionally similar services using an algorithm based on multiple criteria for multiple candidates. In our approach, this allows us to reduce the pool of candidate services by guaranteeing the satisfaction of the multiple criteria quality requirements defined by the designer. In this paper we enhance the Popularity Score metric coverage by integrating some relevant QoS metrics. The covered QoS attributes with their associated metrics and their calculation formulas are depicted in Table II.

TABLE II. TABLE OF QUALITY OF SERVICE ATTRIBUTES AND THEIR APPROPRIATE METRICS

QoS Attribute	QoS Metric	Calculation Formulae
Availability	**+ The Availability** metric (Av) is the percentage of time, in a specific time interval, during which the service can be reachable and functional. The commonly used formula uses uptime and downtime values.	$Av(s) = Uptime(s)/(Uptime(s) + Downtime(s))$ $Av(s) = MTBF(s)/(MTBF(s) + MTTR(s))$ *MTBF is the Mean Time Between Failure, and MTTR is the Mean Time To Repair.*
	- The Mean Time To Repair (MTTR) refers to the amount of time required to repair the service and restore it to full functionality.	$MTTR(s) = \sum MT(s) / MN(s)$ *MT represents the maintenance time for a specific service, and MN defines the number of Maintenance actions for that service.*
Reliability	**+ The Mean Time Between Failure** (MTBF) refers to the amount of time a service is up before it fails. It is the average (expected) time between two successive failures to reach the service.	$MTBF(s) = \sum OpT(s) / FN(s)$ *OpT represents the operational time for the service, where FN defines the number of failures actions for that service.*
	- The Failure Rate metric (FR) is the frequency with which the service fails, expressed in failures per unit of time.	$FR(ws) = FN(ws)/T$ *FN defines the number of failures actions for the service, while T defines the amount of time.*
	- The Defects per Million factor (DPM) refers to the number of defects for each million attempts of user's requests. It is defined as the ratio of the number of defects in the service to the total number of defect opportunities multiplied by 1 million.	$DPM(ws) = FReq(ws) * 1000000/TReq(ws)$ *FReq defines the number of unsuccessful (Failed) Requests, while TReq defines Total Requests performed.*
	+ Reliability (Re) refers to the service ability to function according to the agreed upon performance requirements in SLA.	$Re(ws) = [(1000000 - DPM(ws))/1000000] * 100\%$ *DPM is the Defects Per Million metric.*
Response Time	**- The Processing Time** (Proc) is the amount of time consumed for fulfilling the request by executing the corresponding functions.	$\mathrm{Proc}(\mathrm{ws}) = \sum \mathrm{ExT}(\mathrm{ws}) / \mathrm{N}(\mathrm{ws})$ *ExT defines the elapsed time during the execution of the service, while N is the total number of calls.*
	- The Transmission Time (Trans) is the total time for communication between the client and the provider's hosting server.	$Trans(ws) = \sum (SendT(ws) + ReplyT(ws))/N(ws)$ *SendT defines the transmission time during the request sending to the server, while ReplyT is the consumed time during the transmission of the reply from the server. N is the total number of calls.*
	- The Response Time (RT) is the amount of time elapsed between sending a request and receiving a response. It is including both transmission and execution time.	$RT(ws) = Trans(ws) + Proc(ws)$ *Trans is the Transmission Time and Proc is the Processing Time.*
Reputation	**+ Usability** (Us) is describing the service characteristic of being easy to use. To measure the usability, we consider the users' feedback to rate the services based on their ease of use.	$Us(ws) = \sum UserRating(ws) / NbUse(ws)$
	- The Age (Age) is measured by using the number of days between the last dates of invoke or discover interaction and the current date. We estimate that the best WS is the one that is also recently used.	$Age(ws) = now() - LastCallDate(ws)$ *LastCallDate refers to the last date concerning the WS invocation or WS discovering operation.*
	+ The Frequency (Frq) metric represents the number of uses of the service by duration (day, week, month or year), and it's presented by the number of use and its duration.	$Frq(ws) = \sum NbUse(ws) / NbMonth(ws)$ *NbUse is the total of WS called by each duration and the NbMonth is the number of months where the WS was consumed.*
Cost	**- Cost** (Co) metric represents the incurred fees by service invocation.	

All aforementioned metrics constitute a more fine-grained service metrics taxonomy. The combination of these metrics will help surely to get eligibility (Popularity) of services. In this score, each quality metric is associated to a coefficient represented by an integer from 0 to 5. This coefficient reflects its importance among other proposed metrics when searching user appropriate services. The one which is more important has higher value.

$$WSPS(ws) = [\sum (Metric\ (ws)\ * \ Coef(Metric))\ ] / \sum (Coef(Metric)$$

Metric in {“Av”, “MTTR”, “MTBF”, “FR”, “DPM”, “Re”, “Trans”, “Proc”, “RT”, “Us”, “Age”, “Frq”, “Co”}.

To evaluate the performance of popularity score, Elfirdoussi et al. developed a framework [30] called DIVISE (DIscovery and VIsual Search Engine). DIVISE is a web service search engine that has the advantage to discover simple, composite or semantic services based on the user’s functional needs and quality metrics in order to select the most appropriate service from a generated list of potentially candidate services. We enhance the DIVISE framework for our QoS based selection module in order to automatically select the best matching service using the popularity score computation.

V. Proposed Approach

In this paper, our contribution aims to propose a comprehensive approach where the designer has a multitude of options for modeling a reliable service composition including both functional goals and NFRs. The workflow designer gains a total control of which services are meant to be implemented, according to the primary goals (FRs). Then he/she adjusts non-functional customizations by refining how these services are meant to be implemented (NFRs). In fact, we distinguish between two disjoint types of NFRs. Each of these types is fulfilled and treated differently: a) The quantifiable NFRs, commonly qualified as measurable NFRs or Quality-of-Service (QoS) requirements, such as availability, reliability, response-time, cost, etc., and b) the Behavioral requirements, as considered as non-quantifiable requirements, such as security requirements [31]. These NFRs cannot be measured using common quality metrics. The proposed approach is based on four main phases. For each phase, a dedicated module is implemented. An overview of the composition process with associated modules is illustrated in the Fig. 2.

A. Modeling Phase

The modeling module is the first interaction point between the designer and the system. It consists on a modeling tool allowing to draw adapted and easy to understand composition automata. It allows to describe desired functional and non-functional requirements by designing an Abstract Functional Automaton. Below we present the four key components used in the automata modeling module:

1) States: Each state is composed of a label which is a non-unique string attribute, and a type to describe whether it is a start, intermediate or final state.

2) Transitions: Each transition is identified by two key elements: the source state and the target state. The source state is the state launching the transition, while the target state is the reached state using that transition. Three more attributes can describe transitions which are: The inputs, the guard conditions and the outputs.

3) Behavioral scopes: The behavioral scopes are used to specify the states concerned by the behavioral requirement to integrate. They are identified using a string attribute in the form of a regular expression to describe the behavioral signature, in addition to a time indicator to specify whether the associated service will be performed before or after the scoped state’s concrete service.

4) Quality scopes: The quality scopes are used to define quality restrictions over discovered concrete services. They are identified using three key elements: a) the quality metrics which are the supported quality properties depicted in Table II, b) the metric weight which constitutes the coefficient attributed to the chosen quality metric, c) the quality condition which is an expression describing the desired restrictions over the chosen quality metric.

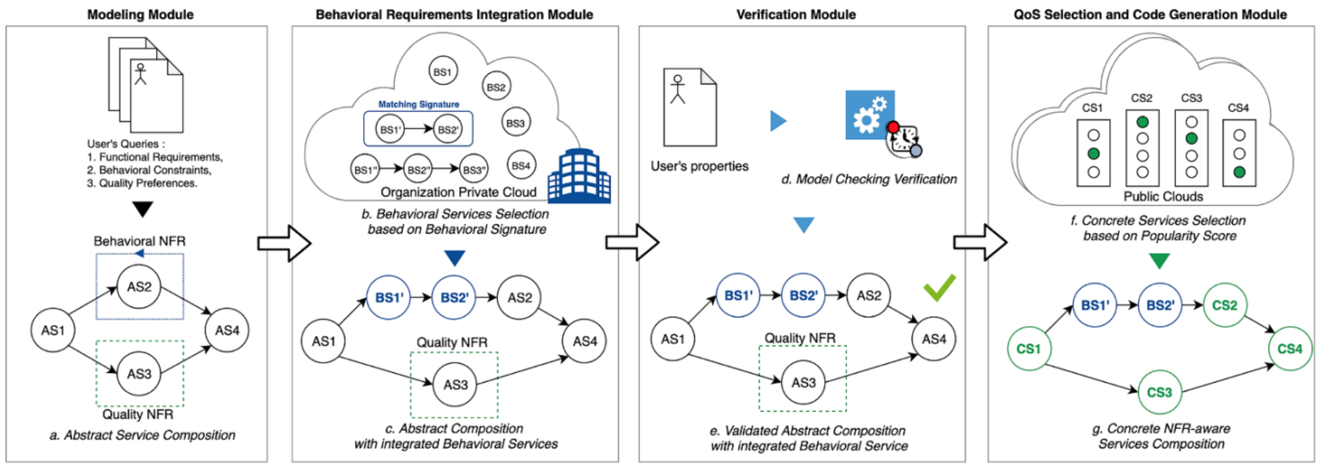
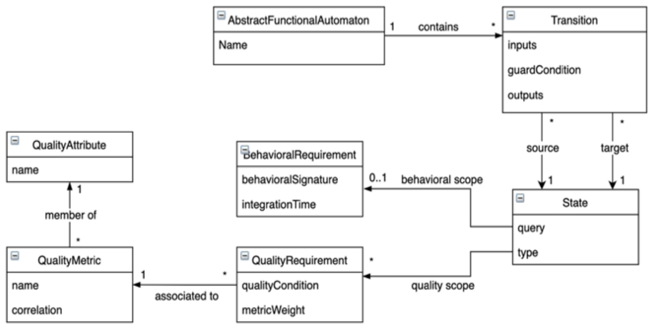
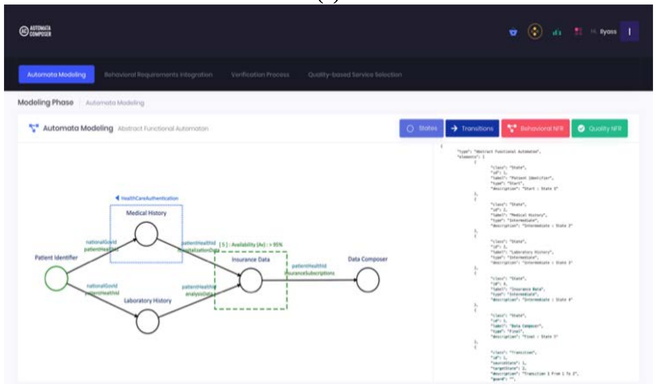

Modeling Module
User's Queries :
1. Functional Requirements,
2. Behavioral Constraints,
3. Quality Preferences.
Behavioral NFR
AS1
AS2
AS3
AS4
Quality NFR
a. Abstract Service Composition
Behavioral Requirements Integration Module
BS1
BS2
BS3
BS4
Matching Signature
BS1'
BS2'
BS1''
BS2''
BS3''
Organization Private Cloud
b. Behavioral Services Selection based on Behavioral Signature
Quality NFR
c. Abstract Composition with integrated Behavioral Services
Verification Module
User's properties
d. Model Checking Verification
e. Validated Abstract Composition with integrated Behavioral Service
QoS Selection and Code Generation Module
CS1
CS2
CS3
CS4
Public Clouds
f. Concrete Services Selection based on Popularity Score
g. Concrete NFR-aware Services Composition

Fig. 2. The Service Composition Process Workflow.

The proposed modeling module is a web-based graphical interface allowing to draw an automata models using four aforementioned key components. It automatically generates a ready-to-use JSON representation of the composition. Additionally, the composition is saved in dedicated database to allow reuse and future improvement. The class diagram we proposed to build the modeling tool is illustrated in Fig. 3(a). The Fig. 3(b) shows an example of an AFA designed using the modeling tool module.



(a)



```
{
    "type": "Abstract Functional Automaton",
    "elements": [
        {
            "class": "State",
            "id": 1,
            "label": "Patient Identifier",
            "type": "Start",
            "description": "Start : State 1"
        },
        {
            "class": "State",
            "id": 2,
            "label": "Medical History",
            "type": "Intermediate",
            "description": "Intermediate : State 2"
        },
        {
            "class": "State",
            "id": 3,
            "label": "Laboratory History",
            "type": "Intermediate",
            "description": "Intermediate : State 3"
        },
        {
            "class": "State",
            "id": 4,
            "label": "Insurance Data",
            "type": "Intermediate",
            "description": "Intermediate : State 4"
        },
        {
            "class": "State",
            "id": 5,
            "label": "Data Composer",
            "type": "Final",
            "description": "Final : State 5"
        },
        {
            "class": "Transition",
            "id": 1,
            "sourceState": 1,
            "targetState": 2,
            "description": "Transition 1 From 1 To 2",
            "guard": "",
```

(b)

Fig. 3. (a). The Class Diagram Related to the Modeling Module. (b). An Example of Designing an AFA using the Modeling Tool.

B. *Behavioral Requirements Integration Phase*

The aim of this phase consists on the integration of behavioral NFRs defined in the scopes at the modeling phase. The integration of behavioral NFRs is based on lookup operation into the behavioral NF-registry to find adequate concrete services able to fulfill the behavioral NFRs. A response is returned to the designer depending on the lookup results. If the process finds an atomic or composite service indexed by the required behavioral signature, it is automatically integrated to the AFA. Otherwise, the designer is redirected to the modeling tool in order to design an automata-based representation of the needed behavioral requirement. Then he/she develops and publishes the associated concrete services in the behavioral NF-registry. The behavioral requirements' records published are indexed using their behavioral signatures in order to facilitate their discovery and integration for further uses. The Algorithm 1 illustrates the process of integrating the behavioral NFRs into the Abstract Service Composition Automaton.

```
------------------------------------------------------------------------
Algorithm 1: Behavioral NFRs Integration
------------------------------------------------------------------------
Input:
 Set(State) states // A set of abstract states.
 Registry nfrRegistry // A non-functional registry.
Output:
 Map(State, BehavioralSignature) validStateMap
 //A map of abstract states with valid signatures.
 Map(State, BehavioralSignature) incompleteStateMap
 //A map of incomplete states: unfound behavioral signatures.
for each state in states do
 signature ← state.getBehavioralRequirement()
  .getBehavioralSignature()
 foundSignature ← nfrRegistry.lookup(signature)
 if foundSignature is NOT NULL
 validStateMap.addElement(state, foundSignature)
 else
 incompleteStateMap.addElement(state, signature)
 end if
end for
return {incompleteStateMap, validStateMap}
------------------------------------------------------------------------
```

C. *Verification Phase*

This phase consists of verifying the composition between the AFA and all behavioral NF automata models obtained from the Behavioral Requirements Integration Module. The resulted Non-Functional Composition Automaton represents a workflow process gathering the user's functional and non-functional requirements combined. In order to validate its conformity, we use Uppsala-Aalborg verification tool (UPPAAL). UPPAAL is a toolbox for verification of real-time systems. In order to automate the model checking verification using UPPAAL, we implement an intermediate adapter allowing to translate automatically our automata models to understandable UPPAAL templates. The composition automaton is reproduced in UPPAAL's formalization using a composition of multiple templates. A template is an automata-based modeling describing a specific system and illustrating interactions between its states.

The automata adapter presented in our approach performs a translation between two different schemes: (1) the first is a JSON-based representation scheme of the proposed modeling, supporting integration of quantifiable and behavioral NFRs with the initial AFA, (2) the second scheme is adapted to UPPAAL's XML description. In verification phase, quantifiable quality attributes associated to the AFA are omitted, as they intervene mainly in the service selection phase and do not affect the composition workflow.

The UPPAAL model checking tool allows also to verify the conformance of native properties such as deadlock freeness, reachability, or custom logical properties defined by the designer. It supports various properties verification [32] such as:

- Reachability properties denoted as E<> φ, which allows to check whether it exists a path starting from the initial state such that φ is eventually satisfied along that path.
- Safety properties, commonly known in the form "Something bad will possibly never happen", are denoted either with the formulae A[] φ to describe that φ should be true in all reachable states, or using the formulae E[] φ to state that there should exist a maximal path such that φ is always true.
- Liveness properties, commonly known in the form "Something will eventually happen", and denoted either using the formulae A<> φ meaning that φ is eventually satisfied, or using the formulae φ --> ψ to state that whenever φ is satisfied, then eventually ψ will be satisfied.

The verification module consists on translating the automata scheme of the Service Composition Automaton into an automata scheme understandable by UPPAAL model checking tool. This verification can be a fully automated verification or a semi-automatic verification to validate the overall modeling. The fully automated verification consists on performing a direct verification of the translated UPPAAL automata model with the designer's desired properties. Whereas the semi-automatic verification consists on adding some adjustments into the generated UPPAAL's automata modeling then running the verification of designer's properties. The main automaton and associated behavioral automata are translated into UPPAAL templates. In our approach, the formalization of SCA is comparable to UPPAAL's formalization. The modeling of SCA using the modeling tool module omits defining synchronization in the transitions, as this information can be automatically concluded from the automata modeling. In other words, parallel and synchronous executions are defined directly from the modeling. A parallel execution is performed when a source state has more than one outgoing transitions with no guard condition. In UPPAAL, we are modeling the main automaton as a main template connected to all concurrent automata using synchronization channels.

The automata adapter is introduced to transform the JSON format corresponding to the modeled AFA with integrated behavioral services into an XML-based representation adapted to UPPAAL templating format. Table III shows the main transformation rules ensuring this transition. The coordinates x and y of all elements composing the modeled automaton are forwarded to fill the attributes of associated elements in UPPAAL's XML file.

TABLE III. TRANSFORMATION RULES FROM JSON SCHEME TO UPPAAL XML SCHEME

Elements	JSON Modeling Tool Scheme	UPPAAL XML Description Sheme
Composition Automaton	{ "name": "Funct. Automaton", "elements": [...] }	<template> <name>Funct. Automaton</name> ... </template>
States	{ "class": "State", "id": 1, "label": "Service 1", "type": "Intermediate" }	<location id="1" x="" y=""> <name> Service 1 </name> </location>
Start state	{ "class": "State", "id": 1, "type": "Start" }	<location id="1" x="" y=""> <name>Service 1</name> </location> <init ref="1"/>
Final state	{ "class": "State", "id": 8, "label": "Service 8", "type": "Final" }	<init ref="1"/> <location id="8" x="" y=""> <name>Service 8</name> </location> <transition> <source ref="8"/> <target ref="1"/> </transition>
Transitions	{ "class": "Transition", "id": 4, "sourceState": 1, "targetState": 3, "description": "Transition 4 From 1 To 3", "guard": "age < 18", "input": "age" }	<transition> <source ref="1"/> <target ref="3"/> <label kind="select"> age </label> <label kind="guard"> age &lt; 18 </label> </transition>

D. The QoS-oriented Service Selection Phase

The last phase consists on building a QoS-aware concrete service composition. It is called when all designer's properties are verified. The associated module stores in a pool for each abstract service in the AFA all functionally-equivalent concrete services. Then, for each pool a QoS-based computation is performed to select the best matching service according to its popularity score. The quality requirements are initially defined in the quality scopes associated to the abstract services in the ASCA. The best matching service is the concrete service with the highest score. We describe in the algorithm below the Quality NFRs integration process.

```
-------------------------------------------------------------------------
Algorithm 2: Quality-oriented NFRs Integration
-------------------------------------------------------------------------
Input:
 Set(State) states //A set of abstract states.

Output:
 Map(State, Service) validServicesMap. /*A map of
 abstract services with appropriate validated concrete
 services.*/

for each state in states do
 stateQualityNFRs ← state.getQualityRequirements()

 for each qualityNFR in stateQualityNFRs do
 // Fetching the state's quality NFRs

 condition ← qualityNFR.getQualityCondition()

 weight← qualityNFR.getMetricWeight()

 criteriaMap.addElement(condition, weight)

 endfor

 servicePool← state
 .selectServicesByQualityNFRs(criteriaMap)
 /* Selecting the services fulfilling the state's main functional
 requirements in addition to the Quality conditions */

 for each candidateService in servicePool do

 popularityScore =
 computePopularityScore(candidateService, criteriaMap)

 scoreMap.addElement(candidateService, popularityScore)

 endfor

 bestService = scoreMap.getBestMatchingService()
 /* Selection of the best matching candidate service according
 to its score */

 validServicesMap.addElement(state, bestService)

endfor

return validServicesMap
-------------------------------------------------------------------------
```

VI. Illustrative Scenario

To illustrate the proposed approach, a collaborative scenario related to the Healthcare domain is studied, due to the diverse NF needs in this field. The aim is to create a service composition allowing to get all patient's history: health records, medical diagnosis, taken medicines, etc., then to generate a folder grouping these data. This collaborative system engages multiple Healthcare centers. We focus mainly on Regional Health Centers (RHC) and University Health Centers (UHC) in Morocco, due to the advanced information system implemented allowing a web service-based interoperability.

To initiate the modeling phase, we elaborate the Abstract Functional Automaton (AFA). In our example, the main goal consists on generating a patient health folder grouping the patient's medical history including diagnosis, medical prescriptions and analysis results. Fig. 1 illustrates the proposed AFA for the demonstrative scenario. In this AFA the states constitute the desired abstract services, and the transitions describe the intended interactions between states' corresponding concrete services.

The designed process described in this example requires as input the identification of the patient. The role of the first abstract service "Patient Identification" is to return the patient's Unique Healthcare Identifier (UHI) recognized by all Healthcare systems. The following step consists on providing, according to the patient's UHI, all patient data grouped from both RHCs and UHCs. To return all patient's health history we propose a parallel invocation of corresponding services for both types: "RHC Data Collector" and "UHC Data Collector" for regional and university healthcare centers respectively. Then, the abstract services defined as "Medicine Data Extractor" and "Diagnosis Data Extractor" aim to extract respectively the diagnosis information and the medicine information from the collected medical data history. The final step consists on searching for exhaustive information about returned medicines and diagnosis from third-party services using respectively "Medicine Data Provider" and "Diagnosis Data Provider". Finally, the abstract service "Patient Health Data Generator" will construct the patient folder with all collected information to allow returning a deep analysis based on the patient's medical history.

The proposed collaboration system involves manipulating sensitive and confidential information (medical history, diagnosis, prescription, medicine, etc.). Since this information is qualified to be very critical, we find necessary to protect the collaboration system by implementing some security policies. These security requirements are considered as Behavioral NFRs, as they define the behavioral aspect of the current process. Thus, the integration of these security-oriented behavioral NFRs should guarantee a result similar to the initial functional process but also acts on improving how this process should behave by adding security restrictions. In order to implement these behavioral NFRs to the current modeling, we integrate behavioral scopes to the Abstract Functional Automaton.

In our example, we integrate four security constraints to the functional process. They are all pre-execution behavioral requirements illustrated by the backward direction arrows on the behavioral scopes. It means that the security requirements should be implemented and executed before invoking the associated services. The behavioral signatures are defined using regular expressions labeling the behavioral scopes. The system will further lookup in the NF-Registry for associated atomic or composite services indexed by the provided behavioral signatures. In case the signature is not found in the NF-Registry, the designer proceeds to model the desired behavior as an automaton, to conceive and to publish the associated service in the NF-Registry indexed with its related behavioral signature. This process allows implementing and reusing specific services with customized behavioral needs.

The automata modeling proposed in our approach supports two possible execution paths. The success path corresponds to all possible executions leading to the valid final state. The second case concerns the executions that doesn't reach the final state, and instead, are reaching the trap state. A trap state, illustrated using the "π" symbol, is an error output. This error output can be enhanced by adding an output message describing the violated constraint in order to keep the user informed about the failure cause. The automata modeling of desired NFRs is illustrated and described below:

- The first applied behavioral NFR using the behavioral signature "RHCAuthentication" associated to the AS "RHC Data Collector" aims to integrate an Authentication system to secure and limit access to the service for registered users only.
- The second behavioral NFR labeled using the behavioral signature "UHCAuthentication.UHCAccessControl" associated to the AS labeled "UHC Data Collector" aims to integrate a Role-based Access-Control (RBAC) to the service allowing to verify the authenticated users' role before performing the related service.
- The third and fourth applied constraints labeled using the behavioral signatures "TokenAuthentication" are associated respectively to the abstract services labeled "Medicine Data Extractor" and "Diagnosis Data Extractor". They aim to restrict access to the service by integrating a Token-based Authentication system.

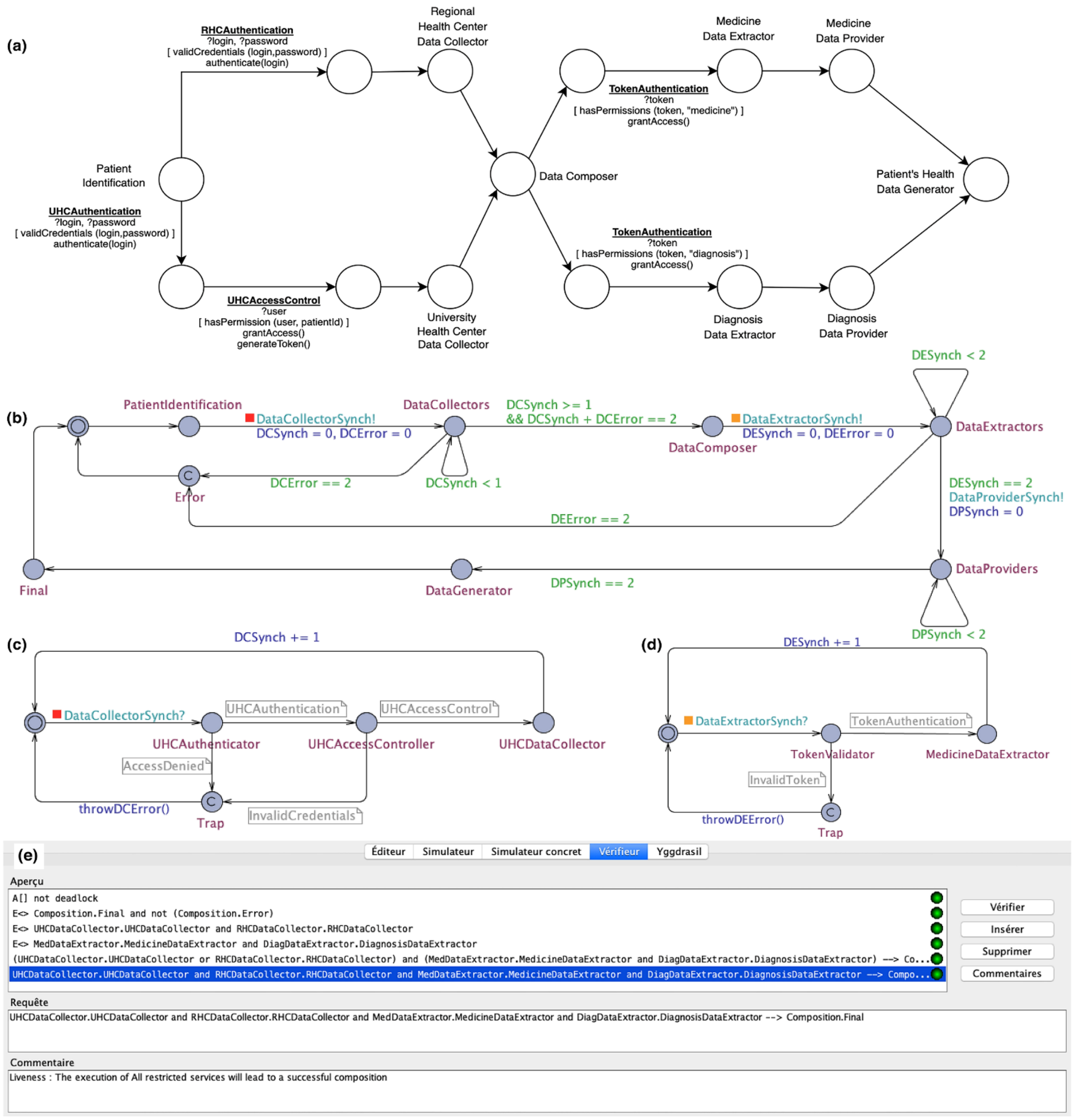
Finally, we can generate the Service Composition Automaton (SCA) by composing all the behavioral automata with their appropriate abstract services. This automaton allows to illustrate in an exhaustive way the interactions of all components. Once the SCA is generated, the following step consists on performing the model checking verification using UPPAAL.

Transitions in UPPAAL are defined as follow: The selection information, the guard conditions labelled in green color, the synchronization labelled in light blue color, and the update information labeled in dark blue color.

Fig. 4 shows the modeling of the SCA automaton using UPPAAL. The automaton illustrated in Fig. 4(b) shows the main process, gathering atomic and composite components together, using sequential and synchronous communication channels. The parallel executions are launched using broadcast channels, allowing to push a synchronization from the composition process using the exclamation mark "!" near the synch expression, and receive it on the other components using question mark "?". The first example of synchronous execution is the invocation of DataCollector services. We use "DataCollectorSynch!" in the composition process (Fig. 4(b)), which is a broadcast channel allowing to start a parallel execution of both UHCDataCollector and RHCDataCollector using "DataCollectorSynch?" in both concurrent target processes (Fig. 4(c)). In the same way the synchronous execution of "Diagnosis Data Extractor" and "Medicine Data Extractor" illustrated in the Fig. 4(d) are launched using the broadcast channel "DataExtractorSynch!" from the main process, and towards "DataExtractorSynch?" in both concurrent processes. Finally, the "Diagnosis Data Provider" and "Medicine Data Provider" are launched using "DataProviderSynch!" broadcast channel from the main process towards "DataProviderSynch?" in both concurrent processes. UPPAAL's model checking allows us to validate the correctness of designed models by verifying safety and liveness properties, in addition to user's custom logical properties related to the deployed service-oriented process. As shown in Fig. 4(e) illustrating the verified properties, the first checked property verifies whether the generated system is deadlock-free as follow "A[] not deadlock". A deadlock is an unmarked state where no events are possible. The automaton jams in a state that we have not specified as a possible final state. In our approach, we use trap states to define undesired events, happening when some predefined conditions are not met. We keep track of successful and error executions using incremental variables. It also allows to control the concurrent components executed. Table IV shows the verified properties with corresponding descriptions.

According to provided modeling we notice that all desired properties are satisfied, which means that the associated automata modeling is valid considering final state reachability, deadlock-freeness and user's custom logical preferences. The next step consists on selecting the best matching service to meet the abstract services having quality scopes. In Fig. 1 we notice that the "Medicine Data Provider" and "Diagnosis Data Provider" abstract services have quality scopes with different criteria: Response-Time and Availability for the Medicine Data Provider service and Usability for the Diagnosis Data Provider. The popularity score computation will be performed on the pool of concrete services associated to the AS "Medicine Data Provider", as they require more than one quality criterion to be fulfilled by the scoped subset. While the AS "Diagnosis Data Provider" needs only one criterion to be met, and then, no weight is required to compute the popularity score.

In Table V, we provide the computed popularity score for concurrent concrete services fulfilling the Medicine Data Provider and Diagnosis Data Provider abstract services. The service Medicine Data Provider requires two quality conditions: Response-Time < 250ms with a weight of 5, and Availability > 90% with a weight of 3. We proceed to the metric normalization in order to compute uniformly the popularity score. For rate-based metrics i.e. metrics calculated in a percentage basis, the score constitutes the value of the measured quality metric when the measurement correlation is positive. Otherwise, when the correlation is negative, the score is the subtraction of the measured value from a basis of 100. In other hand, for non-rate-based quality metrics, we use the proportional computation of the service metric value according to the maximal value for the target metric. The maximal value for a non-rate-based quality metric is concluded by using the desired value as a median. For instance, for the desired value of Response-Time less than 250ms we use this value as a median to conclude that the maximal value for the Response-Time is 500ms. The second method consists on providing the maximal values for each quality metric by the expert rather than concluding it using the reversed median calculation. A deeper explanation of the proposed popularity score computation methods is provided in a previous work [5].



Aperçu

```
A[] not deadlock
E<> Composition.Final and not (Composition.Error)
E<> UHCDataCollector.UHCDataCollector and RHCDataCollector.RHCDataCollector
E<> MedDataExtractor.MedicineDataExtractor and DiagDataExtractor.DiagnosisDataExtractor
(UHCDataCollector.UHCDataCollector or RHCDataCollector.RHCDataCollector) and (MedDataExtractor.MedicineDataExtractor and DiagDataExtractor.DiagnosisDataExtractor) --> Co...
UHCDataCollector.UHCDataCollector and RHCDataCollector.RHCDataCollector and MedDataExtractor.MedicineDataExtractor and DiagDataExtractor.DiagnosisDataExtractor --> Compo...
```

Requête

```
UHCDataCollector.UHCDataCollector and RHCDataCollector.RHCDataCollector and MedDataExtractor.MedicineDataExtractor and DiagDataExtractor.DiagnosisDataExtractor --> Composition.Final
```

Commentaire

Liveness : The execution of All restricted services will lead to a successful composition

Fig. 4. The Service Composition Automaton and its Appropriate UPPAAL Modelling and Verification.

TABLE IV. UPPAAL'S VERIFIED PROPERTIES WITH CORRESPONDING DESCRIPTIONS

Verified properties	Descriptions
E<> Composition.Final AND NOT (Composition.Error)	There exists eventually a path leading to the final state without reaching any error state.
E<> UHCDataCollector.UHCDataCollector AND RHCDataCollector.RHCDataCollector	There is eventually a parallel execution of the Data Collectors in both UHC and RHC. It assumes that the associated security services are preliminarily performed, i.e. UHCAuthenticator and UHCAccessController before UHCDataCollector, and RHCAuthenticator before RHCDataCollector.
E<> MedDataExtractor.MedicineDataExtractor AND DiagDataExtractor.DiagnosisDataExtractor	There is eventually a parallel execution of Medicine Data Extractors and Diagnosis Data Extractor.
(UHCDataCollector.UHCDataCollector OR RHCDataCollector.RHCDataCollector) AND (MedDataExtractor.MedicineDataExtractor AND DiagDataExtractor.DiagnosisDataExtractor) → Composition.Final	A successful composition is conditioned by an execution of at least one Data Collector of either UHC or RHC (OR), additionally to an execution of both Medicine Data Extractor and Diagnosis Data Extractor (AND)
UHCDataCollector.UHCDataCollector AND RHCDataCollector.RHCDataCollector AND MedDataExtractor.MedicineDataExtractor AND DiagDataExtractor.DiagnosisDataExtractor → Composition.Final	An execution of all restricted services which are UHCs' and RHCs' Data Collectors in addition to Diagnosis and Medicine Data Extractors will lead to a successful execution of the composition

TABLE V. CONCRETE SERVICE COMPARISON BASED ON POPULARITY SCORE

Abstract Service	Associated Concrete Services	Popularity Score
Medicine Data Provider	**WS$_{11}$ (Response-Time: 150ms, Availability: 94%)**	**90.28**
	WS$_{12}$ (Response-Time: 200ms, Availability: 98%)	84.85
Diagnosis Data Provider	WS$_{21}$ (Usability:8.2)	82
	WS$_{23}$ (Usability:6.9)	69
	WS$_{24}$ (Usability:8.8)	**88**

The final step consists on generating a ready-to-execute BPEL code using the engine provided by the previously developed framework, i.e., Discovery and Visual Interactive Web Service Engine (DIVISE) [30]. The produced code of the composite service assembles all selected concrete services with their appropriate interactions according the validated process. In this contribution we aim to enhance the engine by providing design and verification-oriented modules allowing an exhaustive modeling taking into account functional requirements in addition to both behavioral and measurable non-functional requirements.

VII. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

Our current contribution consists on defining a more fine-grained composition process workflow and its implementation and handling the main phases from the design time to code generation. This workflow integrates both behavioral and measurable quality-oriented NFRs into service composition process. An Automata-based modeling of the functional requirements (FRs) and non-functional requirements (NFRs) is suggested with an explicit distinction between measurable quality-oriented NFRs and the newly introduced behavioral NFRs. These NFRs are integrated into the abstract functional automaton using scopes. The needed behavioral NFRs are modeled separately then merged to the functional abstract automaton in order to perform a model checking verification using UPPAAL. In addition, the desired measurable quality-oriented NFRs have no impact on the behavioral workflow of the composition automaton, and are explored in the selection phase using Popularity score enabling to return the best matching concrete services for each associated abstract service. A use case process using Patient Health Records in Regional and University Health Centers in Morocco is used to experiment our approach.

Although this approach handles the overall process of service composition from design to execution phases, it can be considered as limited to the current state of evaluated QoS properties of services, as we did not integrate the tracking module in the current contribution. Thus, the service selection based on Popularity Score is using provided values for each quality metric. These values are provided mainly by the provider. However, for an accurate classification, we are orienting our research to perform a new computation based on service tracking of service quality over time using Machine Learning techniques and technologies. It will allow us to have a clear idea regarding variation of service quality in a wide timeline, and compute the Popularity Score either using average function for the whole time or partially for a recent limited period of time.

REFERENCES

[1] L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos, "Non-Functional Requirements in Software Engineering", 2000, DOI: 10.1007/978-1-4615-5269-7.

[2] J. Eckhardt, A. Vogelsang, D. Méndez Fernandez, "Are Non-functional Requirements really Non-functional? An Investigation of Non-functional Requirements in Practice", in 38th IEEE International Conference on Software Engineering (ICSE '16), pp. 832–842, 2016. DOI: https://doi.org/10.1145/2884781.2884788.

[3] M. H. Beek, A. Bucchiarome, S. Gnesi, "Formal Methods for Service Composition", in Annals of Mathematics, Computing & Teleinformatics, vol. 1, pp 1-10, 2007.

[4] I. El Kassmi, Z. Jarir, "Security Requirements in Web Service Composition: Formalization, Integration, and Verification", in 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Paris, France, 2016. DOI: 10.1109/WETICE.2016.47.

[5] I. El Kassmi, Z. Jarir, "Toward a Smart Cloud Service Composition: Popularity-Driven Approach", in The 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 522-528, 2018. DOI: 10.1109/SITIS.2018.00085.

[6] Y. Chen, J. Huang, C. Lin, X. Shen, "Multi-Objective Service Composition with QoS Dependencies", in IEEE Transactions on Cloud Computing, vol. 7, pp. 537-552, 2019. DOI: 10.1109/TCC.2016. 2607750.

[7] S. Deng, H. Wu, H. Hu, J. Leon Zhao, "Service Selection for Composition with QoS Correlations", in IEEE Transactions on Services Computing, vol. 9, pp. 291-303, 2016. DOI: 10.1109/TSC.2014. 2361138.

[8] S. K. Gavvala, C. Jatoth, G. R. Gangadharan, and R. Buyya, "QoS-aware cloud service composition using eagle strategy," in Future Generation Computer Systems, vol. 90, pp. 273–290, Jan. 2019.

[9] Y. Liang, H. Hu, W. Song, J. Ge, "QoS-aware Automatic Web Service Composition Considering QoS Correlations", in Proc. of the 7th Asia-Pacific Symposium on Internetware, pp. 39-42, 2015. DOI: https://doi.org/10.1145/2875913.2875940.

[10] Y. Feng, L. Ngan, R. Kanagasabai, "Dynamic Service Composition with Service-Dependent QoS Attributes", in IEEE 20th International Conference on Web Services, pp. 10-17, 2013. DOI: 10.1109/ICWS .2013.12.

[11] C. Jatoth, G.R. Gangadharan, U. Fiore, R. Buyya, "QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation", in Future Generation Computer Systems, vol. 86, pp. 1008-1018, 2018. DOI:http://dx.doi.org/10.1016/j.future.2017.07. 042.

[12] H. Jin, X. Yao, Y. Chen, "Correlation-aware QoS modeling and manufacturing cloud service composition", in Journal of Intelligent Manufacturing, vol. 28, pp. 1947-1960, 2017. DOI:https://doi.org/10. 1007/s10845-015-1080-2.

[13] H. Liang, Y. Du, "Dynamic service selection with QoS constraints and inter-service correlations using cooperative coevolution", in Future Generation Computer Systems, vol. 76, pp. 119-135, 2017. DOI:https://doi.org/10.1016/j.future.2017.05.019.

[14] H. Wang, D. Yang, Q. Yu, Y. Tao, "Integrating Modified Cuckoo Algorithm and Creditability Evaluation for QoS-Aware Service Composition", in Knowledge-Based Systems, vol. 140, pp. 64-81, 2017. DOI: 10.1016/j.knosys.2017.10.027.

[15] J. Lu, Z. Huang, C. Ke, "Verification of Behavioral-aware Privacy Requirements in Web Service Composition", in Journal of Software, vol. 9, pp. 944-951, 2014. DOI:10.4304/jsw.9.4.944-951.

[16] W. Dou, X. Zhang, J. Liu, J. Chen, "HireSome-II: Towards Privacy-Aware Cross-Cloud Service Composition for Big Data Applications", in IEEE Transactions on Parallel and Distributed Systems, vol. 26, pp. 455-466, 2015. DOI: 10.1109/TPDS.2013.246.

[17] Xx5 A. Souri, A.M. Rahmani, N.J. Navimipour, "A hybrid formal verification approach for QoS-aware multi-cloud service composition". Cluster Computing vol. 23, pp. 2453–2470, 2020. DOI: https://doi.org/10.1007/s10586-019-03018-9.

[18] A.D. Brucker, B. Zhou, F. Malmignati, Q. Shi, M. Merabti., "Modelling, validating, and ranking of secure service compositions", in Journal of Software: Practice and Experience, vol. 47, pp. 1923–1943, 2017. DOI: https://doi.org/10.1002/spe.2513.

[19] G.N. Rai, G.R. Gangadharan, "Model Checking Based Web Service Verification: A Systematic Literature Review", in IEEE Transactions on Services Computing. DOI: 10.1109/TSC.2018.2845401.

[20] M. H. Beek, A. Bucchiarome, S. Gnesi, "A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods", in Technical Report, 2006.

[21] A. Charfi, M. Mezini, "Aspect-Oriented Web Service Composition with AO4BPEL", in Web Services, ECOWS 2004, vol. 3250, pp. 168-182. DOI: https://doi.org/10.1007/978-3-540-30209-4_13.

[22] Z. Shen, J. Su, "Web service discovery based on behavior signatures", in 2005 IEEE International Conference on Services Computing, vol. 1, pp. 279-286, 2005. DOI: 10.1109/SCC.2005.107.

[23] L. Chung, J.C.S. do Prado Leite, "On Non-Functional Requirements in Software Engineering" in Conceptual Modeling: Foundations and Applications, pp. 363-379, 2009.

[24] ISO/IEC9126-1:2001, "Software Engineering – Product Quality-Part1: Quality Model", 2001.

[25] ISO/IEC25010:2011, "System and Software – System and Software Quality Requirements and Evaluation Engineering: System and Software Quality Model", 2011.

[26] M. Galster, E. Bucherer, "A Taxonomy for Identifying and Specifying Non-functional Requirements in Service-oriented Development", in IEEE Congress on services, pp. 345-352, 2008. DOI: 10.1109/SERVICES-1.2008.51.

[27] D. Mairiza, D. Zowghi, N. Nurmuliani, "An investigation into the notion of non-functional requirements", in ACM Symposium on Applied Computing, pp. 311-318, 2010. DOI:https://doi.org/10.1145/1774088. 1774153.

[28] M.M. Hasan, P. Loucopoulos, M. Nikolaidou, "Classification and Qualitative Analysis of Non-Functional Requirements Approaches", in Enterprise, Business-Process and Information Systems Modeling, vol. 175, pp. 348-362, 2014.

[29] S. Elfirdoussi, Z. Jarir, M. Quafafou, "Ranking Web Services using Web Service Popularity Score" in International Journal of Information Technology and Web Engineering, vol. 9, pp. 78-89, 2014. DOI: 10.4018/ijitwe.2014040105.

[30] S. Elfirdoussi, Z. Jarir, M. Quafafou, "Discovery and Visual Interactive WS Engine based on popularity: Architecture and Implementation", in International Journal of Software Engineering and Its Applications, vol. 8, pp. 213-228, 2014. DOI: 10.14257/ijseia.2014.8.2.21.

[31] D.G. Firesmith. "Engineering Security Requirements" in Journal of Object Technology, vol. 2, 2003. DOI: 10.5381/jot.2003.2.1.c6.

[32] G. Behrmann, A. David, K.G. Larsen, "A Tutorial on UPPAAL", in Formal Methods for the Design of Real-Time Systems, pp. 200-236, 2004. DOI: 10.1007/978-3-540-30080-9_7.