The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

QoS Based Framework for Ubiquitous Robotic Services Composition*

A. Yachir[1,2,3], K. Tari[1], Y. Amirat[1], A. Chibani[1], and N. Badache[3]

Abstract— **With the growing emergence of ubiquitous computing and networked systems, ubiquitous robotics is becoming an active research domain. The issue of services composition to offer seamless access to a variety of complex services has received widespread attention in recent years. The majority of the proposed approaches have been inspired from the research undertaken jointly on Workflow and AI-based classical planning techniques. However, the traditional AI-based methods assume that the environment is static and the invocation of the services is deterministic. In ubiquitous robotics, services composition is a challenging issue when the execution environment and services are dynamic and the knowledge about their state and context is uncertain. The services composition requires taking into account the parameters of quality of service (QoS) to adapt the composed service to context of the user and the environment, in particular, dealing with failures such as: service invocation failures, network disconnection, sensor failures, context change due to mobility of objects (robots, sensors, etc.), service discovery failures and service execution failures. In this paper, we present a framework which gives ubiquitous robotic system the ability to dynamically compose and deliver ubiquitous services, and to monitor their execution. The main motivation behind the use of services composition is to decrease time and costs to develop integrated complex applications using robots by transforming them from a single task issuer to smart services provider and human companion, without rebuilding each time the robotic system. To address these new challenges, we propose in this paper a new framework for services composition and monitoring, including QoS estimation and Bayesian learning model to deal with the dynamic and uncertain nature of the environment. This framework includes three levels: abstract plan construction, plan execution, and services discovery and re-composition. This approach is tested under USARSim simulator on a prototype of ubiquitous robotic services for assisting an elderly person at home. The obtained results from extensive tests demonstrate clearly the feasibility and efficiency of our approach.**

I. Introduction

With the growing emergence of ambient intelligence, ubiquitous computing, sensor networks and wireless technologies, "ubiquitous robotics" known also under the name of "Network Robot Systems" is becoming an active research domain of autonomous robotics. It targets new innovative applications in which robot technologies will be integrated into ubiquitous ICTnetworks to become the agents of physical action, in smart home, office and public environment. Robots as units capable of coordinating their activities with other devices and systems, moving around, sensing, actuating, decision making and acting will become part of these networks of artefacts, for delivering, individually or collectively as a group, novel capabilities, applications and assistive services for people [1].

In this context, assistive robotics in general and socially assistive robotics in particular targets the enhancement of quality of life for broad populations of users: the elderly, individuals with physical impairments and those in rehabilitation therapy, and individuals with cognitive disabilities and developmental and social disorders [2]. A ubiquitous service robot can be seen as mobile robot that can provide anywhere and any time assistive services dedicated to dependent people, like some everyday life domestic tasks, but also independent health and living monitoring [3].

Ubiquitous robotics framework supporting the paradigm of context-aware services imposes the following main requirements: 1) the ubiquitous robotic system must support ubiquity and heterogeneity of services, sensors and devices and 2) the services must always be available even though there are changes in the service and user situation and/or environment's context. Within this framework, a ubiquitous service robot can be seen like a smart object interoperating with other smart objects and providing various assistive services. Our main objective in this work consists in integrating in a seamless manner all smart objects located in a ubiquitous space (mobiles handsets, sensors/actuators, robots, etc.) using services composition paradigm.

The services composition consists in creating complex service by combining existing atomic services in order to take into account the user's context and environment context. It consists of two main composition processes, vertical and horizontal composition [4]. Vertical composition consists of defining an appropriate plan of services to perform composition task. Horizontal composition process consists of determining the most appropriate service, from a set of functionally equivalent ones for each component of the plan. In this paper, we focus on the horizontal composition which explores the context of services and their probability of response. Our vertical composition approach is already

* Manuscript received March 1, 2009. This work was supported in part by the LISSI Laboratory of Paris12 University (France) and LSI Laboratory of USTHB University (Algeria).

[1] Laboratory of Images, Signals and Intelligent Systems (LISSI), Paris12 University, France (e-mails: karim.tari@univ-paris12.fr, amirat@univ-paris12.fr, achibani@gmail.com).

[2] Department of Computer Science, Military Polytechnic School (EMP), Bordj El Bahri, Algiers, Algeria. (e-mail: a_yachir@yahoo.fr).

[3] USTHB University, Algiers, Algeria(e-mail: badache@mail.cerist.dz)

developed in [5]. It is based on an automatic composition algorithm named FCoSC (*Feasibility and Construction of a Services Composition*) and an automatic discovery algorithm named SDT (*Services Discovery for a Task*).

By the proliferation of the number of available ubiquitous services, the problem of selecting and combining several appropriate services to satisfy user's needs, known as horizontal services composition, becomes a very complex task. The services selection should integrate the quality provided by each service in respect with user's specifications and environment context. Furthermore, the ubiquitous services have a stochastic behaviour and evolve in a dynamic and uncertain environment which is marked by a high mobility of objects and where the information about services is incomplete and change continuously due to the different failures that may occur: request failures, network disconnection, sensor failures, service discovery failures and service execution failures.

To address these concerns, we present in this paper a framework for services composition monitoring. Primary, the relevance degree of each service is estimated using the service context captured from the environment or provided in its OWL-S [6] specification as QoS parameter. Each quality is weighted by the user's preference represented as weight of quality. Secondly, our proposed approach deals with all type of failures in a number of ways. First, it attempts to construct an abstract composition plan using FCoSC algorithm. If no plan is available to satisfy the asked task, a message is returned to the task manager to change this task. Otherwise, the abstract plan is passed to the execution step. Second, the system selects an appropriate service for each plan component. The selected service is then executed and its probability of response is updated using the Bayesian learning. When the parameter of learning precision is exceeded without finding any appropriate service, the system tries to replace the failed service by the discovery of a new abstract service using SDT algorithm. If there is no available service in the registry, the system tries to construct a new plan by re-composition mechanism.

The rest of the paper is organized as follows. Section II gives some related work. Section III introduces the ubiquitous robotic services description. Section IV explains our quality of services estimation algorithm. Section V describes our framework for services composition monitoring. Section VI explains our services composition monitoring algorithm. Section VII describes the detailed scenario for assisting an elderly person. Section VIII shows the experiment results of our approach. Finally, section IX gives the conclusion and future work.

II. Related Work

Several projects for developing and integrating robotic services in Ambient Intelligence spaces (AmI) have been conducted [3][7][8]. We give hereafter the following non-exhaustive list of projects: Robotic Room (Japan) [9], SmartBo (Sweden) [10], ISH (Intelligent Sweet Home, Korea) [11], URC (Ubiquitous Robotic Companion, Korea) [12], PEIS Ecology (Sweden) [13], URUS (Ubiquitous Networking Robotics in Urban Setting, Europe)[14]. In [15], Chibani et al. have proposed a Context-Aware Middleware for supplying intelligent robotic services to users through robots and environmental devices. In [16], Doshi et al. propose an approach based on MDP formalism to model the stochastic behaviour of the Web services. They assume that the services composition problem has already been described as workflow. Mokhtar et al. [17] present an approach for context awareness service composition based on the integration of the workflow in a ubiquitous environment. In [18], Qiu et al. use the *statecharts* and the DAG (*Direct Acyclic Graph*) to represent the composition plans. They suppose that the plan has already been constructed by the HTN approach. Nahrstedt et al. [19] use the idea of the global planning algorithms for the dynamic service composition which calculates an initial plan. Thereafter, this plan is revised if necessary during the execution.

III. Ubiquitous Robotic Services Description

Unlike a traditional robot, a ubiquitous robot can be seen as a set of independent services able to interact with ubiquitous environment. These services, called ubiquitous robotic services, characterize capacities and competences of the robot. The functionalities offered by each service represent an action to perform when some conditions are satisfied. An action acts on some input parameters to produce some output parameter after performing a set of parallel or sequential processes. So, each robot offers a set of services according to its own capacities.

When the number of ubiquitous robots increases, the number of available services becomes more significant. In this case, several robots can have one or more common services. We refer to these services as functionally equivalent services in terms of their input and output parameters. Indeed, two services are considered equivalent if they act in the same manner on the environment in terms of used and produced parameters (input/output). However, two equivalent services can differ in terms of their precondition and offered quality. So, it is necessary to pass through an abstract layer in order to group all the equivalent services. The abstraction consists in defining high summary services in standard and understandable format. Here an example of two equivalent services: let two services of camera, one installed in the Room1, and the other installed in the Room2. The two services take the parameter (position) as input and give parameter (picture) as output. The precondition of the two services depends on the value of the parameter position. For example, if the parameter (position) concerns the Room1, the first service runs successfully and the other fails. Here some other examples of ubiquitous robotic services:

Camera Service: The camera has only two degrees of freedom. So, the camera service needs only two rotations to reach the target. First, it calculates the two angles (α, β) allowing the alignment of the target position with the camera

axis. Second, the camera perform a rotation α, then a rotation β. If the target position is reached after these rotations, the camera sends the picture of the target (Fig.1).

Robot_Move Service: The Robot_Move service computes the difference (d) between the actual robot orientation and a given trajectory. After that, it makes a correction on the robot orientation according to the value of d by performing an appropriate process (Fig.2).

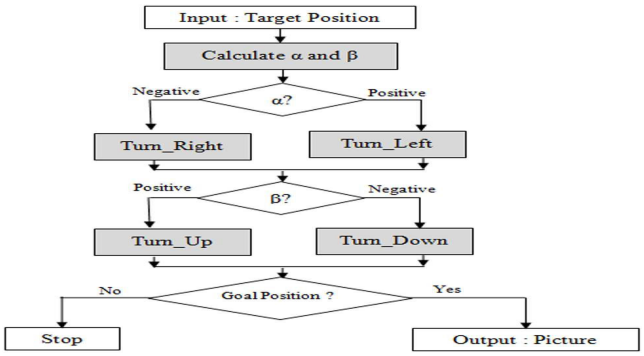

Fig.1. Camera Service

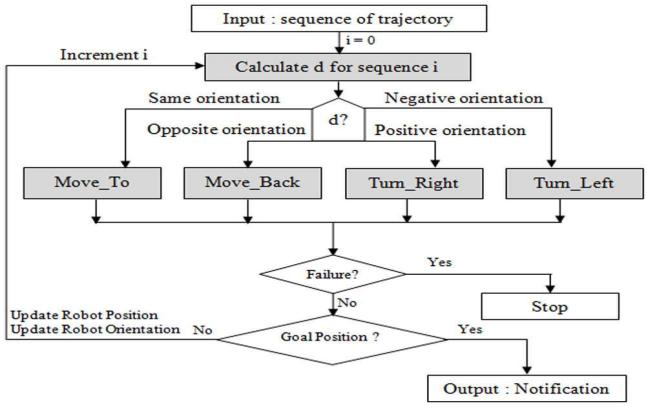

Fig.2. Robot Move Service

IV. PROBABILITY AND QUALITY OF SERVICES ESTIMATION

According to OWL-S specification [6], we have defined formally in [5], the abstract and concrete service. An abstract service is seen as a class of concrete services which possesses the same inputs and outputs. Whereas a concrete service is defined by five components as follows: *(cs_i^{in}: Inputs, cs_i^{out}: Outputs, Prec: Preconditions, Eff: Effects, and QoS: Quality of Service)*. In this work, we introduce a sixth component which is the probability of response of a concrete service in order to take into account the stochastic nature of the ubiquitous environment. We propose a new definition for concrete service as follows:

Concrete service**:* is an action ***a which is achieved by a set of parallel or sequential process. It is defined by six components as follows: *(a_i^{in}: Inputs, a_i^{out}: Outputs, Prec: Preconditions, Eff: Effects, QoS: Quality of Service, and P: Probability of Response).*

1. Probability of response: A concrete service is an action that can be performed. In an uncertain environment, when an action is performed, two types of possible responses can be considered: *positive* with a probability *p* and *negative* with a probability *q = 1-p*:

- ***Positive response**:* the service answers with all its required output parameters. It means that, after execution of the service, all the values will be assigned to its output. These values represent the effect of the executed service.

- ***Negative response**:* the service does not answer with all its required output parameters. Several reasons can be at the origin of this situation i.e. the degradation of the quality of service, such as: network breakdown, wireless sensor/actuator node breakdown, server maintenance, service crash, suppression, modification or momentary unavailability of a service, etc. This situation regroups all the different failures that we have already discussed. Also, when the reason is unknown, the response is considered as negative.

In our model, the probabilities represent the response estimation of each action. Initially, it is possible to have some estimates of these probabilities according to some knowledge about the services such as the availability and reliability. These initial estimations can be inaccurate or completely unknown. Therefore, the introduction of learning mechanisms becomes necessary. It is possible to introduce, at this level, several model of learning. The Bayesian learning model appears well suited in our case.

2. Bayesian Learning: it is a model which associates equal probabilities of response for each action. Thereafter, this model tries to learn them through the interactions with the environment. The responses obtained by the invocation of the services are used to update these probabilities. In this manner, execution and learning will be intercalated. The Bayesian learning algorithm [20] maintains an experience counter, noted *exper,* initialized to 1 for each action. Suppose that after executing an action ***a***, we obtain a positive response. The probabilities *of the action* ***a*** are updated in the following manner:

$$p' = \frac{[p * \exp er] + 1}{\exp er + 1} \qquad \text{(Positive response)}$$

$$q' = 1 - p' \qquad \text{(Negative response)}$$

3. QoS Estimation: The quality of service is closely related to the considered application. Some applications require a high level of security because of the confidentiality of the data which they handle. Some others require a short and bounded computational time such as real time applications. Others also take into account the parameter of energy such as embedded applications, etc. For example, we can define some quality parameter to select appropriate robot service: *Robot_Batterylevel*: measures the actual energetic autonomy of the robot, and *Robot_LightIntesity*: measures the light intensity of the robot headlight [21]. For example, when it is night (environment context), we give more importance for the *Robot_LightIntesity* parameter. This difference can be described by a set of weights associated for each QoS parameter depending on the desired application and

environment context. Therefore, all the concrete services of the same abstract service can be classified according to each QoS parameter. For example: if we are interested in parameter *cost*, the service which has the minimum cost will be classified in first position (0), and the service which has the maximum cost will be classified in last position (N-1). The following matrix (Table1) gives the services classification for each considered parameter of QoS (PQ_i):

TABLE I: MATRIX GIVING CLASSIFICATION OF THE CONCRETE SERVICES

	PQ_1	PQ_2	.....	PQ_k
Service 1	$C_{1,1}$	$C_{1,2}$	.....	$C_{1,k}$
Service 2	$C_{2,1}$	$C_{2,2}$	.....	$C_{2,k}$
...			.....	
Service N	$C_{n,1}$	$C_{n,2}$	.....	$C_{n,k}$

N : Number of concrete services;

PQ_i : Parameter of Quality *i;*

P_k : Weight that reflects the degree of importance of the parameter PQ_k for user and application.

$C_{i,k} \in [0, N-1]$: Class of the service *i* in the column PQ_k ;

$QoS(i) \in [0,1]$: Quality obtained when achieving the service *i*. It is estimated by the following formula:

$$QoS(i) = \frac{\sum_k P_K * (N - C_{i,k})}{N * \sum_k P_k} \quad \text{or} \quad QoS(i) = 1 - \frac{\sum_k P_K * C_{i,k}}{N * \sum_k P_k}$$

4. QoS Estimation Algorithm: Before estimating a quality of a service, the *QoSEA* algorithm determines if all its preconditions are satisfied using the instantiation of all the satisfied parameters provided by *State*. After computing the quality of each satisfied service by the previous formula, the algorithm provides the best one in terms of quality and probability of response. This service becomes the selected action to be executed.

QoS Estimation Algorithm (QoSEA)

Inputs
AS: Abstract Service
State: parameter instantiation
Outputs
Action
Begin
Read the Weights of quality
Read the classification matrix of the action of AS
For each *action a in AS* ***do***
If *(Precondition (a) satisfied)* ***then***
Calculate QoS (a)
End if
End for
$Action = ArgMax(QoS(a) * P(a))_{a \in AS}$
End

V. FRAMEWORK FOR SERVICES COMPOSITION MONITORING

The scheme given in Fig. 3 describes a Framework for services composition monitoring which is designed for flexible and failure-tolerant service composition. This Framework employs a layered design approach to separate the following three stages of the composition process: abstract service composition, plan execution, and abstract service discovery and re-composition.

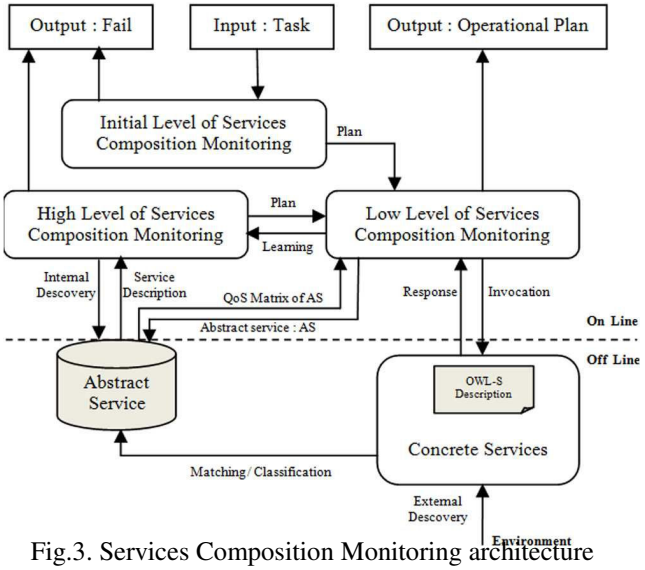

Fig.3. Services Composition Monitoring architecture

1. Initial Level of Services Composition Monitoring (*abstract service composition*): at this level, we construct an abstract plan using FCoSC algorithm. If the asked task cannot be satisfied, due to a missing service, a message is transmitted to the task manager to modify this task or to opt for another one.

2. Low Level of Services Composition Monitoring (*plan execution*): this level manages the services execution failures using the learning mechanism. For each abstract service in abstract composition plan, the aim is to find an optimal action in terms of quality of service and probability of response using *QoSEA* algorithm. After getting the action response, it updates its probability using Bayesian learning. If this response is positive, the system passes to the next abstract service, else it tries finding another action. Once the parameter of learning precision is exceeded without finding any appropriate service, this level passes the composition failure back to the High Level of Services Composition Monitoring.

3. High Level of Services Composition Monitoring (*abstract service discovery and re-composition*): when all the actions of an abstract service fail, this level executes the SDT algorithm to discover a new abstract service at the level where failure occurs. If there is no available service in the registry, the system tries to construct a new plan by re-composition mechanism. The Bayesian learning is also used at this level for managing the services re-composition.

VI. SERVICES COMPOSITION MONITORING ALGORITHM

The previous architecture (Fig.3) is detailed in the following monitoring process (Fig.4):

1. Description: QoS Based Services Composition Monitoring Algorithm is based on the FCoSC algorithm [5] to construct the abstract composition plan. Thereafter, this plan is passed for execution in a sequential order. Each component of this plan constitutes an abstract service. Its action is selected using *QoSEA*. Then, it intercalates the execution of the actions and the Bayesian learning on their probabilities. When the response of an action is negative, this algorithm tries to replace it immediately by the best action of the current abstract service. If all its actions do not answer after exceeding threshold parameter (*local precision learning parameter*), the abstract service is regarded as failed. The SDT algorithm [5] is then executed at this level in order to replace this failed service. The goal here, represent just the output parameter of the failed services. These parameters are saved in the marking table generated by FCoSC (see [5] for more detail about this table). If no service is available, the re-composition mechanism is then executed to generate new plan and the probability of re-composition *P (Re-composition)* is decreased using the Bayesian learning. A *global precision learning parameter* is defined at this level to control the re-composition process.

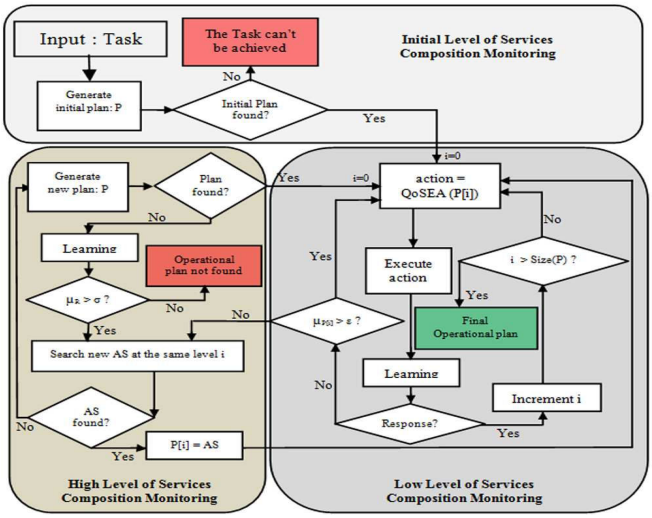

Fig.4. Services Composition Monitoring Process

2. Complexity: Let *S* the number of component of the abstract plan, and assume that *A* is the number of actions included in each component. In the worst case, the monitoring algorithm explores all the actions of each component. By integrating the learning mechanism, we suppose that, in the low level monitoring each action is explored L1 times and in the high level monitoring the discovery and re-composition are done L2 times. So, the complexity becomes O(*S*A*L1*L2*). But in practice, S represents the size of an abstract plan. It is negligible compared to the number of the actions. Then, computational complexity of the monitoring algorithm is on O (A*L1*L2). We have assumed that A is very large. So, L2 is negligible compared to L1 because that, it is more probable to found an action by learning and consequently we avoid the high level monitoring. The complexity is then on O (A*L1).

QoS Based Services Composition Monitoring Algorithm

Inputs
T: Task (inputs, outputs), TAS: Table of Abstract Services
ε : local precision learning parameter
μ : measures the distance between two probabilities
σ : global precision learning parameter
Outputs
TaskFeasibility, CompositionPlan
Begin
State = inputs of T; Initialize i; TaskFeasibility = True
If *(FCoSC (**in**: T, TAS; **out**: Plan))* ***then***
  CompositionPlan = Plan
  PlanSize = SizeOfCompositionPlan
  While *(i ≤ PlanSize) and (TaskFeasibility = True)* ***do***
    AS = CompositionPlan[i]
    Goal = {p / Source (p) = AS} from marking table
    *QoSEA (**in**: AS, State; **out**: action)*
    Execute action
    Update P (action) using Bayesian Learning
    If *(response (action) = negative) then*
      If *(* $\|\mu_{AS}\|_{\infty} < \varepsilon$ ***then***
        Critic = True
        TAS = TAS – {AS}
        Repeat
          Update P (Re-composition) using Bayesian learning
          If *SDT (**in**: State, Goal; **out**: NewAS)*
            CompositionPlan[i] = NewAS
            Critic = False
          Else
            If *(FCoSC (**in**: T, TAS; **out**: NewPlan))*
            CompositionPlan = NewPlan
            PlanSize = SizeOfCompositionPlan
            State = inputs of T
            Initialize i
            Critic = False
          End if
        End if
        Until *(Critic = False) or (* $\|\mu_{\text{Re-composition}}\|_{\infty} < \sigma$ *)*
        If *(Critic = True)* ***then***
          TaskFeasibility = False
        End if
      End if
    Else
      Increment i
      State = State U Goal
    End if
  End While
Else
  TaskFeasibility = False;
End if
End

VII. MOTIVATING EXAMPLE

1. Environment description: The test scenario is implemented with Java and integrated in USARSim environment. USARSim is a high fidelity simulation of urban search and rescue robots and environments intended as a research tool for the study of human-robot interaction (HRI) and multi-robot coordination [21]. The robot used in this scenario is a 4-wheel drive all-terrain pioneer robot called P2AT. It is equipped with a PTZ camera, INU (Inertial Navigation Unit) Sensor, odometry sensor, RFID Sensor (RFID Reader on the P2AT robot, RFID Tags on the person), Human Motion Sensor, Sound Sensor and 16 Sonar

sensors equally distributed around it, enabling the robot to sense obstacles in all directions.

The test scenario implies several mobile robots P2AT which offer different concrete services. A number of other concrete services are available in the ubiquitous environment. For example, several concrete services for person indoor localization using RSSI (received signal strength indicator) based Localization Scheme are associated for each group of 3 wireless sensors [22]. Each one of these services possesses its own field of visibility and optimal use of its proper wireless sensors. For example, a living room is equipped by a group of 3 wireless sensors to localize a person. This service fails if the person leaves the visibility field of the 3 wireless sensors.

The composition system receives all the data (parameters) collected by the various sensors about the person. After analysis of these data, selected robot follows the giving trajectory to reach the person. While arriving successfully, the robot sends a notification to the robotic assistance service which takes the control of the robot and assists remotely the person waiting for the social assistance.

TABLE II : TABLE OF CONCRETE SERVICES

AS	CS	Description
AS_{12}	a_1	Human_Location_By_Motion
	a_2	Human_Location_By_Sound
	a_3	Human_Location_By_RSSI
AS_2	a_4	Human_Location_By_RfidTagsId1
	a_5	Human_Location_By_RfidTagsId2
	a_6	Human_Location_By_RfidTagsId3
	a_7	Human_Location_By_RfidTagsId4
AS_6	a_8	Room_Camera
	a_9	Kitchen_Camera
	a_{10}	Corridor_Camera
	a_{11}	Outside_Camera
	a_{12}	Robot1_Camera
	a_{13}	Robot2_Camera
AS_4	a_{14}	HumanSound_Loudness&Duration_Test1
	a_{15}	HumanSound_Loudness&Duration_Test2
	a_{16}	HumanSound_Loudness&Duration_Test3
AS_3	a_{17}	HumanMotion_Probability_Check1
	a_{18}	HumanMotion_Probability_Check2
	a_{19}	HumanMotion_Probability_Check3
AS_8	a_{20}	Robot1_Move
	a_{21}	Robot2_Move
AS_9	a_{22}	Local_Alarm_Operator
	a_{23}	Central_Alarm_Operator
AS_{11}	a_{24}	Neighbours_Contact
	a_{25}	Family_Contact
	a_{26}	Hospital_Contact
	a_{27}	Emergency
AS_1	a_{28}	Robot_State_By_Odometry&INUSensor1
	a_{29}	Robot_State_By_Odometry&INUSensor2
AS_5	a_{30}	Optimal_Trajectory_Computing
	a_{31}	Quick_Trajectory_Computing
AS_7	a_{32}	Human_State_Analysis1
	a_{33}	Human_State_Analysis2
AS_{13}	a_{34}	Human_MotionSound_Analysis1
	a_{35}	Human_MotionSound_Analysis2
AS_{10}	a_{36}	Local_Robotic_assistance
	a_{37}	Remote_Robotic_assistance

TABLE III: TABLE OF PARAMETERS

Parameters	Description
a	Robot_Data
b	Environment_Map
c	Position_Data
d	Human_Motion_Data
e	Human_Sound_Data
f	Robot_State_Data
g	Human_Location
h	Human_Motion_Probability
i	Human_Sound_Loudness
j	Human_Sound_Duration
k	Trajectory
l	Human_Map
m	Human_State_Analysis
n	Notification
o	Alert
p	Human_Id

TABLE IV: TABLE OF ABSTRACT SERVICES

Abstract Services Description	Input	Output
Robot_State (AS_1)	a	f
Read_RfidTags_Memory (AS_2)	c	g, p
Human_Motion_Analysis (AS_3)	d	h
Human_Sound_Analysis (AS_4)	e	i, j
Trajectory_Computing(AS_5)	b, f, g	k
Camera (AS_6)	g	l
Human_State_Analysis (AS_7)	h, i, j	m
Robot_Control(AS_8)	k	n
Remote_Alarm_Operator (AS_9)	m	o
Robotic_Assistance(AS_{10})	o, n	-
Social_Assistance(AS_{11})	o	-
Sensor_Human_Location (AS_{12})	c	g
Human_Motion&Sound_Analysis (AS_{13})	l, m	o

2. Context description: the context in our experience takes the QoS of all the used concrete services. Here, we have considered three parameters of QoS : *cost*, *response time*, and *energy*. For example, the abstract service AS_2 possesses 4 concrete services of location by RfidTags. We have two kinds of RfidTags: passive and active. Active RFID is continuously powered, whether in the reader field or not. It uses an internal power source (battery) within the tag, whereas Passive RFID uses energy transferred from the reader via RF. Moreover, Passive RFID operation takes much time because it requires a very strong signal from the reader to power the tag, whereas Active RFID can generate high-level signals back to the reader, driven from its internal power source. So, it has a quick response then passive RFID. For the cost, Passive RFID is cheaper then Active RFID. So, we obtain the following classification of the concrete services of location by RfidTags in the abstract service AS_2:

TABLE V : TABLE OF RFIDTAGS QOS

	$PQ_1(cost)$	$PQ_2(time)$	$PQ_3(energy)$	QoS
a_4 *(passive)*	*0*	*2*	*0*	***0.833***
a_5 *(active)*	*2*	*1*	*1*	*0.625*
a_6 *(active)*	*2*	*0*	*1*	*0.708*
a_7 *(passive)*	*1*	*2*	*0*	*0.708*

In our experimentation, we give more importance for *cost* (P_1: weight = 3), *response time* (P_2: weight = 2), and finally *energy* (P_3: weight = 1). First, *QoSEA* algorithm cheek the

precondition of each action: the RfidTags reader should be in the field of the RfidTags. Second, it chooses one action according to its estimated *QoS* by the formula seen in section IV. With the same manner, the context is estimated for all the actions of the other abstract services. Initially, the response probabilities are the same for all the actions. Thereafter, these probabilities are updated according to the availability of the invoked actions.

VIII. IMPLEMENTATION AND TESTS

1. Motivation: We have used USARsim to implement a prototype of our algorithm in a simulation environment. Since the number of provided services in USARsim is limited, it is not well appropriate for testing the efficiency and scalability of our approach in ubiquitous environment. So, the idea is to provide a prototype which run well under USARsim, and randomly generate other services with the same characteristics in term of QoS and probability. Therefore, our solution can be used on two steps: off line step, in which virtual tasks are generated to learn the services and on line step which exploit learned services to receive and process real tasks. Also, in the real setting, the input and output of the services should be processed and matched in the off line step (as shown in Fig.3) because this process takes much time. After parameters matching, it is easy to associate a unique identifier for each set of identical parameters. Our algorithm manipulates only the parameter identifier in the on line step. In this experience, all the input/output parameters of the services and the tasks are randomly generated. Also, maximum number of input and output of each service and task is randomly generated between 1 and 10. This chose is motivated by the fact that, in real word, the services need just a few number of parameter to perform an action, and also it provides a few numbers of outputs.

2. Requirements: All experiments were performed on a PC with the HP Core 2 Duo with 2039Mo RAM, running Windows Vista at 1.8GHz. Our algorithm was implemented in Java under Eclipse Platform. The tests have been performed over 100000 randomly generated concrete services separated into 1000 abstract services (classes). The number of QoS parameter is fixed to 20. We have also generated random initial probabilities and QoS for each concrete service. To measure the parameter μ , we have used a metric called KLD *(Kullbach Leibler Divergence)* [16]. The KLD between two probability distributions p and q is defined as follows:

$$D(p//q) = \sum_x p(x)\log_2 \frac{p(x)}{q(x)} \text{ and } D(p//q) = 0 \text{ if } p=q$$

3. Success rate: it measures the quality of monitoring by computing the number of really achievable plan from a set of feasible abstract plan. Fig. 5 shows that, approximately, the entire giving abstract composition plans are correctly monitored in their execution. The success rate is measured by the number (%) of achieved plan. Its evolution is shown by the red plot. It is about 90% in average, and it tends to 100%. This rate demonstrates the quality of monitoring failures during the execution step.

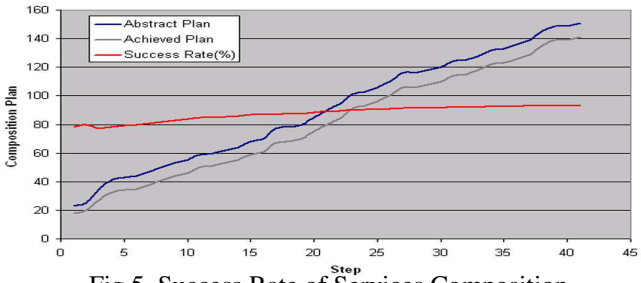

Fig.5. Success Rate of Services Composition

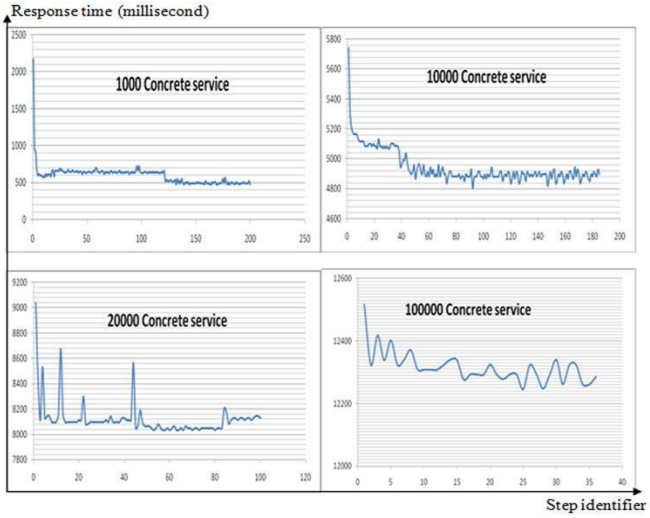

Fig.6. Convergence of Services Composition

4. Convergence of learning: to perform our tests, we try to learn, step by step, the true probabilities of the concrete services. At each step, we randomly generate 1000 tasks. Each task will invoke the services which appear in its composition plan, and update their probabilities of response. For each step, we measure the necessary time (*in millisecond*) to perform the 1000 generated tasks. Since the initial probabilities are completely unknown (*random*), the response time in the first step is high. In average, this time decreases as the number of steps increases. Because the number of learned services increases and so the time taken by each task to learn its services is reduced. We show also that, oscillations appear in the response time evolution. These oscillations (fluctuations) are due to the fact that, the generated tasks are random, and each task can invoke services that are not already updated, so it takes much time in its execution. Since the initial probabilities are random, the oscillations peaks are in their maximum at the beginning of the learning process. But after some initial fluctuations, we note a clear reduction on the amplitude of the peaks because there are less updates on the probabilities. Therefore, the running time is stabilized and can be estimated as the average of the last fluctuations. We show also in fig.6 that the large of the oscillations increases by increasing the number of the concrete services. For a small number of concrete services, each task possesses a limited

number of services. So, when the invoked services fails, the learning process is done until obtaining positive response or the threshold parameter is reached. But, when the number of services is large, each tasks possess a large number of concrete services, it is more probable to choose appropriate services (using *QoSEA algorithm*) for each task without learning. So, the oscillation takes more time to reach its maximum, by the fact, it become large. But, we note that the maximum of the oscillation for a small number of concrete services is less comparing to a large number of services. Because that, our algorithm takes more time in searching on large set of services. Moreover, our proposed method can deal with well-known problems such as local maximum problems. This kind of problems occurs when all the actions of an abstract service fail after exceeding threshold parameter. In this case, our method tries to replace just the non achievable abstract service. Here, another local maximum problem can occur if no abstract service is available. This case is monitored by the re-composition mechanism. Also, at this level, a local maximum problem can occur if no plan is found for the task after exceeding threshold parameter. In this case, our algorithm stops and declares that the task can't be achieved. For example, in our motivating scenario, the abstract service AS_2 possesses 4 concrete services of location by RfidTags. Suppose that we have failure of all these services. So, it becomes impossible to get the location of the person using AS_2. In this case, we execute the SDT algorithm for services discovery at the level where the failure occurred. So, to replace AS_2, the SDT algorithm can find and return the abstract service AS_{12} which is the localization by sound, movements and RSSI. If this discovery fails, the system tries a new re-composition to find a new plan which can be successfully performed.

IX. CONCLUSION

In this paper, we have presented a Framework for robotic services composition and QoS based services selection. It is based on the Bayesian learning to take into account the stochastic nature of services which is modelled by probabilities of response. The Bayesian learning serves to learn these probabilities through the interaction with the environment. This approach integrates also the QoS parameter to adapt the composed service to the context of the user and the environment. Our approach is based on the FCoSC algorithm to construct the abstract composition plan. This plan is executed with learning, and if there is failure of all the actions of a giving abstract service, the plan will be affected. In this case, the SDT algorithm will be performed at the same level of composition, i.e. at the level where the failure occurs. If there are no discovered services, the system attempts dynamic re-composition. The proposed framework has been successfully tested for implementing a prototype of ubiquitous robotic services for assisting an elderly person. Extensive tests measuring success rate and convergence of learning have also shown the efficiency of our approach. Actually, we are testing the proposed framework by integrating Markovian Decision Process (MDP).

REFERENCES

[1] ftp://ftp.cordis.europa.eu/pub/ist/docs/europ/rob-plat-4.pdf

[2] A. Tapus, M. J. Mataric, B. Scassellati. "The Grand Challenges in Socially Assistive Robotics". IEEE Robotics and Automation Magazine, 14(1), March 2007.

[3] P. Harmo, T. Taipalus, J. Knuuttila, J. Vallet, A. Halme. "Needs and Solutions- Home Automation and Service Robots for the Elderly and Disabled". In Proc. Of the IEEE IROS 2005, pp. 2721 – 2726. 2005.

[4] A.B. Hassine, S. Matsubara, and T. Ishida. "A Constraint-based Approach to Horizontal Web Service Composition". Language Grid Project, National Institute of Information and Communications Technology, NTT Communication Science Laboratories, and Kyoto University. 2006.

[5] A. Yachir, K. Tari, A. Chibani, and Y. Amirat. "Towards an Automatic Approach for Ubiquitous Robotic Services Composition ". In Proc. Of the IEEE IROS 2008 conference. pp. 3717-3724. 2008.

[6] The OWL Services Coalition. "OWL-S: Semantic Markup for Web Services" (http://www.daml.org/services/owl-s/1.0/owl-s.html), 2003.

[7] D. H. Stefanov, Z. Bien, W.C. Bang. "The smart house for older persons and persons with physical disabilities: Structure, technology arrangements, and perspectives". IEEE Trans. on Neural Systems and Rehabilitation Engineering, Vol. 12, No. 2, 228-250, 2004.

[8] N. Hagita, V. Kumar, A. Sanfeliu, H. Ishiguro. "Network Robot Systems:Toward Intelligent Robotic Systems Integrated with Environments".WorkShop Notes/ICRA 2006, May 19, 2006, Orlando.

[9] T. Nakata, T. Sato, H. Mizoguchi, T. Mori. "Synthesis of robot-to-human expressive behavior for human-robot symbiosis". In Proc. of the 1996 IEEE/IROS 1996, pp. 1608-1613, 1996.

[10] G. Elger and B. Furugren. "SmartBO: An ICT and computer-based demonstration home for disabled". In Proc. of the 3rd TIDE Congress, Helsinki, Finland, 1998.

[11] K. H. Park, Z. Bien, J. J. Lee, B. K. Kim, J. T. Lim, J. O. Kim, H. Lee, D. H. Stefanov, D. J. Kim, J. W. Jung, J. H. Do, K.H. Seo, C. H. Kim, W.G. Song, W. J. Lee. "Robotic smart house to assist people with movement disabilities, Autonomous Robots Journal, 2004 Volume 2, 2, Number 2 / February, 2007, pp. 183-198.

[12] Y. G. Ha, J. C. Sohn, Y. J. Cho. "Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic Web services technology". In Proc. Of the IEEE IROS 2005, pp. 3947 – 3952.2005.

[13] M. Gritti, M. Broxvall and A. Saffiotti. "Reactive self-configuration of ecology of robots". In Proc. of the ICRA-07 Workshop on Network Robot Systems, pp. 49-56. Rome, Italy, 2007.

[14] A. Sanfeliu, N. Hagita, A. Saffiotti : «Network robot systems» Robotics and Autonomous Systems 56(2008) 793–797, 2008 Elsevier.

[15] A. Chibani, K. Djouani, Y. Amirat, "Context Aware Service Agents for Ubiquitous Robots Applications". In Proc. Of the IEEE URAI 2007, Nov. 22-24, 2007.

[16] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. "Dynamic Workflow Composition using Markov Decision Processes". Int. Journal of Web Services Research, 2(1): 1-17, Jan-March 2005.

[17] S. B. Mokhtar, N. Georgantas, and V. Issarny. "Ad hoc composition of user tasks in pervasive computing environments". In Proc. of the 4th workshop on software composition, LNCS3628. 2005.

[18] L. Qiu, Z. Shi, F. Lin. "Context Optimization of AI planning for Services Composition". In Proc. Of the IEEE International Conference on e-Business Engineering (ICEBE'06), 2006.

[19] K. Nahrstedt, D. Xu, D. Wichadakul, and B.Li. "Qos-aware middleware for ubiquitous and heterogeneous environments". IEEE Communications magazine, 39(11):2–10, 2001.

[20] D. Spiegelhalter, A. Dawid, S. Lauritzen,and R. Cowell. "Bayesian analysis in expert system". 1993.

[21] J. Wang, S. Balakirsky. "USARSim Manual V3.1.1: A Game-based Simulation of mobile robots". University of Pittsburgh, National Institute of Standards (NIST), USA. 2007.

[22] M. Sugano, T. Kawazoe, Y. Ohta, and M. Murata, "Indoor Localization System using RSSI Measurement of Wireless Sensor Network based on ZigBee Standard," In. Proc. of Wireless Sensor Networks, vol. 7, pp. 54–69, 2006.