Combining planning and learning for context aware service composition

Tarik Fissaa*

EVEREST Team, STRS Lab,
INPT Rabat, Morocco
Email: Fissaa.tarik@gmail.com
*Corresponding author

Mahmoud El Hamlaoui and Hatim Guermah

IMS Team, ADMIR Lab, ENSIAS,
Mohammed V University in Rabat, Morocco
Email: mahmoud.elhamlaoui@gmail.com
Email: hatim.guermah@um5s.net.ma

Hatim Hafiddi

EVEREST Team, STRS Lab,
INPT Rabat, Morocco
Email: hatim.hafiddi@gmail.com

Mahmoud Nassar

IMS Team, ADMIR Lab, ENSIAS,
Mohammed V University in Rabat, Morocco
Email: nassar.ensias@gmail.com

Abstract: Computing vision introduced by Mark Weiser in the early ‘90s has defined the basis of what is called now ubiquitous computing. This new discipline results from the convergence of powerful, small and affordable computing devices with networking technologies that connect them all together. Thus, ubiquitous computing has brought a new generation of service-oriented architectures (SOA) based on context-aware services. These architectures provide users with personalised and adapted behaviours by composing multiple services according to their contexts. In this context, the objective of this paper is to propose an approach for context-aware semantic-based services composition. Our contributions are built around following axes: 1) a semantic-based context modelling and context-aware semantic composite service specification; 2) an architecture for context-aware semantic-based services composition using artificial intelligence planning; 3) an intelligent mechanism based on reinforcement learning for context-aware selection in order to deal with dynamicity and uncertain character of modern ubiquitous environment.

Keywords: context awareness; ontology; service composition; semantic web; AI planning; reinforcement learning.

Reference to this paper should be made as follows: Fissaa, T., El Hamlaoui, M., Guermah, H., Hafiddi, H. and Nassar, M. (2021) ‘Combining planning and learning for context aware service composition’, *Int. J. Data Analysis Techniques and Strategies*, Vol. 13, Nos. 1/2, pp.151–169.

Biographical notes: Tarik Fissaa is an Assistant Professor of Software Engineering at the INPT Rabat, Morocco, before he was a postdoctoral researcher at the Basque Center for Climate Change (BC3) working on ARIES project aiming at applying semantic web technologies and machine learning for sustainability. He received his Master’s in Computer Science and Telecommunications from the Faculty of Science Rabat, Morocco in 2011 and a PhD from the ENSIAS Engineering School, Mohammed V University of Rabat, Morocco. His research interests are context-aware service-oriented computing, semantic web and artificial intelligence.

Mahmoud El Hamlaoui is an Assistant Professor of Software Engineering at ENSIAS@UM5R. He is involved with both research team IMS of the Rabat IT Center and the research team SM@RT of the Research Institute in Computer Science of Toulouse (IRIT). His research areas include model-driven (software) engineering, component-based software engineering, software language engineering, domain specific modelling languages and metamodelling, cloud computing.

Hatim Guermah is a Professor in the Software Engineering Department at National Higher School for Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco. He received his Engineering degree and PhD in Software Engineering from ENSIAS of Rabat. His research activities focus on context-aware service-oriented computing ontologies and semantic web, artificial intelligence and machine learning, model driven architecture and aspect oriented programming.

Hatim Hafiddi is a Professor of Software Engineering at National Institute of Posts and Telecommunications (INPT), Rabat, Morocco. He holds an Engineering degree and a PhD in Software Engineering from the National College of IT (ENSIAS), Rabat, Morocco. His research interests are context-aware service-oriented computing, mobile information systems engineering and IS governance in cloud computing.

Mahmoud Nassar is Professor and Director of Doctoral Studies Centre on Information Technologies and Engineering Sciences (ST2I) at National Higher School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, Morocco. He is also the Head of IMS (IT Architecture and Model Driven Development Systems) Team of Rabat IT Centre. He received his PhD in Computer Science from the INPT Institute of Toulouse, France. His research interests are context-aware service-oriented computing, model-driven engineering, cloud computing, big data and machine learning. He leads numerous R&D projects related to the application of these domains in complex systems, social networks, e-health, and e-tourism.

This paper is a revised and expanded version of a paper entitled ‘An intelligent approach for context-aware service selection using machine learning’ presented at LOPAL 2018, Rabat, Morocco, 2–5 May 2018.

1 Introduction

The adoption of *as a service* paradigm promotes a new approach to the design and development of new applications as they can be created by composing different services that already exist. As the number of web services deployed on the internet increases, their composition becomes a fundamental feature for potential customers and poses significant challenges. The service composition process can be divided into two phases: definition of abstract schema and the selection of concrete executable services. In recent years we have witnessed a proliferation of approaches that tend to automate some phases or the composition process (Alrifai et al., 2012).

In parallel, ubiquitous computing, especially context awareness has reached its peak in recent years thanks to the revolution in computing devices that have become faster, smaller, more widespread and universally connected as the wireless revolution continues. At the same time, advances in sensor technologies and the development of extraction and knowledge management capabilities have allowed context-aware systems to grow.

Several works have attempted to formalise the meaning of context in the computing area. However, a universally accepted definition is yet to be agreed. Dey and Abowd (2000) defined context as: "...any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". With the large number of services currently available, several concurrent and functionally equivalent services may participate in a composition. A popular strategy of addressing these issues is to decompose the overall process into two subsequent phases. In order to satisfy the requested functionalities, an abstract workflow is generated (Shin et al., 2009; Wang and Yu, 2009). Afterward, concrete services are selected by choosing the best composite service among the candidate services based on the non-functional parameters. This second phase is also known as service selection problem.

In this paper, we present an intelligent approach to enable context-aware service composition in automatic and dynamic manner. Our strategy uses planning to generate abstract plans and reinforcement learning which is a branch of machine learning concerned on how to take actions to maximise some notion of cumulative rewards in order to select concrete services at run-time. Thus, we model the composition phase as an AI planning problem to automate the composition and the context-aware service selection as a Markov decision process (MDP) (Puterman, 2014) to tackle the uncertain aspect of modern computing environment.

By combining planning and learning, our approach responds to the challenges of the context aware service composition. The composition is conducted to choose a valid plan given a contextual request. Afterwards, the selection is conducted at run-time (when users invoke the services), through reinforcement learning. The learning aims to obtain the optimal policy of the Markov decision process that delivers the best quality of service. Unlike the majority of existing approaches, our method requires no prior knowledge of services context properties, which are anyway dynamic. Instead, it learns these properties by interacting with the environment.

The remainder of this paper is organised as follows: Section 2 tackles the modelling of context awareness by presenting an ontology-based context model and extending semantic web services with context element, Section 3 presents the proposed architecture for combining planning and learning, Section 4 tackles the problem of abstract service composition by using AI planning techniques, Section 5 presents our proposal on how to model service selection as MDP and to solve it via Reinforcement Learning. Section 6 presents a case study and Section 7 compares our approach against some related works. Finally, a conclusion provides plans for future work.

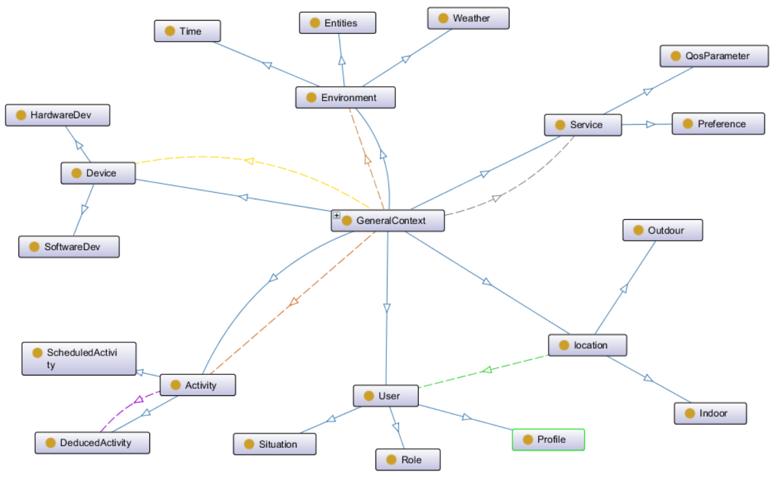
2 Context awareness modelling

The use of context awareness in the service composition process is of vital importance in modern ubiquitous environment. Thus, composite service adapt to its surrounding settings automatically by exploiting semantic information. To do so, there is need to extend semantic descriptions of services with context parameters and different contextual conditions that can change and adapt the result of service composition. In this section, we present our proposal for context and services modelling by using ontologies and semantic web services.

2.1 Ontology-based context modelling

One of the activities to perform during the development of context-aware applications is to define a context model to represent and manage context information. Currently, there is a lack of consensual models, and this supposes a handicap when developing these applications. As the context may be considered as specific kind of knowledge, it can be modelled as an ontology.

Figure 1 Context ontology model (see online version for colours)



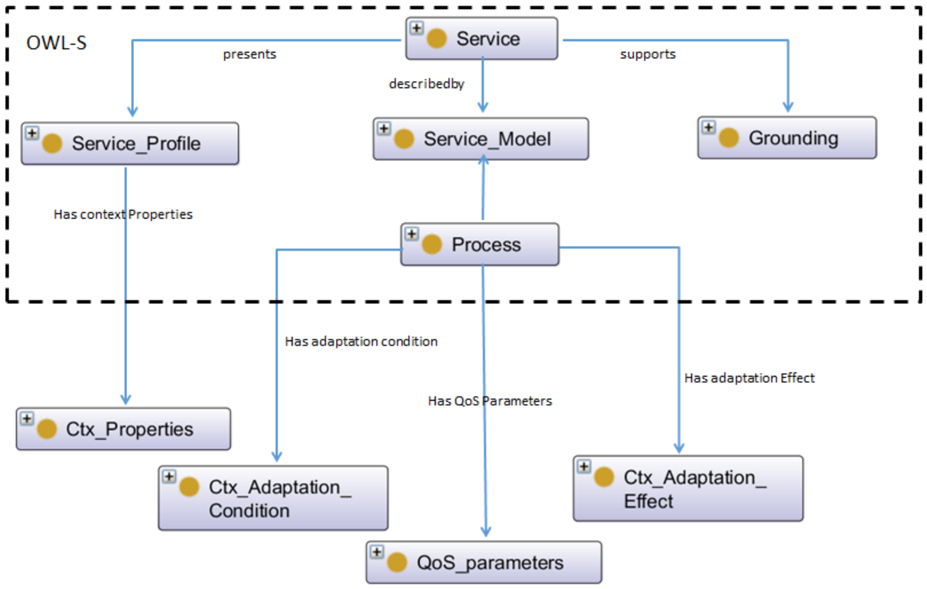
The context ontology consists of two levels, a domain independent level containing the general aspects of a context-aware systems: 'user', 'activity', 'environment', 'service' and 'device' and a specific level depending on the application domain for example for example: medical information, vital parameters of the patient, hospitals, etc. The key context concepts used are presented in Figure 1:

- user: contains the information about the user, his profile, his situation
- activity: describes the different activities in a context
- environment: describes the concepts related to the environment such as weather, weather and entities interacting with the user (hospitals, pharmacies)
- location: describes the location
- device: describes the different devices used in a ubiquitous environment
- service: describes the characteristics of a service (e.g., QoS).

2.2 *Context aware service modelling*

OWL-S (Martin et al., 2007) is an ontology for the description of semantic web services expressed in the web ontology language (OWL). OWL-S is composed of three ontologies: the Profile for discovery, the service model describes the behaviour and the grounding describes how the service can be accessed.

Figure 2 CA-OWLS: semantic context aware service (see online version for colours)



To be context-aware we need certain mechanisms to adapt services (composite or single ones)to the context of use, which is not the case in actual OWL-S. To tackle this issue we extend the language with context elements. The extension contains context conditions, context parameter and QoS/QoE parameters.

The CA-OWLS is based on the following specifications (Figure 2):

- the semantic context aware service has a ServiceModel, ServiceProfile and ServiceGrounding
- the service model can be viewed as a process
- the process contains AtomicProcess, CompositeProcess or SimpleProcess
- the ServiceProfile is related to a context property
- each ContextProperty contains contextual attributes
- adaptation condition: the service may require certain external pre-condition to be satisfied to execute the process
- adaptation effect: the execution of the service may result in certain external effects
- quality of services (QoS) parameters.

QoS refers to the quality of a service, it can respond to a query, perform related tasks with a certain quality of service, and provide quality of service to meet expectations. In the web services domain, the most commonly used non-functional attribute parameters include cost, response time, availability, security, reliability, and reputation. Based on other research (Alrifai and Risse, 2009), this article focuses primarily on general properties related to the performance of the web service, namely the cost of the service, the response time, reliability, availability and especially we propose to add another parameter related to user satisfaction or feedback (user experience quality). The definition of each parameter is as follows:

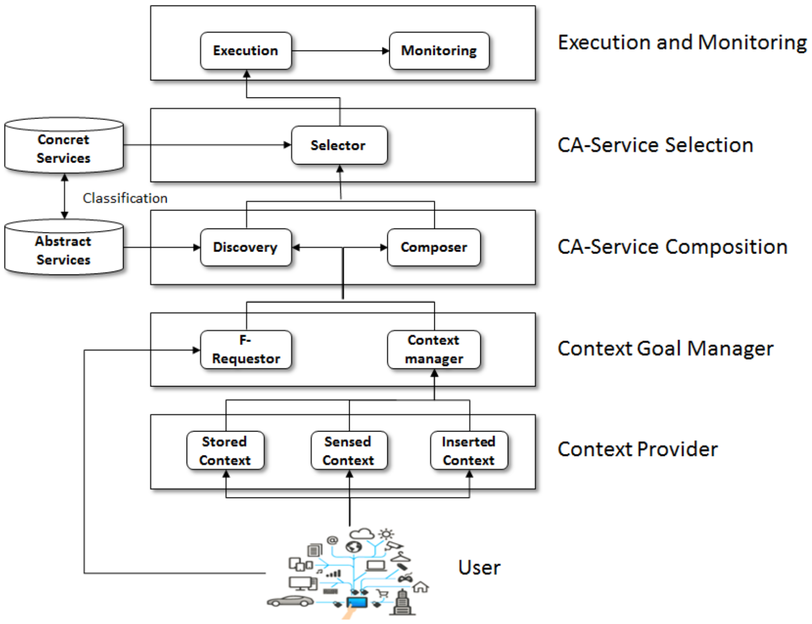
- Cost: $q_c(s)$, represents the cost that the consumer must pay to invoke the service.
- Service availability: $qa(s) = \frac{T(s)}{t}$, where t is a time interval and $T(s)$ is the execution time of a service in the time interval.
- Response time: $q_t(s) = T_e + T_t$, where T_e is the execution time of the service. T_t is the communication time between the service consumer and the service providers.
- Service reliability: $q_r(s) = \frac{N_s}{N_t}$, where N_s is the number of successful services execution. N_t is the total number of invoked services.
- User experience: $q_f(s)$ refers to the quality of the user's satisfaction or user feedback (user experience quality), this parameter is entered directly by the user (for example a graphical interface with a cursor to put on the desired feedback).

3 CASC: an architecture for CA service composition

In this section, we present the architecture of the framework 'CASC', based on the requirements stated above. It describes its main layers, presents the features offered by each layer and the interactions between them in order to facilitate the Composition. The use of learning in all composition steps automates tasks related to the context-aware composition of services.

Figure 3 shows an overview of the framework. These layers correspond to the four main stages of the service composition life cycle in addition to the context provider layer.

Figure 3 Architecture overview (see online version for colours)



4 AI planning for context aware service composition

The proposed approach for context aware service composition uses AI planning. At first, we'll see how to specify the user's contextual request. Next, we will see how to use AI planning to generate abstract compositions.

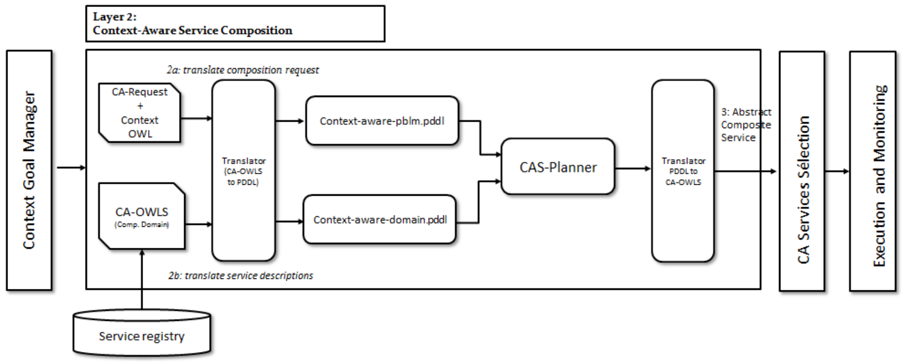
4.1 Context goal manager

The contextual request management is the first component of the system. The user or an agent on his behalf sends a contextual request. This request is personalised by adding context element for every consumer. Then, the composition system responds to this contextual query. If the context changes during the composition, the control is passed back to the composition context goal manager layer, which will transform the user request into an alternative one. The description of the goal that represents the user task or request is given in the form of an abstract CA-OWLS process, in order to take into account the contextual conditions in the initial state and goal state. The main difference between the context goal descriptions and service descriptions is that, in contrast to the atomic processes involved in the services processes those involved in user task processes are not bound to any concrete service, since services to be invoked are dynamically discovered. Thus the OWL-S grounding corresponding to the request is generated at run-time from the Groundings of the composed context aware services. The context manager collects information from different entities that affect context (e.g., sensors, users, devices, etc.). Then, it uses a context ontology to provide contextual information formatted in this ontology.

4.2 *Context aware service composition*

The context aware composition layer consists of pretreatment and planning modules. A set of semantically described services associated with OWL context ontology containing all concepts related to user context and the contextual request are provided as input and the composition layer uses AI planning algorithms to extract a plan of services according to a given context. The architecture in Figure 4 includes the following modules:

Figure 4 Context aware planning layer


Layer 2:
Context-Aware Service Composition
Context Goal Manager
2a: translate composition request
CA-Request + Context OWL
CA-OWLS (Comp. Domain)
Translator (CA-OWLS to PDDL)
Context-aware-pblm.pddl
Context-aware-domain.pddl
CAS-Planner
Translator PDDL to CA-OWLS
3: Abstract Composite Service
2b: translate service descriptions
Service registry
CA Services Sélection
Execution and Monitoring

4.2.1 *CA-OWLS to PDDL*

The role of the ‘translator’ module is to translate the descriptions of all types of semantic services (atomic/composite OWLS or atomic/composite CA-OWLS) and the context goal ontology into a domain and problem planning respectively in planning domain definition language (PDDL) to be used by an AI planner. The choice of PDDL is justified by the fact that it is the standard language in the AI planning domain, the resemblance between semantic web services and PDDL, and the possibility to use a wide variety of planners that support PDDL syntax. Formally the translator implements a function that map a context aware composition problem to an AI planning one. The function is represented by:

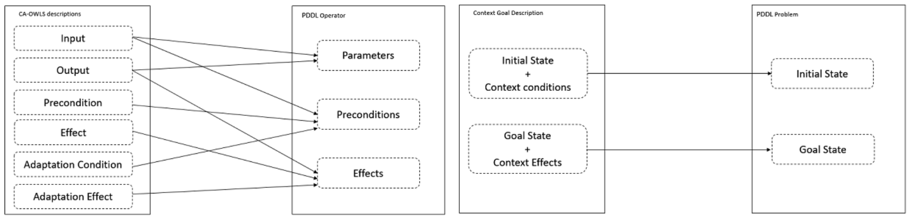
$$(IC \times GC \times 2S) \rightarrow (DC, PC)$$

where IC and GC represents the contextual request (initial and goal states), S is a set of semantic context aware services, DC stand for the contextualised domain, and PC stands for the contextualised problem.

The input/output and context parameters in the service description are converted into additional conditions and effects. For this purpose, for each input/output parameter or context parameter, a specific predicate ‘hasKnowledge’ is created. These parameters are defined in the context ontology and the CA-OWLS as the predicate variable. For example, if a service has an entry related to the concept ‘location’ as a context parameter for the service, the predicate ‘hasKnowledge (Location)’ is considered as an additional adaptation condition.

Every web services in a repository is called to create a corresponding PDDL action.

Figure 5 CA-OWLS to PDDL translation



Every other parameter (IO and context elements) are transformed into additional preconditions and postconditions. To this aim, we create a predicate (‘has knowledge’) defined in the ontology to facilitate the inclusion of IO and context in the composition process. For example, the context parameter location we create a predicate ‘hasKnowledge(Location)’ as additional information (Figure 5). The initial state (respectively the goal state) of the context goal ontology is translated to equivalent initial and goal state in the planning problem (Figure 5).

4.2.2 Automatic CA service composition as AI planning

We define the context aware service composition problem as AI planning. Thus, planning for the C3S focuses on selecting the appropriate actions and their order to achieve a certain goal in a given context. An AI planning for CA service composition can be formalised by $< S, S0, G, A, \Gamma >$, where:

- S is the set of all possible facts.
- $S0 \subset S$ denotes the initial state.
- $G \subset S$ denotes the goal state.
- A is the set of services
- $\Gamma \subset S \times A \times S$ a transition function.

A state can be either all the conditions necessary to the execution of a service (preconditions), or all the effects produced after the execution (effects). The problem entries and the desired outputs can be expressed as predicates in the initial state s0 (describing the available knowledge) and predicates in the final state (describing the additional knowledge produced by the service execution). Take the example of a navigation service, a user wants to have the route to the hospital. If the adaptation condition is in the ‘walking’ state, then the adaptation effect will be to display the service on its phone, in case the state is ‘driving’ the adaptation effect will be to use the ‘in-vehicle’ infotainment system.

So we have two types of states, ordinary ones and context ones. The ordinary state represents functional parameters, and also the preconditions and the effects of those parameters. The contextual states can be accessible by users, devices, etc. They are not essential for the system. Their role is to personalise services and enhance the user experience; it includes all context information.

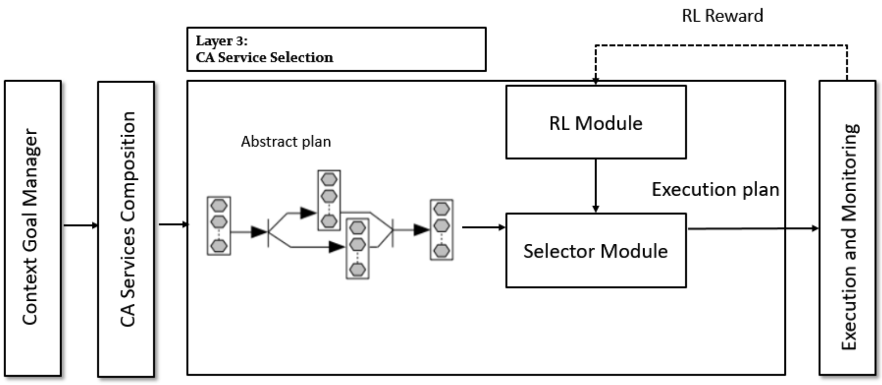
Two states are connected by a transition if there is a service that can be invoked to go from the first state to the second. A service triggers a transition from conditions to effects. The expression represents this transition relationship.

In our case we used the Graphplan algorithm (Blum and Furst, 1997), but any other planning algorithm can be used. The Graphplan algorithm has been proposed by Blum and Furst to effectively solve planning problems. It consists of two intertwined phases: a forward phase, where the graph is built, and a backward phase which is a phase for searching a valid plan in the planning graph.

5 Reinforcement learning for context aware service selection

Figure 6 shows the architecture of the context-aware service selection approach based on the quality of service and user preferences and experience.

Figure 6 Context aware selection layer



The architecture receives as input an abstract plan issued from the previous layer. The next step is to choose the concrete services according to the user’s preferences and QoS parameters. After that, the result is passed to the execution and monitoring layer, then, the reinforcement learning module receives a contextual reward to use it in learning during the next selections. The architecture is composed of the following layers:

- service selection layer
- execution and monitoring layer.

Following we explain how to model the problem of context aware selection by Markov decision processes.

5.1 A MDP model for context-aware service selection

In this section, we will mainly introduce a MDP model to solve the context-aware service selection problem, we formally define key concepts used in the model. Next, we show how to solve it using reinforcement learning technique.

5.1.1 Problem formalisation

A context-aware service selection problem based on a MDP is a 6-tuples $\langle S, s_0, s_t, A(s), P, R \rangle$ where:

- S: a finite set of states representing abstract services.
- $s_0 \in S$: is an initial state, the state before the execution of the actions. That is, the initial value of the selection process.
- $s_t \in S$ is the set of terminal states, also denotes the set of objective state for the user. Arriving at one of these states, the service selection ends.
- $A(s)$: is the set of concrete web services in a certain state $s \in S$. It is composed of web services whose preconditions are satisfied at the current state.
- $P : [P_{iaj}]S \times A \times S \to [0, 1]$: the transition function for the agent. It means the probability of the system going to the next state by calling the services available in current states, which can also be interpreted as the reliability of this service.
- $R : [R_{iaj}]S \times A \to R$: is the contextual reward that the agent receives when he transfers from the state s state to s' after calling the service a.

A MDP graph can be viewed as a graph transition. As illustrated in Figure 4 Section 5 (e-health scenario), the scenario presents a medical emergency management case. The graph in the figure (Section 5) shows a process consisting of three classes of abstract services each containing a number of concrete services. Thus, the service selection problem can be modelled as a transition graph of Markov decision process. This service selection presents several possibilities for users depending on the context (preference and quality of services). When performing the service execution, the system operates and selects the composite service with the highest contextual reward.

Definition (policy)

A policy π is a transition from the state $s \in S$ to a service $ws \in A$, which indicates which service $ws = (s)$ must be invoked in state s. An example of policy can be expressed by:

$$\pi = Alm2 \to HF3 \to MA1.$$

Each policy of a MDP can be converted into a single composite service, and the user can get some contextual reward after running the selected composite service. This reward means the cumulative reward of all services when performed. For a given MDP, the main task of our selection system is to find the optimal policy or composite service that can generate a maximum cumulative reward thus we have to formally define the reward in our case. With the continuous changes in the environment of a service selection, the transition function P and the contextual reward function R change frequently. As a result, the optimal policy of our system will change over time.

5.1.2 Applying reinforcement learning

The previously introduced service selection model allows to integrate multiple alternative services into each abstract service. During the execution of a service composition, the system can dynamically select the optimal policy that would give the best possible reward.

Theoretically, when the full MDP is known (e.g., R and P are known), the optimal policy can always be calculated. However, this is not true in practice. First of all, we cannot have complete knowledge of state transition functions, because the execution results of a service are not always predictable. Secondly, we do not have enough knowledge about the reward functions of a MDP. The user experience of a service selection is rarely predictable, before interacting with the uncertain environment. In addition, as the environment of a service changes, state transition functions and reward functions change over time.

Because of the above problems, we choose to learn the optimal policy of a MDP at runtime through the use of reinforcement learning.

In this section, we introduce a reinforcement learning schema to select services based on MDP. We first give a brief overview of a reinforcement learning algorithm called Q-learning. Following this we show how to apply reinforcement learning to the context-aware service selection problem.

Q-learning algorithm

In reinforcement learning, the agent task is to learn a MDP policy that maximises the amount of expected reward. As the selection must improve over time, an agent is expected to learn continuously. Therefore, the cumulative reward consisting of starting from a state s_t and following a policy π is defined as:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{1}$$

where r_{t+i} is the expected reward received at each step, and γ is a discount factor. Based on equation (1), the optimal policy is the policy that maximises $V^{\pi}(s_t)$ for all st. Let π^* be the optimal policy. So:

$$\pi^* = argmax_{\pi} V^{\pi}(s_t) \tag{2}$$

To facilitate the learning process, Q-learning uses a Q function to simulate the cumulative reward. Let s be the current state of the agent, a the action taken by the agent and s' be the resulting state of action a, The Q function is:

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma V^*(s)] \tag{3}$$

The Q function represents the best cumulative reward possible for an action to the state s, it represents the quality of the pair (s, a). Based on equation (3), we obtain the optimal policy for each single state:

$$\pi^*(s) = argmax_a Q(s, a) \tag{4}$$

Applying equation (4) to solve the $V^*(s')$ In equation (3), we obtain a recursive definition of $Q(s, a)$:

$$Q(s, a) = \sum_{s'} P(s'|s, a)[R(s'|s, a) + \gamma max_{a'} Q(s', a')] \tag{5}$$

The update is done following the equation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)] \tag{6}$$

This recursive definition of Q forms the basis of the Q-learning algorithm. Q-learning starts with some initial values of $Q(s, a)$, and updates $Q(s, a)$ recursively using the actual reward received by the agent in a trial and error process. The complete learning process is represented in the following algorithm.

Algorithm 1 Context aware Q-learning for service selection

```
   Input  : Abstract Composition, $P$ and $CR$
   Output: Policy $\pi$
1  Initialise $Q \to 0, \forall(s, a) \in (S, A)$;
2  for each episode do
3      $t \leftarrow 0$;
4      Initialise initial state $s_0$;
5      L=set of services  repeat
6          Choose action $a = a_t = \pi(s_t)$;
7          Observe next state $s' = s_{t+1}$ and contextualReward $r = r_t$;
8          Use experience to update $Q$:
9          $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$;
10     until final state ;
11     ;
12 end
13 $\pi$=best policy based on $Q(s, a)$;
```

To successfully apply a reinforcement learning algorithm to a MDP problem, the key is to define the reward function in the learning process. The context of the web service can be used as a measure after each service has been executed. To use various quality of service parameters into a single return value, the reward function in our method is defined as follows:

$$R(s) = \sum_{i=1}^{m} w_i * Qs_i \tag{7}$$

where m means that the service has m quality of service attributes. $\sum_{i=1}^{m} w_i$, w_i is the weight of each QoS parameter according to user preferences for each parameter. Qs_i is the result of the process of normalising Quality of services attributes. The goal of learning is to maximise the total rewards received on selecting a concrete service for each class of abstract service. The reward is defined in formula (7). During the learning process, the agent will determine his optimal policy in the environment, that is, the policy that maximises the sum of the received rewards.

6 E-health scenario

To illustrate how context aware compositions can be built, our approach was tested on a simplified version of the application scenario (emergency monitoring) introduced at the beginning of this paper. The proposed scenario is about managing and monitoring emergency situations of chronic patients (e.g., diabetes, heart attack, epilepsy, etc.). In this scenario, the proposed approach can be used to plan emergency management flows of actions to be executed. The composition has to take into account the surrounding context (location, activity, patient situation, etc.). To this end, the proposed approach is able to generate all the planning processes responding to the user's needs. The assumption is that such cooperating organisations (emergency centre, hospitals, doctors, medical transportation) would have made some or all of their emergency management resources or capabilities available as web services. Each service would have a WSDL (web service description language) and a CA-OWLS context aware semantic description. CA-OWLS description refers to a general domain ontology that describes the emergency e-health context. To accommodate user requests, the application coordinates atomic, disparate services. For example, some of the services in this scenario include:

- alarm service: for notification
- hospital finder: directory of hospitals.
- make app: service to make appointment
- directions service: navigation service
- transport service: medical transportation service
- contact family service: service to contact families and friends in case of a critical condition.

User requests are enriched with context information. For example, the system takes into account the following context types: user location, user activity, and the computing device in use, provided by a context provider. The following two scenarios are described:

- Scenario 1: A patient equipped with e-health sensors to measure his vital signs (blood sugar level for diabetics), while driving he suddenly he feels pain, his personal agent on his device (car infotainment system – CIS) launches a composition request to the emergency centre to handle his situation. The resulting composition is made from the following atomic services: notification service, hospital finder a directory of hospitals and clinics, directions service for navigation.

- Scenario 2: In the second case while the patient is walking near his home and he feels severe pain, he can use smartphone instead of CIS. For this scenario additional services such as a transport service and a contact family service are required in the new composition.

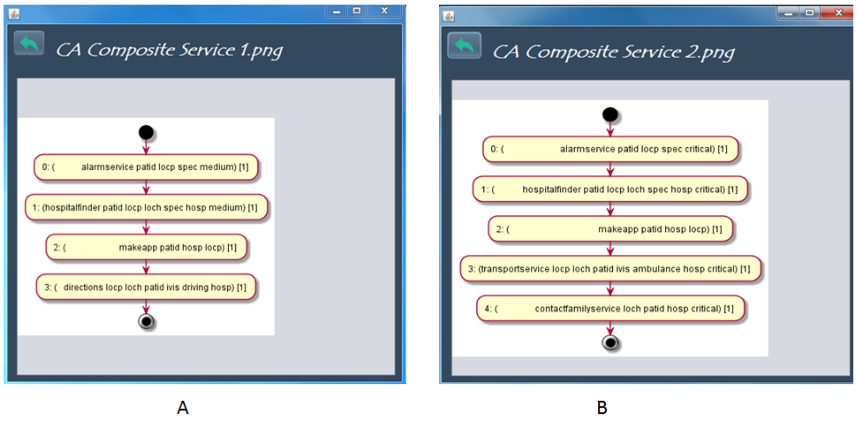
In both cases, the users have the same goal: managing the patient's emergency situation. However, the two requests result in the composite services being constructed from different atomic services, because of the different context in which the requests are submitted. As seen, the result of the planning phase differs as the context changes which makes our approach dynamic. For example, if the planner generates a first plan (scenario 1) and the monitoring component observes a change in the context before the execution of the composite service, a re-planning procedure is launched and a new plan is generated according to the new context. Plans are generated automatically using semantic web services and AI planning techniques.

7 Tool support

In this section, we present the developed tool for automatic context aware service composition called 'CAS Planner'. This tool uses Graphplan as a planning algorithm and requires both a context request and a domain of available services. The tool generates a graphical representation of composites services.

Figure 7 shows an example of plans generated after execution of the composition by the tool 'CAS Planner'. We distinguish the two scenarios discussed previously in the e-health scenario:

Figure 7 Generated plan for the two scenarios (see online version for colours)


CA Composite Service 1.png
0: (alarmservice patid locp spec medium) [1]
1: (hospitalfinder patid locp loch spec hosp medium) [1]
2: (makeapp patid hosp locp) [1]
3: (directions locp loch patid ivis driving hosp) [1]
A
CA Composite Service 2.png
0: (alarmservice patid locp spec critical) [1]
1: (hospitalfinder patid locp loch spec hosp critical) [1]
2: (makeapp patid hosp locp) [1]
3: (transportservice locp loch patid ivis ambulance hosp critical) [1]
4: (contactfamilyservice loch patid hosp critical) [1]
B

7.1 *Experimental evaluation*

In this section, we present the evaluation of the CAS Planner tool. The purpose of the evaluation is to test the ability of planning to work under the following changes:

1 variation in the size of the domain, i.e., the number of available services

2 variation in the size of the contextual query, i.e., the number of adaptation conditions.

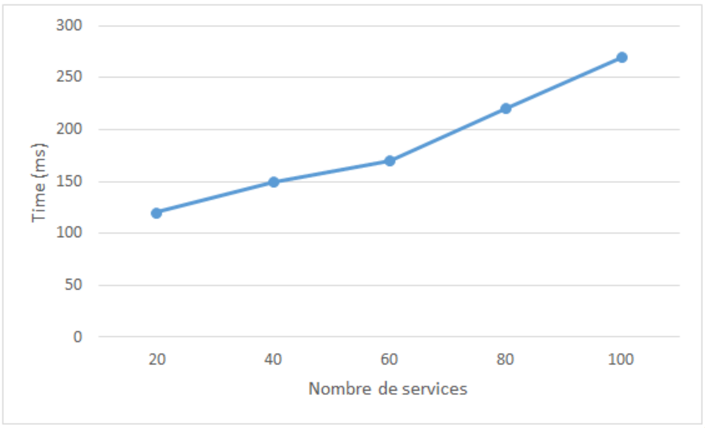
The configuration used for performance consists of:

- CPU: Intel(R) Core(TM) i5-5300U CPU @ 2.30 GHz
- memory: 8.00 GB
- OS: Windows 10 Pro 64-bit
- Java: version 1.8.0_131.

Performance when increasing the size of the service domain

This experiment varies the number of available services in the registry from 20 to 100. For each step we calculate the time required to complete the composition. The process was run ten times to measure the average time. Figure 8 shows the results of this experiment, demonstrating that the composition can evolve to a realistic domain size (100 services), while requiring less than 270 ms of CPU time to compose services.

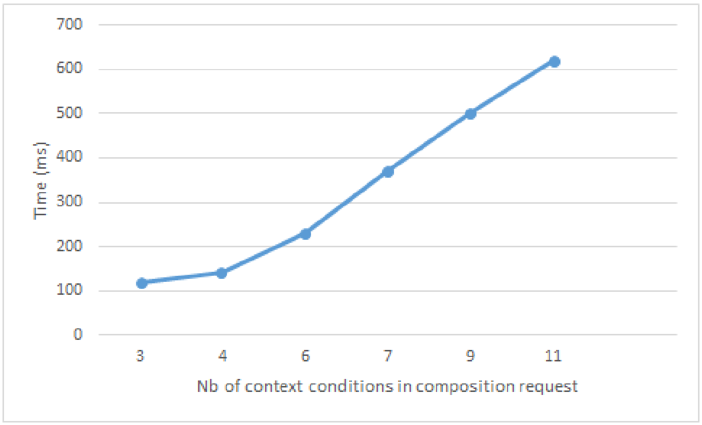
Figure 8 The impact changing the number of services on planning time (see online version for colours)



7.1.1 Performance when increasing the size of contextual conditions

This experiment measures the impact of changing the size of the composition query on the composition time. The number of context conditions in the composition query varies from 3 to 11. Figure 9 shows the impact on runtime of increasing the size of the query. The process was run ten times to measure the average time. We find that our CAS Planner tool performs well by increasing the number of contextual conditions. Thus the execution time for up to 11 contextual conditions is about 0.6 s. This shows that for realistic scenarios the results are satisfying.

Figure 9 The impact changing the number of services on planning time (see online version for colours)



8 Related work

A semantic composition of services aims to automate the process of assembling existing services to fulfil a complex goal. One desirable feature in modern ubiquitous environments is the need to adapt automatically and dynamically to changing contexts. Another important aspect is the need for a generic approach that supports the different stages of service composition lifecycle from defining the request to the execution of the composite service. Another important aspect is the need for a generic approach that supports the different life cycle stages of a service composition. Another important aspect to consider is the dynamic nature of service environment execution, hence the need to use intelligent mechanisms for self-adaptation. Several approaches use semantic web services to automate the composition. METEOR-S (Aggarwal et al., 2004) is based on BPEL and then uses semantics for the discovery and selection of services. SHOP2 (Sirin et al., 2004) uses hierarchical planning for the generation of the composition. The other approaches [WSPlan (Peer, 2006), OWLSXplan (Klusch et al., 2005), PORSCE (Hatzi et al., 2015)] use traditional planning for composition. The discovery is made using the reasoning on the capabilities of the services. However, the major disadvantage of these approaches is the lack of context and adaptation to these changes. Another drawback is that they do not consider the indeterministic aspect of modern computing environments.

The approach proposed by Mrissa et al. (2007) only deals with context mediation and assumes already existing composition schema. The approach proposed in Li et al. (2011) uses aspects for a semantic composition but the dynamic adaptation is not supported and the composition schemas are defined beforehand. The approach proposed in Fki et al. (2017) uses the intentions for adaptation, however the description of context aware services is not specified.

The idea of using MDP in the field of web services is not new, and several approaches have modelled the problem of service composition using Markov decision processes. In Doshi et al. (2004) and Gao et al. (2005), the authors proposed the application of MDP in web services composition, which assumed a fully observable environment and required explicit reward and transitions functions. These requirements are too strict for a real scenario. In general, early research on automatic service composition did not take into account non-functional attributes of services.

To remedy this problem, Wang et al. (2010) deal with the composition of services in a dynamic environment where the rewards and transitions functions are not known. They propose a model for composition that integrates the knowledge of reinforcement learning. However, in this work the authors incorporate several workflows into the same composition and then choose the best composition, they do not use the concept of abstract and concrete services.

Another disadvantage is that the approaches presented previously do not take into account user context and user experience or feedback in the selection of services. So, taking into account the learning of the user experience improves the result of the selection and more important to satisfy the user who is the main actor in any context-oriented approach.

9 Conclusions

In this paper, we presented our idea of transforming the service composition and selection process into a planning and a decision making problem. Thereafter, we modelled the composition of abstract service as an AI planning problem and we modelled the selection problem with Markov decision process. Assuming the uncertainty of today's ubiquitous environment, a reinforcement learning technique is applied to learn the best policies from experience. Thus, our proposed approach can evolve over time ensuring self-adaptivity and transparency for potential user which is the aim of ubiquitous computing.

For future work, we aim to conduct more extensive evaluation to ensure the scalability of the proposed approach, we also project to a full stack approach going from request specifying to generate abstract plans and then applying the proposed approach for selecting executable services.

References

Aggarwal, R., Verma, K., Miller, J. and Milnor, W. (2004) 'Constraint driven web service composition in METEOR-S', *Proceedings: 2004 IEEE International Conference on Services Computing, 2004 (SCC 2004)*, pp.23–30, IEEE.

Alrifai, M. and Risse, T. (2009) 'Combining global optimization with local selection for efficient QoS-aware service composition', *Proceedings of the 18th International Conference on World Wide Web*, pp.881–890, ACM.

Alrifai, M., Risse, T. and Nejdl, W. (2012) 'A hybrid approach for efficient web service composition with end-to-end QoS constraints', *ACM Transactions on the Web (TWEB)*, Vol. 6, No. 2, p.7.

Blum, A.L. and Furst, M.L. (1997) 'Fast planning through planning graph analysis', *Artificial intelligence*, Vol. 90, Nos. 1–2, pp.281–300.

Dey, A. and Abowd, G. (2000) 'Towards a better understanding of context and context-awareness', *CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness*.

Doshi, P., Goodwin, R., Akkiraju, R. and Verma, K. (2004) 'Dynamic workflow composition using Markov decision processes', *Proceedings: IEEE International Conference on Web Services*, pp.576–582, IEEE.

Fki, E., Tazi, S. and Drira, K. (2017) 'Automated and flexible composition based on abstract services for a better adaptation to user intentions', *Future Generation Computer Systems*, Vol. 68, pp.376–390, Special Issue: Advanced Technologies Enabling Adaptive and Collaborative Smart Systems.

Gao, A., Yang, D., Tang, S. and Zhang, M. (2005) 'Web service composition using Markov decision processes', *International Conference on Web-Age Information Management*, pp.308–319, Springer.

Hatzi, O., Nikolaidou, M., Vrakas, D., Bassiliades, N., Anagnostopoulos, D. and Vlahavas, I. (2015) 'Semantically aware web service composition through AI planning', *International Journal on Artificial Intelligence Tools*, Vol. 24, No. 1, p.1450015.

Klusch, M., Gerber, A. and Schmidt, M. (2005) 'Semantic web service composition planning with OWLS-Xplan', *AAAI Fall Symposium on Semantic Web and Agents, USA*.

Li, L., Liu, D. and Bouguettaya, A. (2011) 'Semantic based aspect-oriented programming for context-aware web service composition', *Information Systems*, Vol. 36, No. 3, pp.551–564.

Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D.L., Sirin, E. and Srinivasan, N. (2007) 'Bringing semantics to web services with OWL-S', *World Wide Web*, Vol. 10, No. 3, pp.243–277.

Mrissa, M., Ghedira, C., Benslimane, D., Maamar, Z., Rosenberg, F. and Dustdar, S. (2007) 'A context-based mediation approach to compose semantic web services', *ACM Transactions on Internet Technology (TOIT)*, Vol. 8, No. 1, p.4.

Peer, J. (2006) *Description and Automated Processing of Web Services*, PhD thesis, the University of St. Gallen, Graduate School of Business Administration, Economics, Law and Social Sciences (HSG).

Puterman, M.L. (2014) *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons.

Shin, D-H., Lee, K-H. and Suda, T. (2009) 'Automated generation of composite web services based on functional semantics', *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 7, No. 4, pp.332–343.

Sirin, E., Parsia, B., Wu, D., Hendler, J. and Nau, D. (2004) 'Htn planning for web service composition using SHOP2', *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 1, No. 4, pp.377–396.

Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W. and Bouguettaya, A. (2010) 'Adaptive service composition based on reinforcement learning', *Service-Oriented Computing*, pp.92–107.

Wang, Y-L. and Yu, X-L. (2009) 'Formalization and verification of automatic composition based on pi-calculus for semantic web service', *Second International Symposium on Knowledge Acquisition and Modeling, 2009 (KAM'09)*, Vol. 1, pp.103–106, IEEE.