This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

A New Approach for the Composition of Adaptive Pervasive Systems

Francisco Cervantes, *Member, IEEE*, Félix Ramos, *Member, IEEE*, Luis F. Gutiérrez, *Member, IEEE*, Michel Occello, *Member, IEEE*, and Jean-Paul Jamont, *Member, IEEE*

***Abstract*—In the real world, the assumption that, once generated, a pervasive system of systems and the services it provides will remain static is false. Systems can leave or enter, and service availability in systems can change along with the environmental conditions. The dynamic composition and adaptation of pervasive systems of systems enable them to expand their functionality through leverage resources in the user vicinity. Most of the adaptation approaches based on *ad hoc* networks use service adaptation created from scratch. That is, each time the environment changes, the system starts the composition process over again, resulting in the depletion of system resources and network capacity. We present a protocol for dynamic composition of a pervasive system of systems and their services in *ad hoc* networks. Our proposal uses a dynamic and distributed constraint satisfaction approach to establish the pervasive-system-of-systems requirements. In this paper, we deal with spatial and temporal constraints. However, any other kind of restriction, for instance, quality, can be addressed in the same manner. Additionally, a heuristic allowing recomposing services based on the identification of service components disqualified due to the dynamism of the environment is presented. Thus, the adaptation of services is not necessarily done from scratch. Finally, we present simulations that show the performance of our proposal regarding the nearest work in the state of the art, both in time and in number of messages consumed for adapting services.**

***Index Terms*—Constraint satisfaction, service adaptation, service composition, system of systems.**

I. Introduction

THE number of mobile systems with constant network connectivity is growing steadily, which has led to the increasing availability of resources (software and hardware). These systems are prompting a user demand for system functionality to remaining available in every time and place. This is driving service-oriented computing and network technology into everyday systems (such as medical, domestic, and surveillance systems), making pervasive systems a reality. A pervasive system integrates network devices with people and their environment; these devices can range from small sensors to dynamic and powerful devices [1]. Usually, developers of pervasive systems know all the elements that comprise the system and its functionality at design time. However, the high mobility of users and their devices between different environments poses the challenge of building and adapting the functionality of pervasive systems at runtime. This is encouraging the emergence of new approaches to address the automatic composition and adaptation of pervasive systems.

An open challenge is how to achieve the automatic composition and adaptation of pervasive systems and their services (based on available systems in the user's local environment and considering user mobility). In our case, we propose an approach based on the paradigm of systems of systems.

Thus, the aim is to automate pervasive system composition and the adaptation process through the discovery of new participant systems in the local environment. Several approaches have been adopted from the service-oriented architecture (SOA) research field, ranging from the automatic configuration of system infrastructure to enterprise interoperability [2]–[4]. Also, the need for dynamic and decentralized solutions in areas such as medicine, security, and crowd evacuation has promoted research into service composition in mobile *ad hoc* networks [5], [6]. However, these approaches rely on a dedicated and reliable infrastructure. They tend to provide basic interoperability through centralized approaches, resulting in little or no suitability for mobile systems in dynamic environments.

Adaptation is a significant topic in the context of service-oriented systems [7], [8]. However, little attention has been given to adapting services instead of restarting the service composition process each time the environment changes. That is, finding "adaptations of previous solutions" is not considered a priority. The effect is that system resources such as battery, processing power, and network bandwidth are unnecessarily depleted.

The problem of composition adaptation, in general, can be proven to be NP-complete [9]. Research in this field follows several trends. A way to characterize these trends is to organize them along several dimensions, where each dimension represents one or more facets of the service-adaptation problem. In the literature addressing this issue, there are several characterizations. However, most of these characterizations, such as [10], are focused on SOA and do not address issues about the environment and system participants.

Starting from a perspective of pervasive systems, and based on the dimensions proposed by Cardellini *et al.* [11], we

Manuscript received June 1, 2016; revised September 15, 2016 and November 30, 2016; accepted January 7, 2017. This work was supported by the National Council on Science and Technology (CONACyT).

F. Cervantes and L. F. Gutiérrez are with the Departamento de Electrónica, Sistemas e Informática (DESI), Instituto Tecnologico y de Estudios Superiores de Occidente, 45604 Tlaquepaque, Mexico (e-mail: fcervantes@iteso.mx; lgutierrez@iteso.mx).

F. Ramos is with the Department of Computer Science, CINVESTAV Unidad Guadalajara, 45019 Zapopan, Mexico (e-mail: framos@gdl.cinvestav.mx).

M. Occello and J.-P. Jamont are with the Laboratoire de Conception et d'Intégration des Systèmes, Université Grenoble Alpes, 26000 Valence, France (e-mail: michel.occello@lcis.grenoble-inp.fr; jean-paul.jamont@lcis.grenoble-inp.fr).

Digital Object Identifier 10.1109/JSYST.2017.2655031

1937-9234 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

characterized the adaptation problem by providing an answer to the following questions.

1) Why should a system be adapted? The primary objective of pervasive systems is to provide services that are required by the user, despite variations in its environment; this is achieved using adaptation. Adaptation aims to make the pervasive system capable of fulfilling functional and/or quality-of-service (QoS) requirements, despite changes in its environment [12]. Our focus is on functional requirements concerning the availability of services to fulfill user needs.
2) When must adaptation actions be executed? Adaptation must be carried out at different stages in a system's life cycle. The existing approaches can be placed between design time and runtime stages [13], [14]. In the pervasive system domain, there is a special interest in performing the system adaptation during runtime [15], [16]. Within this stage, we can distinguish between reactive and proactive approaches. In the proactive approach, systems predict possible future changes to perform the adaptation. In the reactive approach, systems are adapted after the detection of changes. Currently, our approach adopts a reactive mode placed in runtime.
3) Where does the adaptation occur? Broadly speaking, pervasive systems may be deployed in static and dynamic environments: static or dynamic. In the pervasive system domain, there is a growing emphasis on building systems tolerant of unexpected changes in their environment [17]–[19]. Within this type of dynamic environments, we may further distinguish between closed and open environments. In closed environments, all participant systems are known at the system design stage, while in open environments, the participants may move from one environment to another. The majority of adaptation approaches are designed for closed environments, using servers for service registration, discovery, composition, and adaptation, and assuming reliable participant systems, for example, [20]–[22]. These approaches often involve preconfigured adaptation managers. They do not cater to highly pervasive, mobile, and *ad hoc* environments, leaving themselves exposed to issues such as central point of failure, mobility, and fault management. This paper adopts an adaptation approach for dynamic environments.
4) Who participates in the adaptation process? This question has to do with the kind of system participants during the adaptation process (for example, autonomous systems, or subordinate and manager systems), that is, the regime that manages the service-adaptation process. In the case of multiple systems, their adaptation could be under the control of a single authority or under the control of multiple authorities (they could be cooperative or/and competitive). We are interested in cooperative autonomous participants, that is, systems that need to cooperate to achieve their objectives, while preserving the autonomy (i.e., each system is the only one that can decide whether to cooperate or not). In our approach, each system is motivated to cooperate with others, due to its need for other systems to cooperate with it.

Diverse approaches have been proposed for service adaptation, for example, dynamic service selection, dynamic coordination pattern selection, etc. However, most of the authors assume that resources are always available, so only consider the dynamic degradation of service quality. In this respect, this paper aims to achieve adaptation of the systems and the services they provide based on a mutual cooperation approach for a distributed selection of participant systems and their services while preserving the principle of autonomy (i.e., any system in the user environment can be forced to be part of a pervasive systems of systems (PSoSs) and only each participating system may decide whether or not to provide a service).

Moreover, *ad hoc* networks are composed of wireless systems, some of which may be mobile, and networks are dynamically created without a dedicated infrastructure or administration, that is, they constitute highly dynamic environments where systems can enter or leave the environment. Therefore, pervasive systems should be user centered, that is, the pervasive systems should be available for a user anywhere and anytime. It is clear that to get this objective, properties like the adaptation according to user mobility and changes in the environment must be achieved. The contributions and objectives of this paper are as follows.

1) An adaptive pervasive system model based on autonomous mobile systems.
2) A dynamic constraint satisfaction problem model for adaptation of the PSoS and their services.
3) A heuristic for an adaptation of dynamic partial solutions (DPSs) for composite services.

The rest of this paper is organized as follows. Section II describes a PSoS. Section III models the composition of a PSoS and the services it provides as a dynamic and distributed constraint satisfaction problem. Section IV presents a service composition protocol. Section V presents an adaptation heuristic. Section VI shows the experimental evaluations. Section VII discusses some related work. Finally, Section VIII gives some concluding remarks and suggests future work.

II. From Pervasive Systems to PSoSs

We can find pervasive systems in many areas of life such as healthcare, sport, and education. Some examples of these applications are tourist guides [23], smart offices [24], and monitoring of the natural world [25]. Most of the current pervasive systems are focused on a single location and are usually based on a hardware layer to acquire changes in the environment, such as user behavior and appliances states, a middleware layer to manage input data from the environment and make adaptation decisions, and an application layer to prompt actuators to execute decisions [26]. However, user mobility and the user demand for system functionality to remaining available at every time and in every place are raising new challenges such as dynamic composition and adaptation, which cannot be faced by classical approaches around a single location. It is raising the need for new approaches to dynamic composition and adaptation of pervasive systems in the user vicinity.

To achieve this pervasive system composition and adaptation, we propose to build pervasive systems based on the system of

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

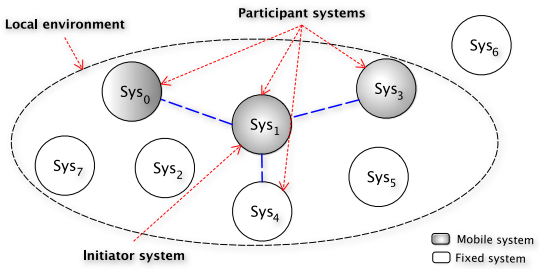

Fig. 1. Abstract model of a PSoS.

the system paradigm to create PSoSs. A PSoS is a federation of participants, where each participant can be considered an individual system, the participants are autonomous (operationally managerially independent) and geographically distributed, and the federation behavior is emergent and its behavior evolves [27]. Also, each system could be fixed or mobile, and, together, they have a network connection and may provide their functionality as public services in their local environment. These enable the composition and adaptation of complex PSoSs. For the sake of simplicity in the rest of this paper, we deal with service composition and adaptation using only one request at a time. However, our proposal can deal with multiple requests simultaneously, using techniques similar to those used in operating systems for scheduling processes.

We define a PSoS based on the following four elements.

1) *Initiator system*: The initiator system is the system in which a user requirement has become active. A system-task (STask) is generated from a user requirement at the application level. An STask is described as duple $\langle TR, TC\rangle$, where $TR = \{tr_i | i = 1, ..., n\}$ is the finite set of services needed to fulfill the task. $TC = \{tc_i | i = 1, ..., m\}$ represents a finite set of constraints over the set TR to allow the intended behavior of a composed pervasive service and exist a subset TP of TC representing a set of preferences managed by the user. The initiator system triggers the PSoS emergence in its local environment, in order to fulfill the user requirement (see Fig. 3).
2) *Participant systems*: The participants of a PSoS are all the autonomous systems $\{\text{sys}_1, \text{sys}_2, ..., \text{sys}_N\}$ that have chosen to participate in providing some composite service. Participants may be fixed or mobile systems from the local environment of the initiator system (see Fig. 1).
3) *Service context*: The service context provides a description of the functionalities of a PSoS and an interface to interact with other PSoSs. The context of a PSoS is defined as the union of the services of its participant systems

$$\text{Service Context} = \bigcup_{i=1}^{N} \text{sys}_i.\text{services} \tag{1}$$

where N is the number of current participant systems.
4) *Interaction state*: The interaction state of a PSoS is the union of all interaction states of its participant systems. Each of these participant systems possesses a part of the global interaction state; we have named it participant system $view$. Thus, the PSoS interaction state is defined as

$$\text{Interaction State} = \bigcup_{i=1}^{N} \text{sys}_i.\text{view} \tag{2}$$

where N is the number of current participant systems.

This kind of PSoS, unlike classical approaches in the literature, such as [27]–[30], is based on small systems over mobile ad hoc networks and is user centered. This PSoS emerges around initiator systems, where a user requirement has become active. Fig. 3 shows the conceptual model of our PSoS context.

III. Dynamic-disCSP Framework

In order to meet the PSoSs outlined in the previous section, we extend the model proposed by Karmouch and Nayak [31] to deal with environment dynamism, that is, one or more nodes may left the network or become unavailable. In the same way, we extend the model in order to deal with dynamism (the user mobility between different environments). Using a dynamic-disCSP approach, we can ensure the satisfaction of the user requirements from the beginning to the end of the service provision. A dynamic-disCSP is a sequence of disCSPs $\langle \text{disCSP}_0, \text{disCSP}_1, ..., \text{disCSP}_n\rangle$, where disCSP_i formalize the current requirements (TR) satisfying their constraints (TC) of the Stask that must be fulfilled during the service delivering, that is the required adaptation of the solution. A disCSP is represented as a 5-tuple $\langle TR, D, TC, SYS\rangle$, where $TR = \{tr_i | i = 1, ..., m\}$ is a finite set of variables; tr_i is a variable corresponding to a service needed by a pervasive system component (i.e., a system) to satisfy a particular user requirement. $D = \{D_i | i = 1, ..., r\}$, such that for each tr_i, there is a service option domain D_i . TC is a finite set of constraints $\{TC_1, TC_2, ..., TC_t\}$ of the user requirement, and SYS is a set of autonomous systems over which the variables and constraints are distributed. In any system, the set of possible values for variable tr_i is its current domain D_i. Each disCSP_i results from a change in the previous disCSP_{i-1} and represents new situations in the dynamic environment. Fig. 2 shows a visual abstraction (each matrix is composed of participant systems sys_i and required services tr_j; each cell represents a service that each participant can provide and its level of relevance) of a sequence of four disCSPs $\langle \text{disCSP}_0, \text{disCSP}_1, \text{disCSP}_2, \text{disCSP}_3\rangle$. Several changes can be observed in the sequence, such as the following.

1) In disCSP_1, the sys_4 has left the PSoS (regarding disCSP_0).
2) In disCSP_2, the sys_7 has joined the PSoS (regarding disCSP_1).
3) In disCSP_3, the relevance level of the services provided by the sys_1, sys_2, and sys_3 has changed (regarding disCSP_2).

These kinds of changes may affect the components in the problem definition: variables (adding or removals), domains (changes in the intentional definition, value adding or removals in the case of extensional definition), constraints (adding or removals), constraint scopes (variable adding or removals), or constraint definitions (changes in the intentional definition, tuple adding or removals in the case of extensional definition). As a

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

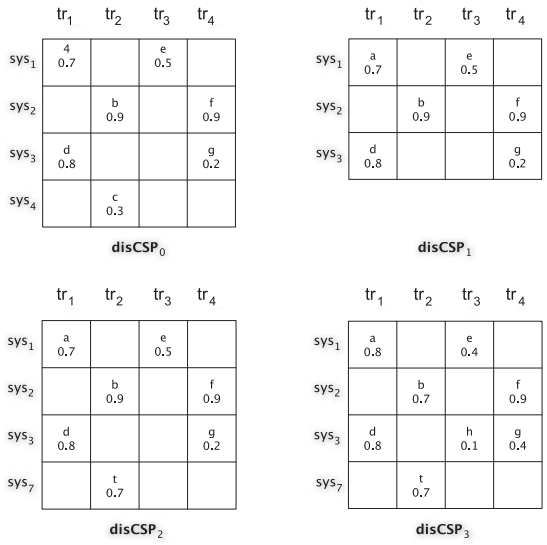

Fig. 2. Matricial representation of a Dynamic disCSP.

result of such changes, the set of solutions of each disCSP$_i$ can potentially decrease (it is a restriction over the disCSP$_i$) or increase (it is a relaxation over the disCSP$_i$).

IV. Pervasive Service Composition Protocol

The main differences between the Karmouch's work and ours are: the first assumes that nodes are always available and are reliable. Also, service selection is based on a function having parameters that determine the network quality. Finally, when the network quality changes, the service composition process starts from scratch. On the other hand, our approach does not assume the availability of the nodes nor reliability, which is an actual hypothesis. Also, when the environment or network condition changes, our approach attempts to recompose the service not from scratch, but by using previous solutions (partial composite service).

From our perspective, pervasive systems must be user centered, with the services remaining available despite user mobility through heterogeneous environments. Our objective is to examine possible pervasive system configurations in the users vicinity, identify a configuration that satisfies user requirements, and fulfill task constraints and user preferences. In order to deploy this type of service, we started from the SoS paradigm. We applied the PSoS notion to mobile systems based on ad hoc networks, where small systems can lead to large geographically distributed systems. Fig. 3 shows the conceptual model of a PSoS composed of autonomous systems (fixed or mobile), connected by an *ad hoc* network and providing one or more services (each of these services can have constraints) that may be requested by peer systems, that is, with the aim of fulfilling one or more tasks (each task can have requirements and preferences).

Below, we describe a protocol for the composition and adaptation of PSoSs and the services they provide. This protocol comprises three phases: PSoS composition, service composition, and service adaptation.

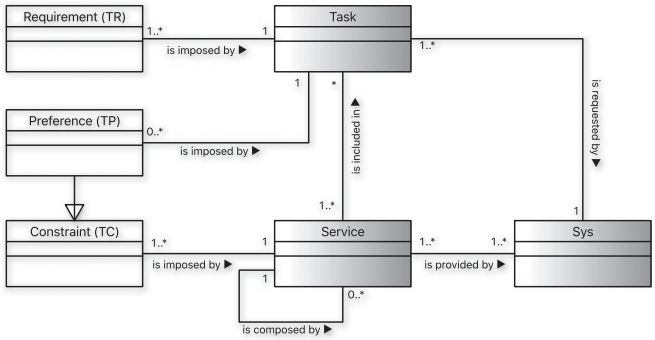

Fig. 3. Conceptual model of the PSoS context.

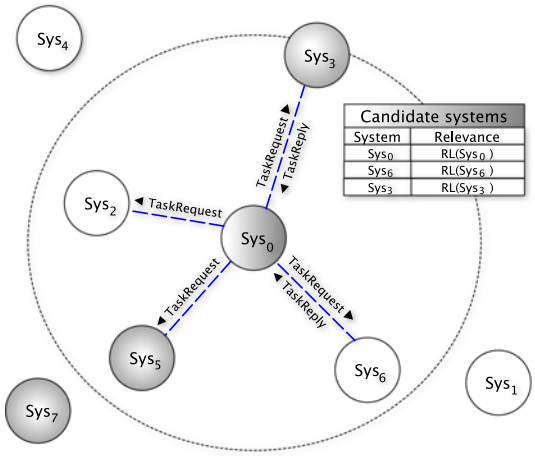

Fig. 4. Candidate systems choice during the PSoS composition.

A. Composition of the PSoS

The objective of the composition phase is to identify potential participant systems for the PSoS in the user's vicinity. When a user requirement becomes activated in the user system, a modified version of contract net protocol is used. That is, an initiator system sends a task request (TaskRequest) message, containing the STask description, to systems in the user's environment (see Fig. 4). The STask is used by receiving systems to decide whether they can contribute with at least one service to fulfill the STask. Systems that may contribute send a TaskReply message to the system initiator. With the TaskReply messages, the system initiator builds a candidate systems list, where each item of the list has the address and its relevance factor to fulfill the STask. Once the list is made, the initiator system sends it to all participant systems. Fig. 4 shows an example where a user requirement becomes activated at system sys$_0$.

The relevance level RL(sys$_i$) is determined by each autonomous system sys$_i$. The factors used to determine the relevance level can vary based on the application domain. These factors could come from the context of the user or system, for example, displacement speed, geographic location, user intent, system resources, and power of the communication signal. For the sake of simplicity, in this paper, we have only considered the displacement speed of the system (for mobile devices), the resources available for the system, and the current workload.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

In addition, the way to calculate the factor of relevance may vary from addition to another expression that best suits the application domain. We use the following functions to calculate the factor of relevance:

$$\mathrm{RL}(\mathrm{sys}_i) = K_1 \cdot R(\mathrm{sys}_i) + K_2 \cdot \frac{1}{\mathrm{WL}(\mathrm{sys}_i)} \tag{3}$$

$$\mathrm{RL}(\mathrm{sys}_i) = K_1 \cdot \frac{1}{S(\mathrm{sys}_i)} + K_2 \cdot R(\mathrm{sys}_i) + K_3 \cdot \frac{1}{\mathrm{WL}(\mathrm{sys}_i)}. \tag{4}$$

For fixed systems (for nonmobile systems), we use (3), and for mobile systems, we use (4). $K_j \in \mathbb{R}$ is a constant for tuning the relevance factors based on specific applications; its value is limited by $0 < K_j < 1$. $S(\mathrm{sys}_i)$ is the displacement speed of the system (in case of a mobile systems), and it is inversely proportional to its relevance factor (i.e., systems with high speed of displacement have low relevance for our PSoS). $R(\mathrm{sys}_i)$ represents a number of the system's resources (based on a specific application, each kind of resource may have different relevance), and $\mathrm{WL}(\mathrm{sys}_i)$ means the current work load of sys_i, which is inversely proportional to its relevance factor.

The relevance factor $\mathrm{RL}(\mathrm{sys}_i)$ has two purposes: to improve the participants selection, and to build the PSoS order that will be required at the composition and adaptation phases.

B. Service Composition

Once the initiator system has the list of participant systems, we apply the pervasive system composition protocol (PSCP) algorithm (see Algorithm 1). This is based on the asynchronous backtracking (ABT) algorithm used for resolving distributed constraint satisfaction problems [32]–[34]. Like similar to algorithms for solving disCSP, this requires a total relevance ordering of participant systems. For each $\mathrm{sys}_i \subseteq$ PSoS, system sys_j has a higher relevance level than sys_i if it appears before sys_i in the total ordering (on the participant list). On the other hand, sys_j has a lower relevance level than sys_i if it appears after sys_j in the total ordering (on the participant list). Therefore, the total order divides all neighboring participants systems of sys_i, $N(\mathrm{sys}_i)$, into higher relevance neighbors, $N^+(\mathrm{sys}_i)$, and lower priority neighbors, $N^-(\mathrm{sys}_i)$. In the real world, communication between systems is not necessarily first-in first-out; therefore, we used a timestamp that is incremented only if sys_i changes its assignments (thus, each assignment has a label).

Before starting with the explanation of the composition algorithm, we describe the main nomenclature (see Table I). This is in order to facilitate understanding of the service composition process.

In order to solve a disCSP_x, systems sys_i generate locally consistent assignments (i.e., any constraint (TC) is being violated) and exchange their new proposals with their neighbors $N^-(\mathrm{sys}_i)$ to achieve a global consistency. As in the ABT algorithm, each sys_i stores assignments received from its neighbors in its system view and a list of no-goods. sys_i stores in its system view the most up-to-date assignments of its higher priority neighbors. sys_i stores in its no-good list no-goods justifying values removal.

Algorithm 1: The PSCP protocol running in system Sys_i.

```
procedure pscp(participantSystems) do
  end ← false, a_i ← null, t_i ← 0;
  checkSystemView();
  while (!solution) do
    msg ← getMessage();
    switch (msg.type) do
      chk: processAssign(msg)   stp: end ← true;
      ngd: resolve(msg)         ada: dpsHeuristic();
  end while
end procedure
procedure ckeckSystemView() do
  if isConsistent(SystemView, a_i)!=true then
    a_i ← newAssignment();
    if a_i ≠ null then
      for each (s_k ∈ N⁻(s_i)) do
        sendMessage(chk, Assignment⟨ s_i, a_i, t ⟩, s_k);
      end for
    else
      backTrack();
    end if
  end if
end procedure
procedure processAssign(msg) do
  updateSystemView(msg.Assignments);
  checkSystemView();
end procedure
procedure resolve(msg) do
  if conflict(lrl(msg.noGood), SystemView) then
    myNoGoodList.add (msg.noGood);
    checkSystemView();
  else
    if hrl(msg.noGood).value == a_i ) then
      sendMessage(chk, myAssignment, msg.sender );
    end if
  end if
end procedure
procedure backTrack() do
  newNoGood ← solve(myNoGoodList);
  if !newNoGood then
    end ← true;
    sendMessage(stp, participantSystems);
  else
    sendMessage(ngd, newNoGood, s_j );
    updateSystemView(a_j ← null);
    checkSystemView();
  end if
end procedure
```

In the initial procedure pscp(), each sys_i assigns a value to its variable and informs its lower neighbors agents. Then, it loops to process the received messages. Procedure checkSystemView checks whether the current value (a_i) is consistent with SystemView. If a_i is inconsistent with assignments of higher priority neighbors, sys_i tries to select a consistent value. During this process, some values from $D(tr_i)$ may appear as inconsistent. Thus, no-goods justifying their removal are added to the no-good list of sys_i. When two no-goods are possible for the same value, sys_i selects the best no-good. If a consistent value exists, it is returned and then assigned to a_i. Next, sys_i informs all systems in $N^-(\mathrm{sys}_i)$ about its new assignments through chk messages. Otherwise, sys_i has to backtrack (using the backTrack() procedure). Whenever sys_i receives a chk message, it processes it by calling for procedure processAssign(msg). The SystemView of sys_i is updated (updateSystemView) only if the received

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE I
MAIN NOMENCLATURE OF THE ALGORITHM FOR SERVICE COMPOSITION

Symbol	Description
chk	sys_i use this message to informs about new assignments to systems in $N^-(sys_i)$
stp	Terminates the execution of the algorithm
ngd	sys_i use this message to informs about conflicts in his assignments to systems in $N^+(sys_i)$
ada	This message triggers the dpsHeuristic() process for adapting the current solution (a partial service composition)
lrl	set of antecedent assignments that generated a new inconsistent assignment (no-good)
hrl	set of consequent assignments (no-good) that were inconsistent due to a set of antecedent assignments

message contains a more assignment up to date than that already stored for the sender, and all no-goods become noncompatible when the SystemView of sys_i is removed.

Then, a consistent value for sys_i is sought after the change in the SystemView (checkSystemView). When every value of sys_i is forbidden by its noGoodList, procedure backTrack() is called for. In procedure backTrack(), sys_i resolves its no-goods, deriving a new no-good, newNoGood. The problem has no solution if newNoGood is empty. sys_i broadcasts the stp messages to all systems and terminates the execution. Otherwise, the new no-good is sent in an ngd message to the system, e.g., sys_j, owning the variable appearing in its lrl. Then, the assignment of sys_j is deleted from the SystemView (updateSystemView). Finally, a new consistent value is selected (checkSystemView).

Whenever sys_i receives an ngd message, procedure resolveConflict is called for. The no-good included in the ngd message is accepted only if its hrl is compatible with assignments on the SystemView of sys_i. Next, the no-good is stored, acting as justification for removing the value on its lrl. A new consistent value for sys_i is then sought (checkSystemView) if the current value was removed by the received no-good. If the no-good is not compatible with the SystemView, it is discarded as it is obsolete. However, if the value of a_i is correct in the received no-good, sys_i resends its assignment to the no-good sender via a chk message.

Afterward, sys_i sends its assignment through a chk message to the request sender if its value is different from that included in the received msg.

V. SERVICE ADAPTATION

Since the environment has changed (for example, a service has become unavailable or a participant has left the PSoS), the PSoS may need to adapt its topology and functionality to meet user needs. In order for the PSoS to be capable of service adaptation, we have proposed a heuristic named DPS that is based on the dynamic addition of new participants into the PSoS (procedure dpsHeuristic() in the Algorithm 1). For this, an identification phase of new potential participant systems for the PSoS in the user's vicinity is initiated. When an adaptation requirement becomes activated in the PSoS, an initiator system sends a task request (TaskRequest) message, containing the STask description with the last partial solution, to systems in the user's environment that are not participating in the current solution. The STask is used by receiving systems to decide whether they can contribute with at least one service to fulfill the STask. Systems that may contribute send a TaskReply message to the system initiator. With the TaskReply messages, the system initiator updates the candidate systems list, where each new item of the list has the address and most important factor in relation to candidates who contributed to the previous solution is assigned. Once the list is made, the initiator system sends it to all participant systems. Fig. 5 shows an example where an adaptation requirement becomes activated at system sys_0.

This heuristic allows PSCP adapts services without starting from scratch. That is, with the heuristic DPS, the algorithm PSCP can reuse the last service composition. Fig. 6 shows an abstraction of an initial service composition ($disCSP_i$) and the service composition state after changes in the environment ($disCSP'_i$) that can be used in the service adaptation process. In this example, sys_4 has left the PSoS.

In order to achieve adaptation, we start from the $disCSP_0$ that models the initial state of the PSoS. At the instant a service fault is detected, the initiator system starts the adaptation process. First, the PSoS attempts to reallocate the services required for the available participating systems. If it is not possible to find an assignment without violating constraints, then the initiator system sends a message TaskRequest to all systems founded in its neighborhood. This is in order to find new participant systems capable of contributing to service adaptation. Once the initiator system has located the potential systems, the heuristic DPS is applied. This heuristic is based on the idea that participating systems of the PSoS are unable to satisfy user requirements (i.e., it is not possible to find an assignment without violating constraints); therefore, it is necessary to a higher level of relevancy to the new systems participants. In practical terms, this idea implies that the initial partial solution (partial composition of a service) should be adjusted according to the contributions of new participant systems.

VI. EXPERIMENTAL EVALUATION

This section evaluates the performance of our pervasive system composition protocol with a dynamic partial solution (PSCP-DPS) by varying the system scale, mobility, service density, and adaptation order (number of elements of the composite service to be adapted). This section also evaluates the PSCP′s ability for system adaptation in order to deal with changes in the environment due to the system's mobility. Our experimental scenario consists of mobile systems capable of providing one or more services in a dynamic environment. These systems are interconnected using an *ad hoc* network. We considered

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

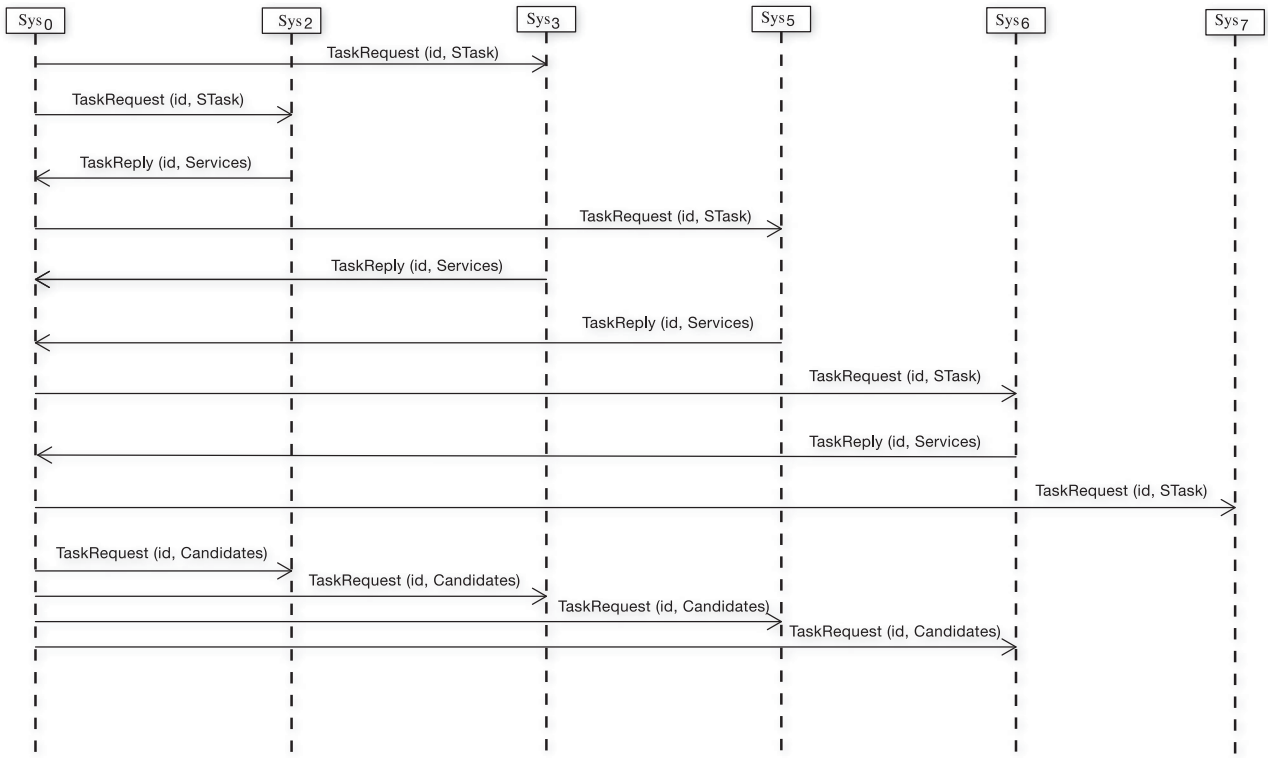

Fig. 5. New candidates for the service adaptation.

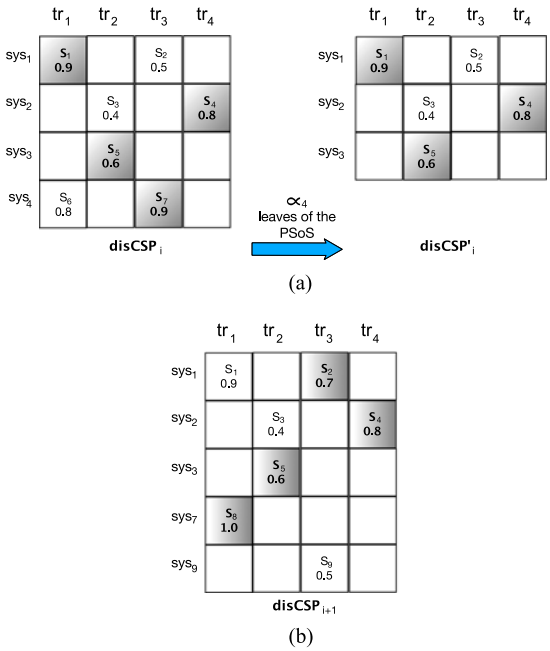

Fig. 6. Example of a possible service adaptation.

adaptations of several orders (in terms of the number of services that the PSoS needs to adapt in order to fulfill the task). We compare our results to those of Karmouch and Nayak [31] who present the virtual device constraint satisfaction protocol (VDCSP). VDCSP is a protocol for the composition and adaptation of virtual devices, which is based on disCSP and uses an adaptation from crash. The aim of the comparison is to identify the improvement of the performance in message utilization, composition time, and adaptation time using our PSCP-DPS (in dynamic environments). At the same time, we want to determine the effects of varying the service density, adaptation order, scale, and mobility on the same metrics.

A. Scenario Description

We built a pervasive system based on an *ad hoc* network and implemented both PSCP-DPS and VDCSP protocols using MASH [35], a platform that enables simulation and execution of embedded multiagent systems, including real-world software/hardware agents according to realistic models. Simulations were carried out over a set of autonomous systems in a previously delimited area. For the sake of simplicity, we have used the random-way-point mobility model [36]; however, we should consider different models of human mobility, vehicles mobility, etc. All broadcasts have a bounded hop count of 1 for the PSCP-DPS and VDCSP protocols. The simulation was carried out for a service composition length of 5. This length was used because it is the average length of service orders used in those related work closest to the problem addressed in our approach [31], [37]. The adaptation orders was carried out from 1 to 4; the variation of 1–4 is used to observe the behavior of the system considering from the best to worst case. We vary the percentage of nodes that have at least one service required to fulfill a requested task (i.e., the service density) from 10% to 100% to observe the behavior of the system considering from the best to worst case. The values of the parameters for calculating the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

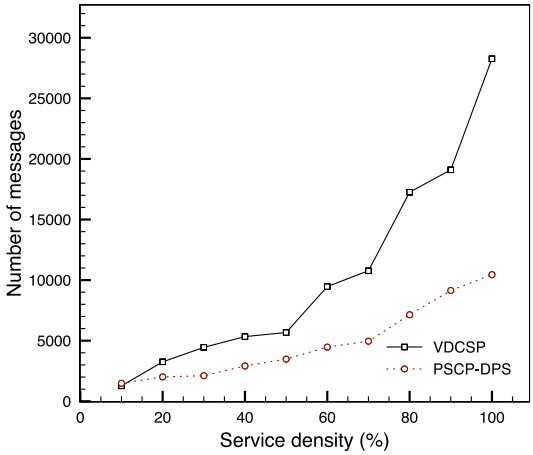

Fig. 7. Number of messages with respect to service density.

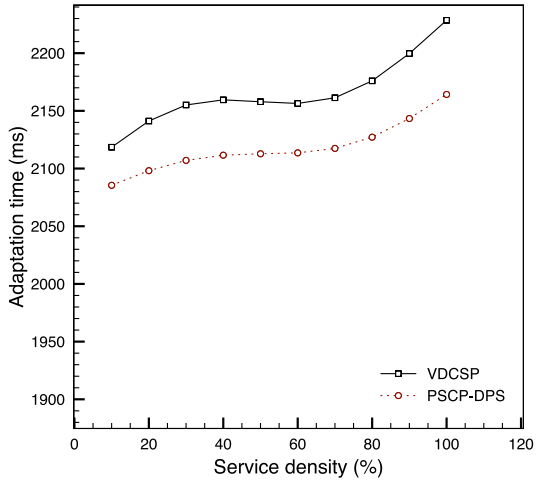

Fig. 8. Adaptation time with respect to service density.

relevance level of each node are randomly established. This is in order to avoid introducing a bias on the observed performance in simulations. For each simulation, we identify the amount of time taken and the number of messages consumed to achieve the adaptation (values are the average of 100 simulations). In order to focus on the performance of the protocols, we have utilized predetermined similarity values (for comparing services with each other) and do not consider the effect of network conditions on the performance of the two protocols.

The parameterization of our simulations uses service density, number of nodes (scale), composition order, adaptation order, and mobility, established in order to simulate potential real scenarios ranging from highly populated places such as malls and airports to places that are sparsely populated like homes.

B. *Effects of Service Density*

We use the concept of service density as the percentage of nodes that have least one of the services required to fulfill a requested task. In order to analyze effects of service density on the performance of each protocol, we used a scenario with 50 static systems (because the mobility of the system may introduce noise into the effect of the service density), a composition length at five services, and an adaptation order of one. The simulation was carried using service densities of 10%–100% in a spatial area of 50 m^2. For each simulation, we turned OFF a participating system that contributed to the composite service after the service composition; then, we identified the number of messages consumed and the amount of time used to achieve the service adaptation. Results indicated that per adaptation, the number of messages consumed by PSCP-DPS (see Fig. 7) is an average of 5669 messages (54%) lower than VDCSP, and the time used for adaptation decreases by 2.1173 ms (2.27%) compared to VDCSP (see Fig. 8). The reduction of consumed messages is because VDCSP needs to restart the composition process each time a participant system is no longer available to provide its service (i.e., it seeks the solution from scratch). The performance of both protocols, PSCP-DPS and VDCSP, is affected by the service density. Although a high service density provides robustness to the PSoS, it also has a negative effect: as the density of services increases, the number of participants for the composition and adaptation also increases. This implies a greater number of messages required to find a solution. However, Figs. 7 and 8 show that the PSCP-DPS (in time and consumed messages) performs better than the protocol VDCSP when the service density is high.

C. *Effects of Scaling*

In order to study the effect of scaling on the performance of protocols, we set the service density at 50% , a composition order of five services, and an adaptation order of one service. Additionally, we progressively increased the number of autonomous systems involved in a spatial area of 50 m^2. For each simulation, we turned OFF a participating system that contributed to the composite service after the service composition; then, we identified the number of messages consumed and the amount of time required to achieve the service adaptation. Results show that the performance of both protocols is affected by scalability issues. As the number of autonomous systems increases and the spatial environment area remains fixed, the time and the number of messages required to achieve a service composition and adaptation increase (see Figs. 9 and 10).

D. *Effects of Adaptation Order*

To examine the effects of adaptation order on the performance of the protocols, we set the service density at 50%, the number of autonomous systems at 50, and the composition order at five services. Additionally, we progressively increased the adaptation order (from 1 to 4) in a spatial area of 50 m^2. For each simulation, we turned OFF a participant system that contributed to the composite service after the service composition; then, we identified the number of messages consumed and the amount of time required to achieve the service adaptation. We repeated the last procedure, varying it by turning OFF two, three, and four participants.

Figs. 11 and 12 show the negative effect of the order of adaptation over time and the amount of messages required to adapt a service to changes in the environment. Results show that the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

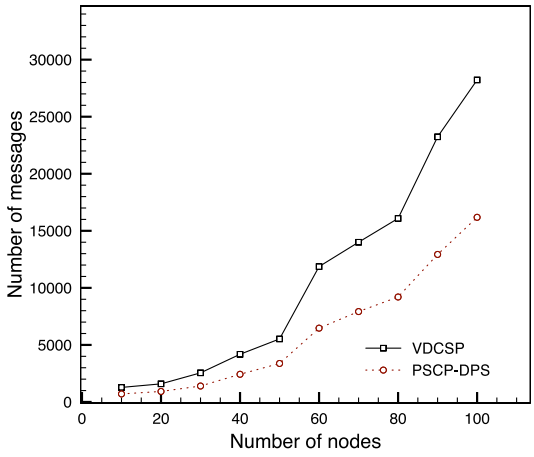

Fig. 9. Number of messages with respect to the number of nodes.

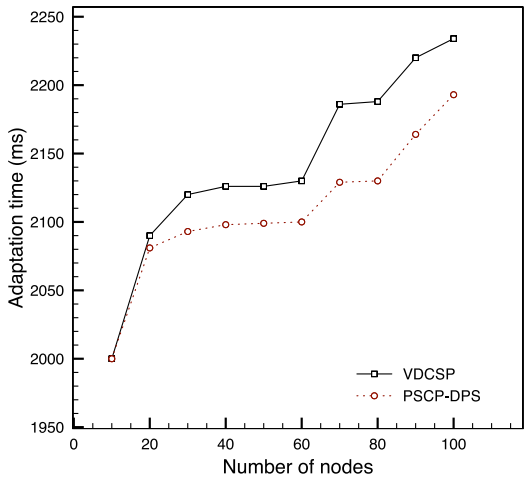

Fig. 10. Adaptation time with respect to the number of nodes.

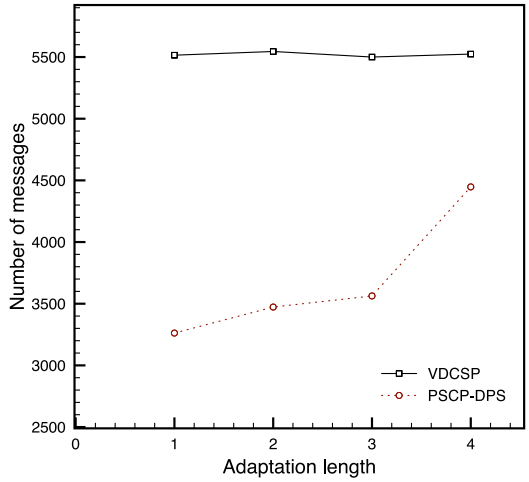

Fig. 11. Number of messages with respect to adaptation length.

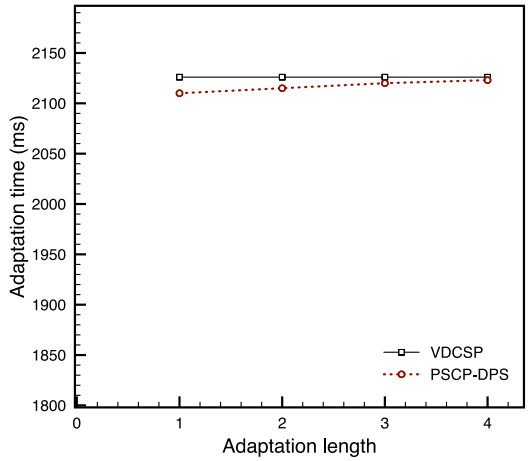

Fig. 12. Adaptation time with respect to adaptation length.

PSCP-DPS protocol performs better than the VDCSP protocol (in terms of messages and time). However, as the adaptation order goes increases, the performance of the PSCP-DPS protocol decreases. The performance of the protocol VDCSP remains almost the same because it does not include any strategy to adapt the system when a participant becomes unavailable. Therefore, in order to adapt the system, it restarts the service composition process from scratch.

E. Effects of Mobility

In order to analyze the mobility effects over the two protocols, we set the service density at 50%, the number of autonomous systems at 50, the composition order at five services, and we use a random waypoint mobility model for representing the user mobility behavior. In this case, the composition length was five, the network topology could change, and an autonomous system moved randomly in an area of 150 $\times$ 500 m. For the sake of simplicity and in order to have control over the scenario, we configured the environment with 50 fixed and one mobile systems. The position of the mobile system was randomly selected within a fixed area of 150 $\times$ 500 m and then moved linearly to the selected position with a consistent random speed. For each simulation, we identified the number of messages transmitted between the participating systems and the percentage of time during which the service remained available from the composition of the service until the end of the simulation. We repeated the experiments for different travel speeds (setting a random speed with values from 1 to 10 m/s).

To interpret the results of Fig. 13, it is important to consider that the graph represents the average of the results obtained from 100 simulations for each displacement velocity (from 1 to 10 m/s). The variation in the number of messages transmitted concerning the moving speed of the mobile system in the environment may seem chaotic. It is because the network topology changes in each simulation and this modifies the scenario of adaptation of the service (i.e., the orders of adaptation could vary from 1 to 5) after each composition process.

While the travel speed of the autonomous system is from 1 to 3 m/s (see Fig. 13), the VDCSP protocol transmits a smaller number of messages compared to the PSCP-DPS protocol. However, observing the results of Fig. 14 in the same range of travel speed, the PSCP-DPS protocol keeps the service available for a greater percentage of time. Thus, we can deduce that the higher

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

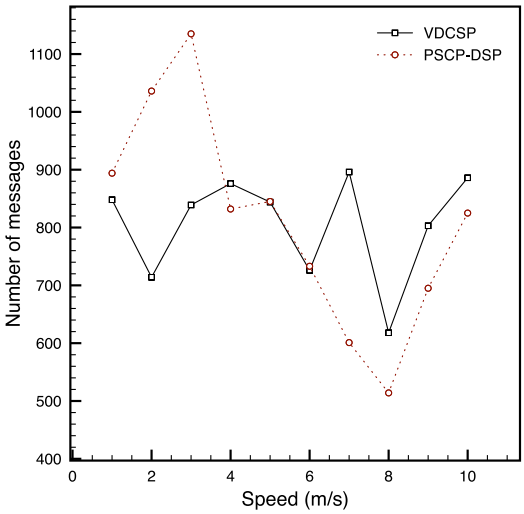

Fig. 13. Number of messages with respect to mobility.

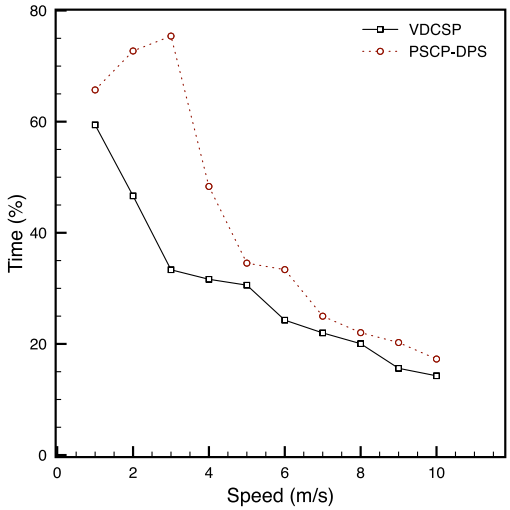

Fig. 14. Service availability with respect to mobility.

number of messages is due to the process of adaptation of the service performed by the PSCP-DPS protocol to keeps the service available.

Additionally, the general performance observed in Figs. 13 and 14 is due to the time during which the participating systems are in the transmission range is small, while the number of messages required to continuously adapt the system and maintain the service availability varies due to random displacement of the autonomous system in its environment.

Finally, the results show that mobility at high speed has an inversely proportional effect on the availability of the service.

Through simulation, we have shown that our protocol can maintain service availability despite the dynamism of the environment. However, it is important to consider the limitations and assumptions of the proposed approach.

Although, in this paper, we provide a solution for the service adaptation based on resources within a user's environment, no consideration was given to situations where there are available resources and services, but participant systems do not know how to interact to perform the service composition and adaptation. Further research is required accounting for scenarios where systems need to learn how to interact with each other in order to achieve some specific objective.

Multihop composition was not considered in this paper because one hop is enough to discover participant systems in the user's environment. However, even though the resources needed to provide a service may not be in the user's vicinity, the proposal can exploit them by using a directed multihop broadcast protocol.

One issue concerning the use of the random waypoint mobility model for representing human mobility behavior has to do with the sharp turn [38]. Sharp turn occurs whenever there is a direction change in the range of 90°–180°. This problem can be eliminated by allowing past direction to affect future direction. The Gauss–Markov mobility model [39] solves this problem to achieve more realistic movement of autonomous systems.

VII. Related Work

Research in service-oriented computing has been split into two main directions. The first direction focuses on languages to specify services [40]–[42]. The second direction faces the challenge of developing service composition architectures [43], [44]. From a service description, such architecture aims to perform the needed service discovery, composition, and execution. However, most of the solutions have been designed for dedicated infrastructures with centralized composition management [22], [45]–[47]. Most of these approaches involve preconfigured composition mechanisms residing on dedicated machines with high resources.

Graiet *et al.* [8] propose a genetic approach for service composition. The proposed genetic-based algorithm is based on the decomposition principle of global QoS constraints into locals one. Thus, the algorithm intends to optimize the performance of a local selection instead of a global one. Their approach is adaptive as it can be used to either define or reconfigure composite services according to the service-level agreements constraints and a set of acceptable termination states. However, the approach is focused on web services and needs of dedicated infrastructure with a centralized composition management.

Recently, the service composition has been introduced to exploit distributed resources in the user environment for purpose of supplying pervasive services, especially in MANETs. For example, Sadiq *et al.* [6] proposed an approach based on opportunistic networks of mobile devices users to take advantage of all components of distributed services to provide users with application-level services. However, the proposed algorithm requires a global view of connectivity between nodes. Wang *et al.* [48] formulate the service composition as a multiobjective optimization problem to minimize the service cost and maximize the QoS and their information. Such solutions fail to meet the requirements of unstable nodes in dynamic environments. This has led composition toward distributed services adaptation.

Service adaptation analyzes different methods suitable for service recomposition without the need for a centralized coordination or repositories. The existing solutions can be split into two

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

groups. The first is adaptation focused on service quality. The second group deals with the functional adaptation of services. An adaptation focused on service quality allows the system to find the best solution through the adaptation of the service according to its environment. Cardellini addressed the problem of service adaptation, determining the most suitable configuration by solving an optimization problem, MOSES [11]. However, MOSES requires nodes to always be available and a reliable network. In dynamic environments, there is the possibility that one or more nodes may leave the network or become unavailable.

Functional adaptation of services searches for functionality providers for service recomposition. Wang *et al.* [49] presented a distributed algorithm. This method was based on directed acyclic graphs as a model for the service flow. The resulting service flow graph is resource efficient and leads to a good performance in terms of bandwidth and latency. However, the solution does not consider the mobility of the service provider. Therefore, the solution is impractical in dynamic environments with user mobility.

Most of the authors propose service adaptation from scratch. Hassine *et al.* [50] proposed a constraint-based approach. The composition method determines the most appropriate service from a functionally equivalent set. However, the proposed protocol is based on an incremental user intervention to find services.

Karmouch and Nayak [31] introduced a distributed constraint satisfaction problem-based approach. The proposal focuses on a constraint-satisfaction-based protocol (VDCSP). VDCSP is a protocol for the composition and adaptation of virtual devices. This protocol tends to solve a problem given available abstract services, service domains, constraints, and an objective function. However, the service adaptation is from crash, and the solution does not consider the mobility of the service provider. Therefore, the solution is impractical in dynamic environments with user mobility.

Our proposal is not intended to find an optimal solution that maximizes the QoS, because the optimal solution can often change as the environment changes or the user moves between environments.

VIII. Conclusion

In this paper, we presented a dynamic disCSP model for pervasive service composition in dynamic environments and a new heuristic useful for adapting partial solutions using an ABT algorithm for solving the dynamic disCSP. The model provides a technique to adapt services without restarting the service composition process each time a system participant is unavailable or leaves the PSoS. Through simulation, we have shown that our protocol can maintain service availability despite the dynamism of the environment.

Our future research trend includes the design of adaptation techniques based on learning with multihop broadcasting and a framework to choice the best composition and adaptation techniques according to the problem and environment characterization. Multihop composition and adaptation were not considered in this paper because one hop is sufficient for discovering autonomous systems in the users' environment. Additionally, the knowledge acquired may be applied as we experiment with autonomous systems in real devices in the real world.

Acknowledgment

The authors would like to thank the anonymous reviewers for their comments and suggestions.

References

[1] F. Zhu, M. W. Mutka, and L. M. Ni, "Service discovery in pervasive computing environments," *IEEE Pervasive Comput.*, vol. 4, no. 4, pp. 81–90, Oct. 2005.

[2] T. Phan and W.-S. Li, "Automated configuration of system infrastructure for SOA-based enterprise computing," in *Proc. IEEE Int. Conf. E-Bus. Eng.*, Oct. 2008, pp. 205–212.

[3] M. Eichhorn, M. Pfannenstein, D. Muhra, and E. Steinbach, "A SOA-based middleware concept for in-vehicle service discovery and device integration," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2010, pp. 663–669.

[4] G. Pan *et al.*, "Pervasive service bus: Smart SOA infrastructure for ambient intelligence," *IEEE Intell. Syst.*, vol. 29, no. 4, pp. 52–60, Jul. 2014.

[5] L. Wang, L. Rui, X. Qiu, W. Li, and K. Jiang, "A self-adaptive recovery strategy for service composition in ubiquitous stub environments," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2013, pp. 892–897.

[6] U. Sadiq, M. Kumar, A. Passarella, and M. Conti, "Service composition in opportunistic networks: A load and mobility aware solution," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2308–2322, Aug. 2015.

[7] P. Choudhury, P. Dutta, S. Nandi, and N. Debnath, "Mobility aware distributed service composition framework in SOA based MANET application," in *Proc. 10th IEEE Int. Conf. Ind. Informat.*, Jul. 2012, pp. 1016–1021.

[8] M. Graiet, I. Abbassi, M. Kmimech, and W. Gaaloul, "A genetic-based adaptive approach for reliable and efficient service composition," *IEEE Syst. J.*, 2016, to be published.

[9] S. Cook, "The complexity of theorem proving procedures," in *Proc. 3rd ACM Symp. Theory Comput.*, 1971, pp. 151–158.

[10] J. Andersson, R. de Lemos, S. Malek, and D. Weyns, "Modeling dimensions of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems (ser. Lecture Notes in Computer Science)*, vol. 5525, B. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Germany: Springer, 2009, pp. 27–47.

[11] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. Lo Presti, and R. Mirandola, "MOSES: A framework for QoS driven runtime adaptation of service-oriented systems," *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1138–1159, Sep. 2012.

[12] C. Baladron, J. Aguiar, B. Carro, L. Calavia, A. Cadenas, and A. Sanchez-Esguevillas, "Framework for intelligent service adaptation to user's context in next generation networks," *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 18–25, Mar. 2012.

[13] V. W.-M. Kwan, F. C.-M. Lau, and C.-L. Wang, "Functionality adaptation: A context-aware service code adaptation for pervasive computing environments," in *Proc. IEEE/WIC Int. Conf. Web Intell.*, Oct. 2003, pp. 358–364.

[14] V. Schwartze, "An interactive user interface adaptation process," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, Mar. 2012, pp. 546–547.

[15] C. Funk, A. Schultheis, C. Linnhoff-Popien, J. Mitic, and C. Kuhmunch, "Adaptation of composite services in pervasive computing environments," in *Proc. IEEE Int. Conf. Pervasive Serv.*, Jul. 2007, pp. 242–249.

[16] P.-A. Avouac, P. Lalanda, and L. Nigay, "Adaptable multimodal interfaces in pervasive environments," in *Proc. IEEE Consum. Commun. Netw. Conf.*, Jan. 2012, pp. 544–548.

[17] S. Hagen and A. Kemper, "Facing the unpredictable: Automated adaption of it change plans for unpredictable management domains," in *Proc. Int. Conf. Netw. Serv. Manag.*, Oct. 2010, pp. 33–40.

[18] J. Cervino, E. Kalyvianaki, J. Salvachua, and P. Pietzuch, "Adaptive provisioning of stream processing systems in the cloud," in *Proc. IEEE 28th Int. Conf. Data Eng. Workshops*, Apr. 2012, pp. 295–301.

[19] Y. Maurel, S. Chollet, V. Lestideau, J. Bardin, P. Lalanda, and A. Bottaro, "fANFARE: Autonomic framework for service-based pervasive environment," in *Proc. IEEE 9th Int. Conf. Serv. Comput.*, Jun. 2012, pp. 65–72.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[20] S. Komorita *et al.*, "Loosely coupled service composition for deployment of next generation service overlay networks," *IEEE Commun. Mag.*, vol. 50, no. 1, pp. 62–72, Jan. 2012.
[21] K. Zielinnski, T. Szydlo, R. Szymacha, J. Kosinski, J. Kosinska, and M. Jarzab, "Adaptive SOA solution stack," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 149–163, Apr. 2012.
[22] W. T. Tsai, P. Zhong, X. Bai, and J. Elston, "Dependence-guided service composition for user-centric SOA," *IEEE Syst. J.*, vol. 8, no. 3, pp. 889–899, Sep. 2014.
[23] F. Bellotti, R. Berta, A. De Gloria, and M. Margarone, "User testing a hypermedia tour guide," *IEEE Pervasive Comput.*, vol. 1, no. 2, pp. 33–41, Apr. 2002.
[24] O. Evangelatos, K. Samarasinghe, and J. Rolim, "Syndesi: A framework for creating personalized smart environments using wireless sensor networks," in *Proc. IEEE Int. Conf. Distrib. Comput. Sensor Syst.*, May 2013, pp. 325–330.
[25] G. Tolle *et al.*, "A macroscope in the redwoods," in *Proc. 3rd Int. Conf. Embedded Netw. Sens. Syst.*, 2005, pp. 51–63.
[26] Y. Liu *et al.*, "Formal analysis of pervasive computing systems," in *Proc. 17th Int. Conf. Eng. Complex Comput. Syst.*, Jul. 2012, pp. 169–178.
[27] M. Jamshidi, "System of systems engineering—New challenges for the 21st century," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 23, no. 5, pp. 4–19, May 2008.
[28] C. Roncoli, C. Bersani, and R. Sacile, "A risk-based system of systems approach to control the transport flows of dangerous goods by road," *IEEE Syst. J.*, vol. 7, no. 4, pp. 561–570, Dec. 2013.
[29] J. Wang, X. Zhao, B. Xu, W. Wang, and Z. Niu, "Immune multi-agent model using vaccine for cooperative air-defense system of systems for surface warship formation based on danger theory," *J. Syst. Eng. Electron.*, vol. 24, no. 6, pp. 946–953, Dec. 2013.
[30] B. Ge, K. Hipel, K. Yang, and Y. Chen, "A novel executable modeling approach for system-of-systems architecture," *IEEE Syst. J.*, vol. 8, no. 1, pp. 4–13, Mar. 2014.
[31] E. Karmouch and A. Nayak, "A distributed constraint satisfaction problem approach to virtual device composition," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 11, pp. 1997–2009, Nov. 2012.
[32] M. Yokoo, T. Ishida, E. H. Durfee, and K. Kuwabara, "Distributed constraint satisfaction for formalizing distributed problem solving," in *Proc. 12th Int. Conf. Distrib. Comput. Syst.*, Jun. 1992, pp. 614–621.
[33] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: Formalization and algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 10, no. 5, pp. 673–685, Sep. 1998.
[34] K. Ghédira, *Constraint Satisfaction Problems*. Hoboken, NJ, USA: Wiley, 2013, ch. 8, pp. 200–203.
[35] J.-P. Jamont, M. Occello, and E. Mendes, "Decentralized intelligent real world embedded systems: A tool to tune design and deployment," in *Advances on Practical Applications of Agents and Multi-Agent Systems* (ser. Lecture Notes Comput. Sci.), vol. 7879, Y. Demazeau, T. Ishida, J. Corchado, and J. Bajo, Eds. Berlin, Germany: Springer, 2013, pp. 133–144.
[36] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless *ad hoc* networks," *IEEE Trans. Mob. Comput.*, vol. 2, no. 3, pp. 257–269, Jul. 2003.
[37] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "Toward distributed service discovery in pervasive computing environments," *IEEE Trans. Mobile Comput.*, vol. 5, no. 2, pp. 97–112, Feb. 2006.
[38] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriyawong, "A comparative study of random waypoint and Gauss-Markov mobility models in the performance evaluation of MANET," in *Proc. Int. Symp. Commun. Inf. Technol.*, Oct. 2006, pp. 894–899.
[39] B. Liang and Z. Haas, "Predictive distance-based mobility management for PCS networks," in *Proc. IEEE 18th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Mar. 1999, vol. 3, pp. 1377–1384.
[40] W3C, "Web services description language (WSDL) 1.1, w3c note 15 march 2001," Jun. 2015. [Online]. Available: http://www.w3.org/TR/wsdl
[41] R. Khalaf, A. Keller and F. Leymann, "Business processes for Web services: Principles and applications," *IBM Syst. J.*, vol. 45, no. 2, pp. 425–446, 2006.
[42] L. Simon, A. Mallya, A. Bansal, G. Gupta, and T. Hite, "A universal service description language," in *Proc. IEEE Int. Conf. Web Serv.*, Jul. 2005, p. 824.
[43] I. Paik, W. Chen, and M. Huhns, "A scalable architecture for automatic service composition," *IEEE Trans. Serv. Comput.*, vol. 7, no. 1, pp. 82–95, Jan. 2014.
[44] D. Chakraborty, F. Perich, A. Joshi, T. Finin, and Y. Yesha, "A reactive service composition architecture for pervasive computing environments," in *Mobile and Wireless Communications: IFIP TC6 / WG6.8 Working Conference on Personal Wireless Communications (PWC'2002) Oct. 23–25, 2002*, Singapore, C. G. Omidyar, Ed. Boston, MA, USA: Springer, 2003, pp. 53–60.
[45] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. C. Shan, "Adaptive and dynamic service composition in eflow," in *Proc. 12th Int. Conf. Adv. Inf. Syst. Eng.*, 2000, pp. 13–31.
[46] F. Cicirelli, A. Furfaro, and L. Nigro, "Integration and interoperability between Jini services and web services," in *Proc. IEEE Int. Conf. Serv. Comput.*, Jul. 2007, pp. 278–285.
[47] K. Rajaram and C. Babu, "Template based SOA framework for dynamic and adaptive composition of web services," in *Proc. Int. Conf. Netw. Inf. Technol.*, Jun. 2010, pp. 49–53.
[48] Y. Wang, I. R. Chen, J. H. Cho, A. Swami, and K. Chan, "Trust-based service composition and binding with multiple objective optimization in service-oriented mobile *ad hoc* networks," *IEEE Trans. Serv. Comput.*, 2015, to be published.
[49] J. Wang, J. Wang, N. Gu, and B. Yang, "Fault tolerant service composition in service overlay networks," in *Proc. IEEE Glob. Telecommun. Conf.*, 2008, pp. 1–5.
[50] A. Hassine, M. Matsubara, and T. Ishida, "A constraint-based approach to horizontal web service composition," in *Proc. Int. Semantic Web Conf.*, 2006, pp. 130–143.

Francisco Cervantes (M'14) received the M.Sc. degree in computer science from CENIDET, Cuernavaca, Mexico, in 2006, the Ph.D. degree in electrical engineering from CINVESTAV, Guadalajara, Mexico, in 2015, and the Ph.D. degree in computer sciences from the Grenoble Institute of Technology, Grenoble, France, in 2016.

He is currently an Associate Professor with ITESO University, Tlaquepaque, Mexico. His research interests include multi-agent systems, self-adaptive systems, and constraint satisfactions problems.

Félix Ramos (A'10–M'13) received the M.Sc. degree from CINVESTAV, Mexico City, Mexico, in 1986, the DEA degree in distributed systems from CNAM, Paris, France, in 1994, and the Ph.D. degree from the University of Technology of Compiegne, Compiegne, France, in 1997.

He is currently a Professor with in the Department of Computer Sciences, CINVESTAV Guadalajara, Zapopan, Mexico. His research interests include distributed systems, multi-agent systems, and virtual reality applications.

Luis F. Gutiérrez (M'15) received the B.E. degree in computer systems from the Autonomous University of Aguascalientes, Aguascalientes, Mexico, in 2007, and the M.Sc. degree in computer science and Ph.D. degree from CINVESTAV Guadalajara, Guadalajara, Mexico, in 2009 and 2013, respectively.

He is currently a Professor with the Department of Electronics, Systems, and Informatics, ITESO University, Tlaquepaque, Mexico. His research interests include multi-agent systems, unstructured data analytics, and virtual reality applications.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Michel Occello (M'17) received the Ph.D. degree in computer science for a work about the blackboard metaphor from the University of Nice-Sophia-Antipolis, Nice, France, in 1993, and the Habilitation a Diriger les Recherches degree on the theme of methodology and architectures for multi-agent design from the University Joseph Fourier, Grenoble, France, in 2003.

He is a Professor with the University of Grenoble, Valence, France. Following a position as a Research Assistant with the I3S CNRS Laboratory, he joined the CNRS IMAG/Leibniz Laboratory, Grenoble, France, as an Associate Professor. In 2002, he contributed to launch the activity around multi-agent systems with the LCIS Laboratory, University of Grenoble, Valence, France. He is currently the Director of this laboratory. His main research interest include multi-agent systems methodology (life cycles study, analysis approaches, componential aspects, CASE Tool for multi-agent design), adaptive self-organizations and recursive architectures, and application of multi-agent systems to collective intelligent embedded systems and new software technologies.

Jean-Paul Jamont (S'03–M'05) received the Ph.D. degree in computer sciences from the Grenoble Institute of Technology, Grenoble, France, in 2005.

He has been an Engineer with the French company EREIMA, where he was involved with industrial embedded systems. He joined the University of Grenoble, as an Associate Professor, where he teaches computer sciences and network systems. He leads the Networks and Telecommunications Department, University Institute of Technology, Valence (University of Grenoble Alpes). He has carried out research on embedded multi-agent systems (methodology, models, and architectures) and self-organized systems.