Optimizing Service Selection and Allocation in Situational Computing Applications

Chiara Sandionigi, Danilo Ardagna, Gianpaolo Cugola and Carlo Ghezzi

***Abstract*—This paper describes a novel model for the service selection problem of workflow-based applications in the context of self-managing situated computing. In such systems, the execution environment includes different types of devices, from remote servers to personal notebooks, smartphones, and wireless sensors, which build an infrastructure that can dynamically change both its physical and logical architecture at run-time. We assume that worflows are defined abstractly; i.e., they invoke abstract services whose concrete counterparts can be selected dynamically. We also assume that concrete service implementations may possibly migrate on the nodes of the infrastructure. The selection problem we address is framed as an optimization problem of the quality of service, which evaluates at run-time the optimal binding to concrete services as well as the trade-off between the remote execution of software fragments and their dynamic deployment on local nodes of the computational environment. The final deployment takes into account quality of service constraints, the capabilities of the physical devices involved, including their performance and energy consumption, and the characteristics of the networking links connecting them.**

***Index Terms*—M.1.0.e Optimization of Services Systems, M.6.1.b Optimization of Services Composition, M.4.4.h Quality of Services.**

I. Introduction

Recent advances in information and communication technology led to the development of pervasive applications that live in a large scale, continuously evolving environment [1]. Billions of mobile phones, intelligent sensors, and embedded mobile systems surround all of us throughout our everyday lives and interact with each other to help in achieving our goals. On the other hand, this requires innovative software applications capable of operating in a *situation* (or *context*)-*aware* manner. Based on the current situation (e.g., the current spatial location) they may choose dynamically the components to interact with, among those discovered in the surrounding environment, in order to achieve their goals.

Many emerging applications fall under this case and the term *ambient intelligence* is often used to indicate their target. As an example, later in the paper we consider health care as our main case study, and in particular assisted living of elderly or impaired people, who need an external help to achieve certain levels of autonomy through automation of assisted interaction with the world.

The *Software as a Service* (SaaS) paradigm [2] is especially relevant in the above context for three main reasons. First, it fosters the idea that each software fragment provides a well-defined functionality of general use to be composed with other software fragments to achieve specific goals [3]. This is obtained through a precisely defined *interface specification* that is separate from its *implementation*. The interface specification defines the functionality and, more generally, the Quality of Service (QoS) promised by the service, whereas the implementation is hidden to the clients [4], [5]. Second, the architecture is based on a *discovery* phase that precedes the *binding* between service invocation and service implementation. In the discovery phase, the description of the required service is used to query a *registry*, where service specifications are published. This step decouples service providers and service requesters, giving rise to the third characteristics of SaaS: *dynamic binding*. Indeed, thanks to the discovery phase the client may specify the requested service *abstractly*, through its *required interface*. The concrete service that is bound to an invocation is selected dynamically, at run-time, depending on its *provided interface*, on the required and provided QoS, and on other considerations (more details later). Dynamic binding allows software architectures to evolve at run-time, since the components that are part of the application and their relationships may change dynamically [4]–[6]. This is particularly important for situation-aware software operating in dynamic environments like the ones we target, as it supports continuous self-adaptation of the application's architecture in response to situational changes [7].

Given these premises, we can look at the binding problem from the situation-aware, pervasive applications viewpoint as an *optimization problem*, which tries to select the best services to invoke in order to maximize the QoS of the overall composition [3]–[5]. In this paper, we propose a novel approach for service selection in a pervasive environment by considering two dominant factors: dynamism of the external environment [1] and the ability of moving code around the network [8]. As we mentioned, the former dimension is typical of modern pervasive scenarios, which involve mobile users, mobile devices, wireless networks, and complex interactions schemas, which change frequently at run-time. It asks for solutions that easily accommodate the need for re-optimization as significant changes occur [9]. The latter dimension allows the choice of two alternative ways of executing a service. The former is *remote execution* in the service provider's run-time environment. This is normally the only possible solution for *pure* service-oriented applications, like those based on Web services. Pervasive applications, however, may support another way of executing the service called *local execution*, which is based on dynamically deploying and executing the service locally on the nodes of the pervasive environment. As an example, the processing of data collected through sensors

The authors are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy. E-mail: lastname@elet.polimi.it.

could be done on a local node, to avoid the need of transmitting the collected data to a remote processing node [10]. Our approach can take local execution into account whenever it is a viable option, both in the case where services are stateless and in the case in which they are stateful, as discussed later in the paper.

The remainder of the paper is organized as follows. Section II presents our reference framework, which is currently under development, and describes a home health care application used as a case study to show the benefits of our approach. Section III introduces the application and quality models we adopt. Optimization and run-time re-optimization are presented in Sections IV and V, respectively. Experimental results in Section VI demonstrate the effectiveness of our approach, while a discussion of other literature proposals is reported in Section VII. Finally, Section VIII draws some conclusions and illustrates future work.

II. Reference framework

The work described in this paper is part of a larger research effort (SMScom [11]), which is devoted to developing systematic approaches and tools to assist design, implementation, operation, and evolution of the self-managing, situation-aware applications. A typical SMScom scenario involves one or more users immersed in a pervasive environment, composed of different devices, connected through wired and wireless links, and capable of collaborating to promptly and efficiently satisfy the users' needs. The logical architecture which results from this vision is shown in Figure 1.

Applications are expressed as workflows built from a set of *abstract* services. This is exemplified by the activity diagram in Figure 1, which represents an application orchestrating four abstract services: as_1, as_2, as_3, as_4. In this paper, we assume that orchestrations are implemented in a BPEL-like language [12]. At run-time, the main end-user device (typically a hand-held device, like the PDA in the figure) acts as the *Application Manager*, running a light-weight BPEL engine to orchestrate the application's execution. As the application is launched, and possibly also as dynamic changes are detected in the environment in which the application is immersed, the Application Manager invokes an *Optimizer module* (running in the same user's device or as a remote service) to dynamically select the binding to the *concrete* services to invoke. The set of available concrete services is retrieved from the *SMScom Service Registry* [11], an extension of a UDDI registry [3], [13], [14], where concrete services are registered and annotated by a semantic description. The Optimizer module may also decide where services have to run. As mentioned, we consider that some concrete services can be executed either by the Service Provider exporting them (*remote execution*), or they can be dynamically deployed and executed on the devices available in the pervasive environment close to the end-user (*local execution*), which provide (possibly limited) computing capabilities. Once the assignment of abstract to concrete services has been established, the user's Application Manager invokes concrete service operations, through wrappers that are dynamically generated according to the WSDL specification of the chosen services. These wrappers leverage the SMScom middleware [15] that runs on the pervasive system glueing all the devices together to support seamless service location and invocation.

Our framework can deal with both *stateless* and *stateful* services. Decentralized execution can be an option only in the case of stateless services or stateful services whose state must be kept to support execution of a single instance of the workflow. In the latter case, once a binding is established and a decision is made concerning remote versus local execution, no further change is possible until execution of the instance terminates. Stateful services whose state must be kept to support execution of a multiple instances are instead not to be transferrable for local execution. A possible decision to decentralize their execution would in fact involve all possible workflows that might access them, whereas here we only consider decisions that optimize single workflow instances.

The Optimizer module represents a core component of our approach. It has to dynamically discover the optimal mapping between each abstract service and a (local or remote) concrete service that implements it. The main goal of the Optimizer is to maximize the QoS perceived by the end-user in running the application, while satisfying, at the same time, the constraints specified as part of the application's requirements. In particular, the SMScom framework supports the specification of both *local* constraints (which predicate on properties of a single abstract service) and *global* constraints (which introduce non-functional requirements on a set of abstract services or on the whole workflow).

The data about services needed by the Optimizer are stored in the SMScom registry. For each service, the registry tells us whether the service can be dynamically deployed on local nodes or not. It also stores data about QoS parameters (e.g., response time, energy consumption, etc.) monitored at run-time. The data retrieved from the registry provide the parameters for the initial service selection problem. QoS parameters, however, are subject to high variability. Indeed, the performance of services available in the Internet can vary up to an order of magnitude within the same business day [16], while the user behavior and the characteristics of the pervasive environment (i.e., the application's *context* [17]) also change at run-time. For example, if a user connected to an ad-hoc WIFI network changes his physical position the network bandwidth might be reduced and one or more physical devices could even become unreachable. In addition, other devices may suddenly stop operating if their batteries deplete. Run-time situational variability is the main obstacle to the continuous fulfilment of the global constraints the user wants to satisfy. To address this problem, execution monitoring, dynamic service selection, and workflow optimization have to be performed continuously, interleaved with application's execution. At the same time, re-optimization has to be triggered by the Application Manager if the invocation of a service fails or if the performance of services degrades, violating global constraints.

The effectiveness of the framework above will be assessed through a proof of concept application borrowed from the healthcare domain, whose goal is to provide a nursing home service involving non-specialised personnel only. In particular,

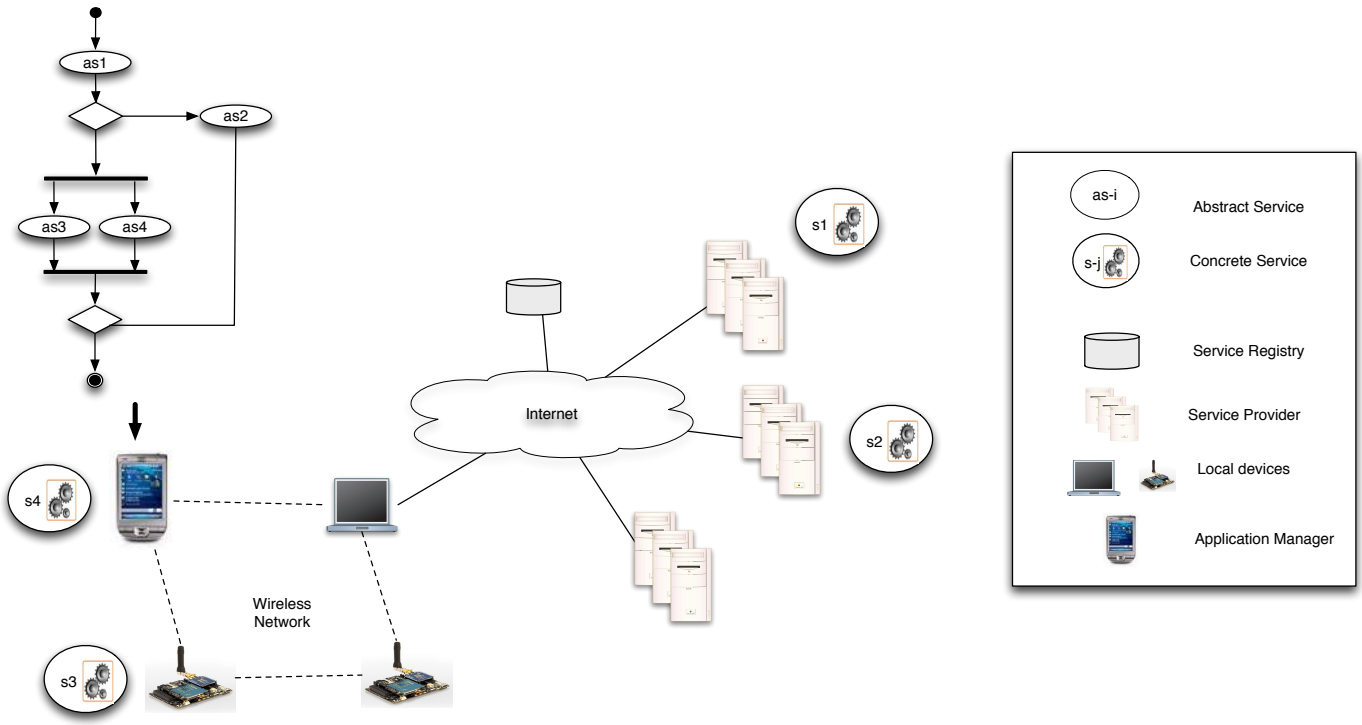

Fig. 1. SMScom reference framework

we consider the workflow illustrated by the activity diagram in Figure 2, where a person with heart and diabetes problems is living at her home, where two networks are available. The former is a *body sensor network*, i.e., a network built by wireless wearable sensors, which collects data on the patient's biological and physiological parameters. More specifically, the patient wears sensors to monitor heart rate, blood pressure and, occasionally, when the medical examination at home takes place, sensors to evaluate glucose and oxygen blood concentration. The latter is a *local area network*, which connects heart rate and blood pressure sensors to a local PC. The local area network provides also WIFI connectivity to the nurse hand-held device, which plays the role of the Application Manager, storing and hosting the execution of the application reported in Figure 2.

When the medical examination at home starts, heart rate and blood pressure are gathered from the corresponding sensors. These services are stateless, and can only be executed remotely on the corresponding devices that offer them. Then, the application evaluates if an electrocardiogram (ECG) is required. In such a case, multi-electrode ECG probes are applied to the patient and an ECG is performed by a portable sensor provided by the medical personnel. The ECG processing service is an expert system which is able to diagnose ECG data while the non-specialized medical personnel is not. The ECG service is stateful, and can be both executed remotely or transferred for local execution. Then, the medical personnel performs oximetry and glucose analyses. Oximetry and glucose analysis services are stateless, and they can be: (i) executed on the portable sensor, (ii) executed on a remote service provider, (iii) dynamically deployed and executed on the nurse's hand-held device, or (iv) dynamically deployed and executed on the local PC. Finally, the application decides if further analyses are needed at a remote center. In this case, a remote (steteful)

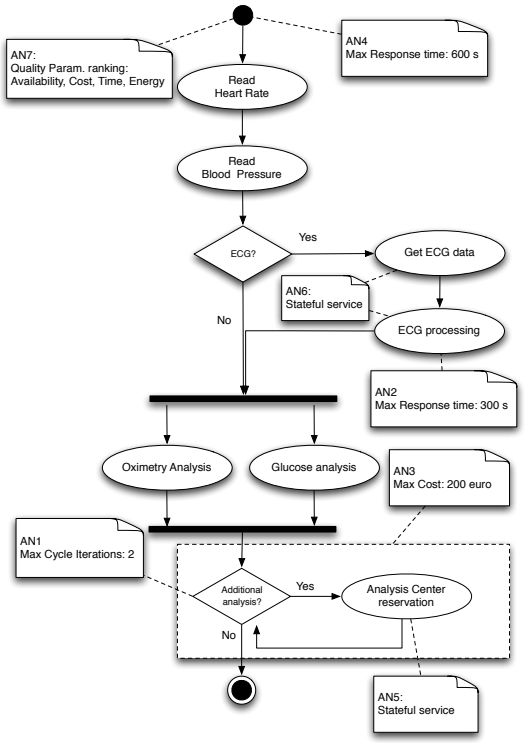

Fig. 2. Home healthcare application case study (Medical examination at home)

service is selected dynamically according to the patient's location (gathered by the Application Manager). The service can only be executed on a remote Service Provider and is accessible by possibly many instances of medical workflows.

III. Application and quality models

In this section we first provide a detailed description of the application model adopted in our framework (Section III-A) and then in Section III-B we discuss the quality model.

A. Application Model

As mentioned, we focus on applications coded as workflows, whose representation may be given in terms of activity diagrams and whose implementation may be written in BPEL [12]. We assume the workflow to be abstract; i.e., all service invocations directed to services that are not known yet and must be instantiated prior to execution. For simplicity, we also assume the activity diagram to have a single entry and a single exit node. To model the peculiar aspects of our framework, activity diagrams are enriched with ad-hoc annotations. Annotations can be associated with the entry node of the activity diagram to indicate the global constraints on QoS dimensions and the ranking of the QoS metrics to be optimized (see AN7 and AN4 in Figure 2). An annotation can indicate whether several abstract operations of the workflow correspond to the same stateful service (see annotation AN6, which is associated with two operations). The "Stateful service" annotation is also associated in Figure 2 to the "Analysis center reservation" service, shared with other workflows, which by its very nature cannot be moved locally. Finally, we assume that all loops are annotated with the maximum number of expected iterations (see annotation AN1, discussed below).

In summary, Figure 2 contains the following annotations relevant for our case study:

- $AN1$: The loop can be executed at most twice (in order to book either a blood analysis or a cardiology examination or both).
- $AN2$: The ECG must be performed within at most 5 minutes.
- $AN3$: The medical center analyses cost has to be lower than 200 Euros, which is the maximum budget available by the patient's health insurance.
- $AN4$: The medical examination must take less than 10 minutes.
- $AN5$: The Analysis Center reservation service is stateful.
- $AN6$: The Get ECG data and ECG processing operations are implemented by the same stateful service.
- $AN7$: The rank in the quality metrics is: availability, cost, time, and energy.

An annotation like $AN1$, which prescribes the maximum number of loop iterations, allows loop unfolding at compile time. The unfolding of the original activity diagram yields an expanded activity diagram that is represented by a directed acyclic graph (DAG). The unfolding is necessary for the optimization procedure to guarantee that global constraints are satisfied [4]. The value to choose for the bound can be evaluated from past executions by inspecting system logs or it can be specified by the software architect at design time, based on previous experience. Annotations like $AN2$ model a local constraint, which predicates on the quality attributes of a single abstract service. In our example, it prescribes that its execution time has to be lower than a given threshold. Vice-versa, annotations like $AN3$ and $AN4$ above model global constraints, i.e., those that predicate on the quality attributes of a group of abstract services (e.g., $AN3$) or they can specify a constraint for the whole application (e.g., $AN4$).

As in [6], we define the following concepts:

Execution paths represent all the possible runs of an application, given the upper bounds on loop iterations. Execution paths may include the execution of abstract services in sequence or in parallel but they do not include branches. Execution paths may be represented graphically as shown by the examples in Figure 3.

Sub-paths differ from execution paths since they cannot include parallel execution of abstract services. A sub-path of an execution path ep is a sequence of abstract services $[as_1, as_2, \ldots, as_n]$ encountered by traversing an execution path from its initial to its final node. For example, the leftmost execution path in Figure 3 defines the following subpaths $[as_1, as_2, as_5, as_7, as_8]$ and $[as_1, as_2, as_6, as_7, as_8]$.

Finally, notice that, as in other proposals in the literature [4], [6], [9], we do not consider exceptions in our optimization problem, and consequently global constraints are guaranteed only for the nominal execution of the application. While in principle exceptions could be easily included in our formulation of the optimization problem, our approach will lead to very conservative solutions. Indeed, we optimize for the worst case scenario in terms of service calls, e.g., we consider the maximum number of cycle iterations. In presence of exceptions, the worst case scenario is the one in which every possible exception happen, requiring every possible compensation action to be run. Clearly this would lead to a very conservative solution, which would be of limited interest for the end-users as it refers to a very unlikely scenario.

B. Quality Model

The problem of selecting the concrete service to be bound to an abstract service is a multi-objective problem, since several quality criteria can be associated with services execution. In this paper, we focus on the following set of quality dimensions, which have been the basis for QoS considerations also in other approaches [4], [6], [18]:

- *Response time*: A numeric value representing the expected delay between the time when an operation is invoked and the time when the result is obtained. Response time is measured in seconds.
- *Availability*: A value in the range [0,1] representing the probability that a given operation of a concrete service is available, i.e., its invocation at run-time will be performed successfully.

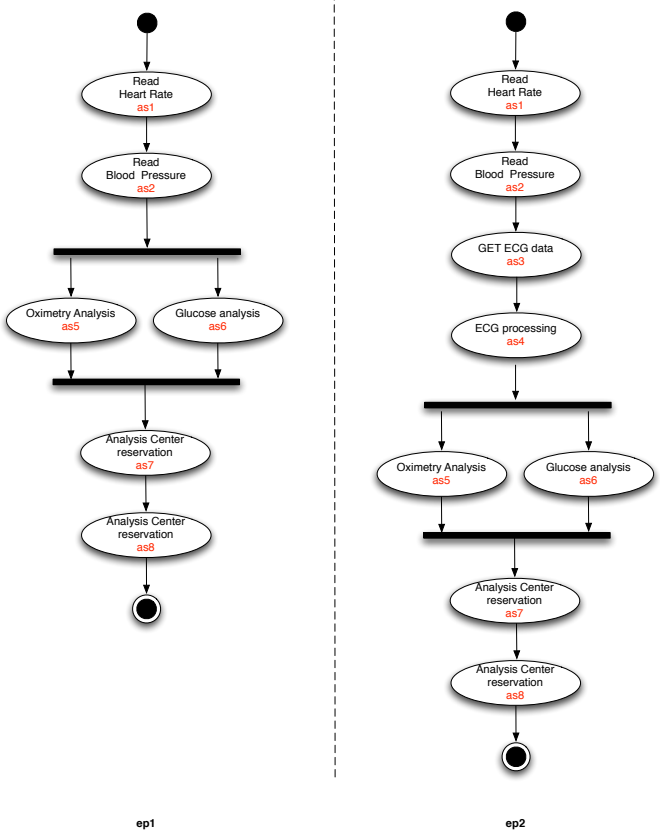

Fig. 3. Execution paths arising from the case study specification and annotations. Horizontal bars indicate the start and the end of parallel operations.

- *Cost*: A number representing the fee (e.g., in Euros) the end-user has to pay to the Service Provider for local or remote service operation invocation.
- *Energy*: A value representing the energy (in Joule) consumed for the deployment of concrete services and execution of operations on local devices.

In our approach, we assume that the end-user can express preferences among quality dimensions; e.g., response time may be valued more than cost. This is expressed by ranking the quality attributes (see annotation $AN7$ in Figure 2).

The response time of a concrete service operation is given by the sum of the time required for the request/response messages transmission and the time required to execute the service operation by a local device or by a remote service provider. Furthermore, if a concrete service has to be deployed locally, then also the deployment time has to be considered, which will further delay the first concrete service operation execution (but not the following ones, since the service is already locally available).

Let $\mathcal{I}$, $\mathcal{J}$, and $\mathcal{K}$ denote the finite sets of abstract services, concrete services, and devices available in the end-user neighbourhood to execute concrete services, respectively. Let $\overline{k} \in \mathcal{K}$ be the device that runs the Application Manager. Furthermore, let $\mathcal{OP}1_j$, where $j \in \mathcal{J}$, denote the set of operations implemented by the j-th concrete service and let $\mathcal{OP}2_i$, where $i \in \mathcal{I}$, be the set of operations that implement the i-th abstract service. Finally, let $\mathcal{S}_i$ denote the set of concrete services that can be bound to the abstract service $i \in \mathcal{I}$ and let $\mathcal{D}_j$ indicate the set of local devices that are available to execute the j-th concrete service.

To precisely characterize the response time of an operation $o \in \mathcal{OP}1_j$ offered by a service j, we will denote by $tw_{j,o,k}$ the time required for the data transfer when the service is executed on the device k, while $TW_{j,o}$ will indicate the time required for the data transfer when the service operation is executed remotely. Similarly, $t_{j,o,k}$ will indicate the execution time of service j operation $o \in \mathcal{OP}1_j$ locally on device k, while $T_{j,o}$ is the time for the remote execution of operation o guaranteed by the Service Provider. Finally, $d_{j,k}$ will denote the time required for the deployment of service j on device k. We assume that a concrete service is monolithic, i.e., the code of the whole set of operation $\mathcal{OP}1_j$ is downloaded and deployed on local devices.

In our framework, once the abstract to concrete service operation binding has been established, the deployment of all concrete services in the pervasive environment starts concurrently for all the involved devices. Alternatively, one could choose to postpone the deployment of concrete services on local devices until concrete services are invoked. This choice would save bandwidth and energy, but it would delay service execution and hence penalize performance.

We assume that concrete service operations have a different availability (cost) for local and remote execution, which will be denoted by $a_{j,o,k}$. ($c_{j,o}$) and $A_{j,o}$ ($C_{j,o}$), respectively. Notice that since physical devices might have different characteristics, the local availability $a_{j,o,k}$ of operation o of concrete service j , depends on the physical device k on which the service will be deployed.

Energy is also a critical factor in most pervasive systems, since battery-operated devices can support local execution only if their battery is not exhausted at the time the services are deployed and operations are executed locally. We will denote by $\hat{b}_{0,k}$ the initial energy of the device k when the application execution starts. Furthermore, we will assume that, as the time elapses, the batteries of devices discharge linearly with slope α_k. We will denote with $e_{j,k}$ the energy consumed for deploying service j on device k and with $\hat{e}_{j,o,k}$ the energy consumed for executing operation $o \in \mathcal{OP}1_j$ of service j on device k. Finally, we will indicate with $ew_{j,o,k}$ the energy consumed by a device $k \neq \overline{k}$ to communicate with the Application Manager for the execution of service j operation $o \in \mathcal{OP}1_j$, and with $EW_{j,o}$ the energy consumed by the Application Manager for the data transfer associated with the remote/local execution of operation o on another device.

For the sake of simplicity, we assume that the energy consumed by a device k to communicate with the Application Manager while executing an operation o only depends on the protocol used for such communication. As a result, the same energy $ew_{j,o,k}$ is also consumed by the Application Manager to communicate during the execution of operation o[1].

The global constraints specified by the end-users on the overall response time, availability, and cost will be denoted by $\overline{T}$, $\overline{A}$, and $\overline{C}$, respectively.

Each quality parameter that depends on the end-user context is evaluated at run-time; for example, the service deployment time, which depends on the service size and on the bandwidth of the wireless network connecting the devices. Conversely, quality parameters that are intrinsic characteristics of each service are stored and retrieved by the Optimizer module from the registry. In particular, if the same service is accessible from the same provider but with different quality characteristics (i,e., quality level), then multiple copies of the same service will be stored in the registry, each copy being characterized by its own quality profile.

As a final remark, we observe that the parameters related with executing a service operation on specific devices (e.g., the energy consumed to execute the operation or to communicate some results) can be roughly estimated by looking at the characteristics of the devices involved and at the operation itself, e.g., by knowing the MIPS/MFLOPS of a micro-controller and the number of instructions of the operation. Part of this information could be provided by device producers and by service providers and they could be stored in the registry. Alternatively, the same figures could be calculated off-line, once for all, by testing each device with the different services or relying on some benchmarking data.

[1]This assumption does not change the nature of the optimization problem and could be easily relaxed.

Table I summarizes the notation introduced so far and adopted in the remainder of the paper.

IV. Optimization model

The main decisions which have to be taken to solve the service selection problem are: (i) determining if services have to be executed locally or remotely, (ii) in case of local execution, determining which is the device that will be devoted to host each concrete service, and (iii) assigning abstract services to specific operations of concrete services.

The following decision variables are used to model this optimization problem:

- $y_{i,j}$: Equals 1 if concrete service j is remotely executed to support abstract service i, 0 otherwise;
- $z_{i,j,k}$: Equals 1 if concrete service j is locally deployed and executed by device k to support abstract service i, 0 otherwise;
- $\tilde{y}_{i,j,o}$: Equals 1 if the abstract service i is executed by operation $o \in OP1_j$ of concrete service j remotely, 0 otherwise;
- $\tilde{z}_{i,j,o,k}$: Equals 1 if the abstract service i is executed by operation $o \in OP1_j$ of concrete service j locally by device k, 0 otherwise.

Finally, in order to correctly evaluate the time and energy required to deploy concrete services in the pervasive environment, the following binary variables are introduced[2]:

- $x_{i,j,k}$: Equals 1 if concrete service j is deployed on device k and invoked for the first time to execute abstract service i, 0 otherwise;
- $w_{i,j,k}$: Equals 1 if concrete service j has already been deployed on device k when abstract service i is executed, 0 otherwise.

The service selection problem includes constraints associated with services assignment, stateful services, cost, time, energy, and availability.

Service assignment constraints: For each abstract service i, at most one remote service (operation) can be selected. Hence, among the variables $y_{i,j}$ ($\tilde{y}_{i,j,o}$) at most one can be set to 1. Thus, the following conditions holds:

$$\sum_{j \in \mathcal{S}_i} y_{i,j} \leq 1, \ \forall i \in \mathcal{I}, \tag{1}$$

$$\sum_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} \tilde{y}_{i,j,o} \leq 1, \ \forall i \in \mathcal{I}. \tag{2}$$

Similarly, considering the pervasive environment, at most one concrete service (operation) deployed on a local device performs abstract service i execution, hence:

$$\sum_{j \in \mathcal{S}_i, k \in \mathcal{D}_j} z_{i,j,k} \leq 1, \ \forall i \in \mathcal{I}, \tag{3}$$

$$\sum_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i, k \in \mathcal{D}_j} \tilde{z}_{i,j,o,k} \leq 1, \ \forall i \in \mathcal{I}. \tag{4}$$

[2]Recall that in our framework the deployment starts when the application execution begins for the whole set of concrete services that have to be deployed locally.

TABLE I
SERVICE SELECTION PROBLEM PARAMETERS AND DECISION VARIABLES

Application Parameters

$\mathcal{I}$	Set of abstract service indexes
$\mathcal{J}$	Set of concrete service indexes
$\mathcal{K}$	Set of device indexes
ep_l	l-th execution path
sp^l_m	m-th sub-path belonging to execution path l
L	Number of execution paths generated by unfolding the application workflow
$\mathcal{S}_i \subseteq J$	Set of indexes of concrete services supporting abstract service i execution
$\mathcal{OP}1_j$	Set of indexes of operations implemented by concrete service j
$\mathcal{OP}2_j$	Set of indexes of operations implementing abstract service i
$\mathcal{D}_j \subseteq \mathcal{K}$	Set of local device indexes supporting concrete service j execution

QoS Parameters

$tw_{j,o,k}$	Time required for the data transfer when the service j operation $o \in \mathcal{OP}1_j$ is executed on the device k	Evaluated at run-time
$TW_{j,o}$	Time required for the data transfer when service j operation $o \in \mathcal{OP}1_j$ is executed remotely	Evaluated at run-time
$t_{j,o,k}$	Local execution time of service j operation $o \in \mathcal{OP}1_j$ on device k	Evaluated at run-time
$T_{j,o}$	Execution time for the remote execution of service j operation $o \in \mathcal{OP}1_j$	Stored in the SMScom registry
$d_{j,k}$	Time required for the deployment of service j on device k	Evaluated at run-time
$a_{j,o,k}$	Availability of service j operation $o \in \mathcal{OP}1_j$ when executed locally by device k	Evaluated at run-time
$A_{j,o}$	Availability for the remote execution of service j operation $o \in \mathcal{OP}1_j$	Stored in the SMScom registry
$c_{j,o}$	Cost for the local execution of service j operation $o \in \mathcal{OP}1_j$	Stored in the SMScom registry
$C_{j,o}$	Cost for the remote execution of service j operation $o \in \mathcal{OP}1_j$	Stored in the SMScom registry
$\hat{b}_{0,k}$	Initial energy level of device k	Evaluated at run-time
α_k	Devices discharge parameter slope	Evaluated at run-time
$e_{j,k}$	Energy consumption for the deployment of service j on device k	Evaluated at run-time
$\hat{e}_{j,o,k}$	Energy consumption for the execution of service j operation $o \in \mathcal{OP}1_j$ on device k	Evaluated at run-time
$ew_{j,o,k}$	Energy consumed by the device k for the data transfer with the application manager for the execution of service j operation $o \in \mathcal{OP}1_j$	Evaluated at run-time
$EW_{j,o}$	Energy consumed by the application manager during the data transfer of the remote/local execution of service j operation $o \in \mathcal{OP}1_j$	Evaluated at run-time

Decision Variables

$y_{i,j}$	1 if concrete service j is remotely executed to support abstract service i, 0 otherwise
$z_{i,j,k}$	1 if concrete service j is executed locally by device k to support abstract service i, 0 otherwise
$v_{i,j,o}$	1 if abstract service i is executed by operation $o \in \mathcal{OP}1_j$, 0 otherwise
$x_{i,j,k}$	1 if concrete service j is deployed on device k and invoked for the first time to execute abstract service i, 0 otherwise
$w_{i,j,k}$	1 if concrete service j has already been deployed on device k when abstract service i is executed, 0 otherwise

Finally, every abstract service has to be executed either locally or remotely:

$$\sum_{j \in \mathcal{S}_i,} \left(y_{i,j} + \sum_{k \in \mathcal{D}_j} z_{i,j,k} \right) = 1, \ \forall i \in \mathcal{I}, \tag{5}$$

$$\sum_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} \left(\tilde{y}_{i,j,o} + \sum_{k \in \mathcal{D}_j} \tilde{z}_{i,j,o,k} \right) = 1, \ \forall i \in \mathcal{I}. \tag{6}$$

Stateful services constraints: If two abstract services i_1 and i_2 have to be executed by the same concrete service, then the following constraint families are introduced:

$$y_{i_1,j} = y_{i_2,j}, \forall j \in \mathcal{S}_{i_1} \cap \mathcal{S}_{i_2} \tag{7}$$

$$y_{i_1,j} = 0, \forall j \in \mathcal{S}_{i_1} \setminus \mathcal{S}_{i_2} \tag{8}$$

$$y_{i_2,j} = 0, \forall j \in \mathcal{S}_{i_2} \setminus \mathcal{S}_{i_1} \tag{9}$$

$$z_{i_1,j,k} = z_{i_2,j,k}, \forall j \in \mathcal{S}_{i_1} \cap \mathcal{S}_{i_2}, \forall k \in \mathcal{D}_j \tag{10}$$

$$z_{i_1,j,k} = 0, \forall j \in \mathcal{S}_{i_1} \setminus \mathcal{S}_{i_2}, \forall k \in \mathcal{D}_j \tag{11}$$

$$z_{i_2,j,k} = 0, \forall j \in \mathcal{S}_{i_2} \setminus \mathcal{S}_{i_1}, \forall k \in \mathcal{D}_j \tag{12}$$

Cost constraints: The costs for the local and remote execution of a concrete service j are given by c_j and C_j, respectively. Since, for equation (6), exactly one variable among $\tilde{y}_{i,j,o}$ and $\tilde{z}_{i,j,o,k}$ is raised to 1, the cost of execution of the abstract service i can be computed as follows:

$$cost_i = \sum_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} \left(C_{j,o} \cdot \tilde{y}_{i,j,o} + \sum_{k \in \mathcal{D}_j} c_{j,o} \cdot \tilde{z}_{i,j,o,k} \right), \quad \forall i \in \mathcal{I}. \tag{13}$$

The cost of each execution path l can be computed as the sum of the costs of the corresponding abstract services:

$$\hat{C}_l = \sum_{i \in ep_l} cost_i, \ \forall l \in [1, L], \tag{14}$$

hence the cost global constraint can be expressed as:

$$\hat{C}_l \leq \overline{C}, \ \forall l \in [1, L], \tag{15}$$

while the overall cost of the application C can be computed conservatively as the maximum cost over all the execution paths:

$$\hat{C}_l \leq C, \ \forall l \in [1, L]. \tag{16}$$

Alternatively, C can be computed as the average value over all the execution paths, as in [6]. In this paper, we will always take the worst case perspective.

Time constraints: The time experienced by the end-users for the execution of each abstract service must consider the execution time of the corresponding concrete service, the possible deployment of the concrete service on a device, and the time required for the data transfers between each concrete service and the application manager. Since for all i exactly one variable among $\tilde{y}_{i,j,o}$ and $\tilde{z}_{i,j,o,k}$ is raised to 1, the time spent for the concrete service execution is given by:

$$\begin{aligned} exeTime_i &= \sum_{j\in\mathcal{S}_i, o\in\mathcal{OP}1_j\cap\mathcal{OP}2_i} (T_{j,o}\cdot\tilde{y}_{i,j,o} + \\ &+ \sum_{k\in\mathcal{D}_j} t_{j,o,k}\cdot\tilde{z}_{i,j,o,k}),\ \forall i\in\mathcal{I}. \end{aligned} \quad (17)$$

Similarly, the data transfer time associated with abstract service i execution is given by:

$$\begin{aligned} dataTransfTime_i &= \sum_{j\in\mathcal{S}_i, o\in\mathcal{OP}1_j\cap\mathcal{OP}2_i} (TW_{j,o}\cdot\tilde{y}_{i,j,o} + \\ &+ \sum_{k\in\mathcal{D}_j\setminus\{\overline{k}\}} tw_{j,o,k}\cdot\tilde{z}_{i,j,o,k}),\ \forall i\in\mathcal{I}, \end{aligned} \quad (18)$$

Indeed, if the service is hosted on the Application Manager $\overline{k}$, the data transfer time is null since it is performed in the device RAM.

The time required to deploy the concrete service j on a device k to support abstract service i is given by:

$$deployTime_{i,j,k} = x_{i,j,k}\cdot d_{j,k},\ \forall i\in\mathcal{I}, j\in\mathcal{S}_i, k\in\mathcal{D}_j. \quad (19)$$

The variables $x_{i,j,k}$, $w_{i,j,k}$, and $z_{i,j,k}$ are related as follows:

$$x_{i,j,k} = z_{i,j,k} - w_{i,j,k},\ \forall i\in\mathcal{I}, j\in\mathcal{S}_i, k\in\mathcal{D}_j \quad (20)$$

$$\sum_{\overline{i}\in Prec(i,l)} z_{\overline{i},j,k} \le M\cdot w_{i,j,k},\ \forall l\in[1,L], \forall i\in ep_l, j\in\mathcal{S}_i, k\in\mathcal{D}_j, \quad (21)$$

where $Prec(i,l)$ denotes the set of abstract services which precede abstract service i along a given execution path, while M is any constant greater or equal to $|\mathcal{I}|$. In this way, when the concrete service j is executed for the really first time on the device k along execution path l, both $x_{i,j,k}$ and $z_{i,j,k}$ are raised to one. Then, if the concrete service j is invoked again on device k, equation (21) raises $w_{i,j,k}$ to 1, $z_{i,j,k}$ is also set to 1, while $x_{i,j,k}$ is forced to 0.

If we denote by ds_i and df_i the starting and finishing time of the deployment of any concrete service for the execution of abstract service i, then the following condition holds:

$$df_i \ge ds_i + \sum_{\overline{i}\in Prec(i,l)} \sum_{j\in\mathcal{S}_i, k\in\mathcal{D}_j} deployTime_{\overline{i},j,k},\ \forall l\in[1,L], \forall i\in ep_l. \quad (22)$$

Let us denote by $ts_{i,m}$ and $tf_{i,m}$ the starting and finishing time of abstract service i along the sub-path m. The execution of abstract service i can start after the finishing deployment time, hence:

$$ts_{i,m} \ge df_i,\ \forall l\in[1,L], \forall sp^l_m\in ep_l, \forall i\in sp^l_m, \quad (23)$$

and the abstract service finishing time is given by:

$$tf_{i,m} = ts_{i,m} + exeTime_i +$$

$$dataTransfTime_i,\ \forall l\in[1,L], \forall sp^l_m\in ep_l, \forall i\in sp^l_m. \quad (24)$$

Moreover, the execution of an abstract service i can start only after the finishing time of its direct predecessor along the sub-path m ($DPrec(i,m)$):

$$ts_{i,m} \ge tf_{\overline{i},m},\ \forall l\in[1,L], \forall sp^l_m\in ep_l, \forall i\in sp^l_m, \forall\overline{i}\in DPrec(i,m). \quad (25)$$

Finally, the execution time of each execution path l is equal to the finishing time of the last abstract service n, hence the execution time global constraint can be expressed as:

$$tf_{n,l} \le \overline{T},\ \forall l\in[1,L], \quad (26)$$

and the execution time of the application T is computed as the maximum execution time over all the execution paths:

$$tf_{n,m} \le T,\ \forall l\in[1,L], \forall sp^l_m\in ep_l. \quad (27)$$

Energy constraints: In order to execute concrete services on local devices, their residual energy after the deployment must be positive and greater than the energy required to execute the concrete service. The energy consumption for deployment of concrete service j on device k to support abstract service i execution is given by:

$$deployEnergy_{i,j,k} = e_{j,k}\cdot x_{i,j,k}\forall i\in\mathcal{I}, j\in\mathcal{S}_i, k\in\mathcal{D}_j, \quad (28)$$

while the energy consumption for the concrete service execution is given by:

$$eExecution_{i,j,k} = \sum_{o\in\mathcal{OP}1_j\cap\mathcal{OP}2_i} \hat{e}_{j,o,k}\cdot\tilde{z}_{i,j,o,k}\forall i\in\mathcal{I}, j\in\mathcal{S}_i, k\in\mathcal{D}_j \quad (29)$$

The residual energy of any device different to the Application Manager after the execution of an abstract service i along the execution path l can be calculated as follows:

$$\begin{aligned} b_{i,l,k} &= \hat{b}_{0,k} - \alpha_k\cdot ts_{i,l} + \\ &- \sum_{\overline{i}\in Prec(i,l), j\in\mathcal{S}_i} \Big(eExecution_{\overline{i},j,k} - deployEnergy_{i,j,k} + \\ &- \sum_{o\in\mathcal{OP}1_j\cap\mathcal{OP}2_i} ew_{j,o,k}\cdot\tilde{z}_{i,j,o,k}\Big), \\ &\forall l\in[1,L], \forall i\in ep_l, \forall k\ne\overline{k} \end{aligned} \quad (30)$$

The first term is the initial energy of the device, the second term entails the energy discharged during the application execution, while the term in parentheses takes into account the energy consumed for the deployment, execution and data transfer with the Application Manager for the invocation of concrete services hosted on the device k.

Particular attention must be reserved to the Application Manager device, whose residual energy can be calculated as follows:

$$b_{i,l,\overline{k}} = \hat{b}_{0,\overline{k}} - \alpha_{\overline{k}} \cdot ts_{i,l} - \sum_{\overline{i} \in Prec(i,l), j \in \mathcal{S}_i} \Big(eExecution_{\overline{i},j,\overline{k}} + - deployEnergy_{i,j,\overline{k}} - \sum_{o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} ew_{j,o,\overline{k}} \cdot \tilde{z}_{i,j,o,\overline{k}} + - \sum_{o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} EW_{j,o} \cdot \tilde{y}_{i,j,o} \Big), \forall l \in [1,L], \forall i \in ep_l, \quad (31)$$

where the last term in parentheses takes into account the energy consumed for the invocation of concrete services hosted by remote Service Providers.

After the execution of any abstract service i, the residual energy of every device cannot be negative, hence:

$$b_{i,l,k} \geq 0, \forall l \in [1,L], \forall i \in ep_l, \forall k \in \mathcal{K}. \quad (32)$$

When the last abstract service n is executed, the difference between the initial and the residual energy of device k along the execution path l equals the overall energy consumed for the application along the execution path. Hence, the energy consumed for the whole application equals the maximum of the sum of energy consumption of every device:

$$E \geq \sum_{k \in \mathcal{K}} \hat{b}_{0,k} - b_{n,l,k}, \forall l \in [1,L]. \quad (33)$$

Availability constraints: The availability of every abstract service can be computed as:

$$\hat{a}_i = \prod_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} A_{j,o}^{\tilde{y}_{i,j,o}} \prod_{j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i, k \in \mathcal{D}_i} a_{j,o,k}^{\tilde{z}_{i,j,o,k}}, \quad \forall i \in \mathcal{I} \quad (34)$$

while the availability of an execution path l is given by:

$$\hat{A}_l = \prod_{i \in epl} \hat{a}_i, \forall l \in [1,L]. \quad (35)$$

Hence, the availability global constraints can be introduced as follows:

$$\hat{A}_l \geq \overline{A}, \forall l \in [1,L], \quad (36)$$

and the availability of the application A can computed as the minimum availability over all the execution paths:

$$\hat{A}_l \geq A, \ \forall l \in [1,L]. \quad (37)$$

The availability formula can be linearized by applying the logarithm function; for example constraint (36) becomes:

$$ln(\hat{A}_l) = \sum_{i \in ep_l, j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i} ln(A_{j,o}) \cdot \tilde{y}_{i,j,o} + + \sum_{i \in ep_l, j \in \mathcal{S}_i, o \in \mathcal{OP}1_j \cap \mathcal{OP}2_i, k \in \mathcal{D}_j} ln(a_{j,o,k}) \cdot \tilde{z}_{i,j,o,k} \leq ln(\overline{A}), \forall l \in [1,L]. \quad (38)$$

The service selection problem includes only binary and continuous variables and linear constraints, hence it is a Mixed Integer Linear Programming (MILP) problem. The problem is multi objective. In our approach, the end-user preferences induce an ordering on the optimization of quality attributes. The optimization problem is solved iteratively, once for each quality attribute, and the optimum value obtained as a solution is introduced as a constraint in the subsequent problem. For example, according to the preferences specified by annotation $AN7$ in Figure 2, the following problems are solved:

- $optA = \max A$
  subject to: (1)-(37)
- $optC = \min C$
  subject to: (1)-(37)
  $$A = optA$$
- $optT = \min T$
  subject to: (1)-(37)
  $$A = optA$$
  $$C = optC$$
- $\min E$
  subject to: (1)-(37)
  $$A = optA$$
  $$C = optC$$
  $$T = optT$$

V. RUN-TIME RE-OPTIMIZATION

In pervasive environments, the dynamic nature of the service selection problem and the high variability of service and device parameters ask for solutions that can accommodate run-time re-optimization.

For example, a re-optimization step should be performed if a service operation fails or provides degraded performance, if a device becomes unavailable because of battery depletion or because it goes out range of the Application Manager network, etc. On the other hand, run-time re-optimization has to be performed carefully since it might introduce a significant overhead (e.g., the UDDI registry has to be accessed again, device specific parameters like energy consumption and execution time parameters for new candidate services have to be evaluated, etc.).

As in other approaches [6], [9], the basic idea we follow is to monitor the QoS of service operation invocations and to re-estimate the quality values expected for the running application. Whenever the new estimates indicate a significant deviation from the values obtained by the initial optimal solution, abstract service invocations that still remain to be executed must be re-optimized in order to avoid QoS violations.

This basic approach is tailored to the SMScom framework by triggering re-optimization in the following cases only:

- If the current QoS values differ from their corresponding prediction more than a given threshold. This addresses the problem of variability of services and local device performance, while keeping the overhead under control through a careful selection of the threshold.
- If a service operation invocation fails, due to a fault in the remote Service Provider or a because the local device assigned for execution has became unreachable, then a substitute service operation has to be selected through a re-optimization step.

- Since the optimization is performed by evaluating a priori the maximum number of cycle iterations, a re-optimization is triggered when a loop execution ends and the current number of iterations differs from the expected one.
- Finally, a re-optimization is also performed periodically to take into account highly dynamic environments in which new candidate services and local devices may appear during long runs of the application.

Notice that the QoS constraints and the optimization parameters may also adapt to the execution context, possibly asking for an explicit intervention of the end-user, who triggers a new re-optimization. As an example, if the battery level of the Application Manager has reached a critical level, then the end-user might be triggered to revise the QoS metrics ranking, prioritizing energy consumption over availability, or relaxing the execution time constraint (in some conditions tolerating lower performance may reduce energy consumption).

Re-optimization requires some information on the current state of the application execution. It starts by revising the application's unfolded activity diagram represented by a DAG, eliminating the abstract services that belong to conditional branches that were not followed during the execution. Additional constraints can be introduced for those abstract services that were already executed, setting their quality parameters to the values monitored for service invocations (see [4], [19], for further details). In general, these two steps allows to remove some decision variables and abstract services from the re-optimization model, which, in turn, reduces the computational effort required for the re-optimization w.r.t. the initial optimization.

VI. Experimental results

To solve our optimization problems we used the IBM ILOG CPLEX 12.1 tool, which implements a parallel branch and cut procedure [20]. In order to prove scalability of the proposed model, an extensive experimental analysis has been performed. The physical system supporting the experiments is based on VMWare ESXi 4.0, running on an Intel Nehalem dual socket quad-core system with 32 GB of RAM. CPLEX is hosted in a Virtual Machine (VM) running Ubuntu 11.04 Linux. The VM has four physical cores dedicated to its execution with guaranteed performance and 8GB of memory reserved.

This section is structured into two parts: The former aims at proving the scalability of the approach, the latter shows the results of the case study simulation.

A. Scalability of the optimization model

The model has been tested considering a large set of randomly generated instances, where the application parameters and QoS values have been varied as reported in Table II. Problems with up to 40 abstract services, 160 concrete services (with up to 5 operations each), and 15 devices have been considered. The set of concrete services supporting abstract services and the set of local devices supporting concrete service execution were randomly generated. The QoS parameters were randomly generated assuming a uniform distribution in intervals chosen as follows.

Availability values: Minimum and maximum availability values were randomly generated in the range [0, 1].

Cost values: As in other approaches proposed in the literature [6], [21], we assume that the cost of a service operation invocation is inversely proportional to the execution time and depends exponentially on availability. Hence, it is evaluated as:

$$c_{j,o} = \eta_i \cdot \frac{1}{t_{j,o,k}} \cdot e^{\delta_i \cdot a_{j,o,k}},$$

where η_i and δ_i are constants that depend on the particular functionality implemented by the abstract service (i.e., the higher the complexity of the functionality, the higher the cost).

Time values: Time parameters were calculated by considering the Internet for remote communication and ZigBee for local communication. Internet, based on WIFI connection, is characterized by 5 ms latency ($i\_latency$) and 2 Mb/s bandwidth (i_bw) [22], while ZigBee by 8 ms latency ($z\_latency$) and 250 Kb/s bandwidth (z_bw) [23], [24]. The service size (s_size) has been randomly generated in the range (500, 2000) byte, while the data size (d_size) in the range (10, 4000) byte according to the data reported in [25]. Hence, the time parameters have been randomly generated in the following ranges:

$$d_{j,k} = i\_latency + \frac{s\_size}{i\_bw} + z\_latency + \frac{s\_size}{z\_bw} = (65, 221)ms$$

$$tw_{j,o,k} = z\_latency + \frac{d\_size}{z\_bw} = (18, 417)ms$$

$$TW_{j,o} = i\_latency + \frac{d\_size}{i\_bw} = (10, 25)ms$$

$$t_{j,o,k} = (0.002, 1)s$$

$$T_{j,o} = (0.01, 40)s$$

Energy values: For each device, the initial energy has been considered in the range (5000, 18000) J [26], while the power consumption has been set equal to 6 mW for execution ($execution\_power$) [27], 65 mW for ZigBee transmission (z_power) [24], 1.5 W for WiFi transmission (i_power) [22] and 40 μW in idle state ($idle\_power$). By considering these parameters and the chosen time values, the energy parameters have been randomly generated in the following ranges:

$$e_{j,k} = d_{j,k} \cdot z\_power = (4, 14)mJ$$

$$\hat{e}_{j,k} = t_{j,o,k} \cdot execution\_power = (0.012, 6)mJ$$

$$ew_{j,o,k} = tw_{j,o,k} \cdot z\_power = (1, 27)mJ$$

$$EW_{j,o} = TW_{j,o} \cdot i\_power = (15, 32)mJ$$

$$\alpha_k = (d_{j,k} + t_{j,k} + tw_{j,k}) \cdot idle\_power = (3.4, 65.52)\mu J$$

In order to evaluate the performance of the proposed optimization model, the average optimization time required for instances of variable size for the iterated multi-objective optimization is reported in Table III. Each value reported in

TABLE II
VALUES OF APPLICATION AND QOS PARAMETERS

Application Parameters

$\mathcal{I}$	20, 30, 40
$\mathcal{J}$	80, 120, 160
$\mathcal{K}$	5, 10, 15

QoS Parameters

$a_{j,o,k}$	(0, 1)	
$A_{j,o}$	(0, 1)	
$c_{j,o}$	(0.02, 27)	[$]
$C_{j,o}$	(0.02, 27)	[$]
$d_{j,k}$	(0.065, 0.221)	[s]
$t_{j,o,k}$	(0.002, 1)	[s]
$T_{j,o}$	(0.01, 40)	[s]
$tw_{j,o,k}$	(0.018, 0.417)	[s]
$TW_{j,o}$	(0.01, 0.025)	[s]
$\hat{b}_{0,k}$	(5000, 18000)	[J]
$e_{j,k}$	(0.004, 0.014)	[J]
$\hat{e}_{j,o,k}$	(0.000012, 0.006)	[J]
$ew_{j,o,k}$	(0.001, 0.027)	[J]
$EW_{j,o}$	(0.015, 0.032)	[J]
α_k	(0.000003, 0.000065)	[J]

the table corresponds to the average time over 10 randomly generated instances. A total number of 180 test cases have been considered.

The CPLEX solver execution time has been limited to 15 s for each objective. However, no difference in the final solution is observed with respect to executions without any time limits, hence each objective is optimally solved for every randomly generated instance considered. For problem instances of maximum size (which are almost one order of magnitude larger than the current applications executed in pervasive environments), the overall CPLEX execution time is around 30 s, while a few seconds are enough for typical problems.

TABLE III
AVERAGE OPTIMIZATION EXECUTION TIME (IN SECONDS)

	$\|K\|$=5			$\|K\|$=10			$\|K\|$=15		
($\|I\|$,$\|J\|$)	80	120	160	80	120	160	80	120	160
20	2.04	3.43	4.88	4.5	6.89	9.47	6.55	10.94	14.94
40	6.06	11.23	13.49	12.86	20.36	31.26	20.4	33.61	33.2

B. Case study execution

In this section we illustrate a simulation of the healthcare case study. As described in Section II, in the envisioned scenario the following local devices are available: i) body sensors, which collect data on biological and physiological parameters, ii) a local PC, and iii) the hand-held device with the role of Application Manager.

The scenario is composed of three main activities, each of which corresponds to a macro-functionality: i) data reading, executable only by the sensors, ii) data processing, executable both locally and remotely, and iii) analysis reservation, executable only remotely.

The characteristics of the candidate concrete services are reported in Table IV. An abstract service is supported on average by two concrete services with up to two operations each.

TABLE IV
CASE STUDY CONCRETE SERVICE OPERATIONS AND DEVICE CANDIDATES

Concrete service	Operation	Supported abstract services	Supporting devices
s_1	o_1	as_1	sensor
s_2	o_2	as_1	sensor
s_3	o_3	as_2	sensor
s_4	o_4	as_2	sensor
s_5	o_5	as_3	sensor
s_5	o_6	as_4	sensor
s_6	o_7	as_3	sensor
s_6	o_8	as_4	sensor
s_7	o_9	as_5	PC, sensor, hand-held, *remote*
s_8	o_{10}	as_6	PC, sensor, hand-held, *remote*
s_9	o_{11}	as_5	PC, sensor, hand-held, *remote*
s_{10}	o_{12}	as_6	PC, sensor, hand-held, *remote*
s_{11}	o_{13}	as_7, as_8	*remote*
s_{12}	o_{14}	as_7, as_8	*remote*

The execution is performed in three steps, as shown in Figure 4. The optimum solution is found and the execution of the application starts. Heart rate, blood pressure, and ECG are provided by the corresponding sensors, whereas the oximetry and glucose analysis are assigned to the hand-held device, while the reservations are assigned to the remote service s_{11} (see Figure 4a).

Due to a failure of the ECG get data implemented by concrete service s_5, a re-optimization is triggered (see Figure 4b) and a different concrete service (s_6) is selected. Finally, at the end of the analysis, the application establishes that the patient requires only one additional exam at an Analysis Center. Hence, a second re-optimization is performed (see Figure 4c) since the current number of iterations of the loop differs from the one expected at design-time. Then, the reservation is performed by the new service s_{12} and the application execution ends.

Together, the three diagrams in Figure 4 show that our approach easily addresses the various situations encountered in our case study, always finding the optimal solution to end the workflow. Most important for our research is the time required for finding this optimal solution: in our case study, optimization and re-optimization have always been performed in less than 100 ms. This shows that in a typical pervasive scenario the time required to find an optimal solution is negligible with respect to the time needed to run the workflow.

VII. RELATED WORK

Pervasive computing [1] has recently attracted attention both from industry and academia. Enabling technology, like Mobile Ad-Hoc Networks (MANETs) [28], [29], Wireless Sensor Networks (WSNs) [10], [30], Wireless Sensor and Actuator Networks (WSANs) [31], have rapidly developed and are now moving from the research to the production stage. Progress in technology has been only partially accompanied by progress in design methodology. As a result, pervasive and situation-aware applications are presently designed in a largely ad-hoc manner, and this may limit their use in large-scale settings and/or settings that require high dependability. A promising research direction is now focusing on designing

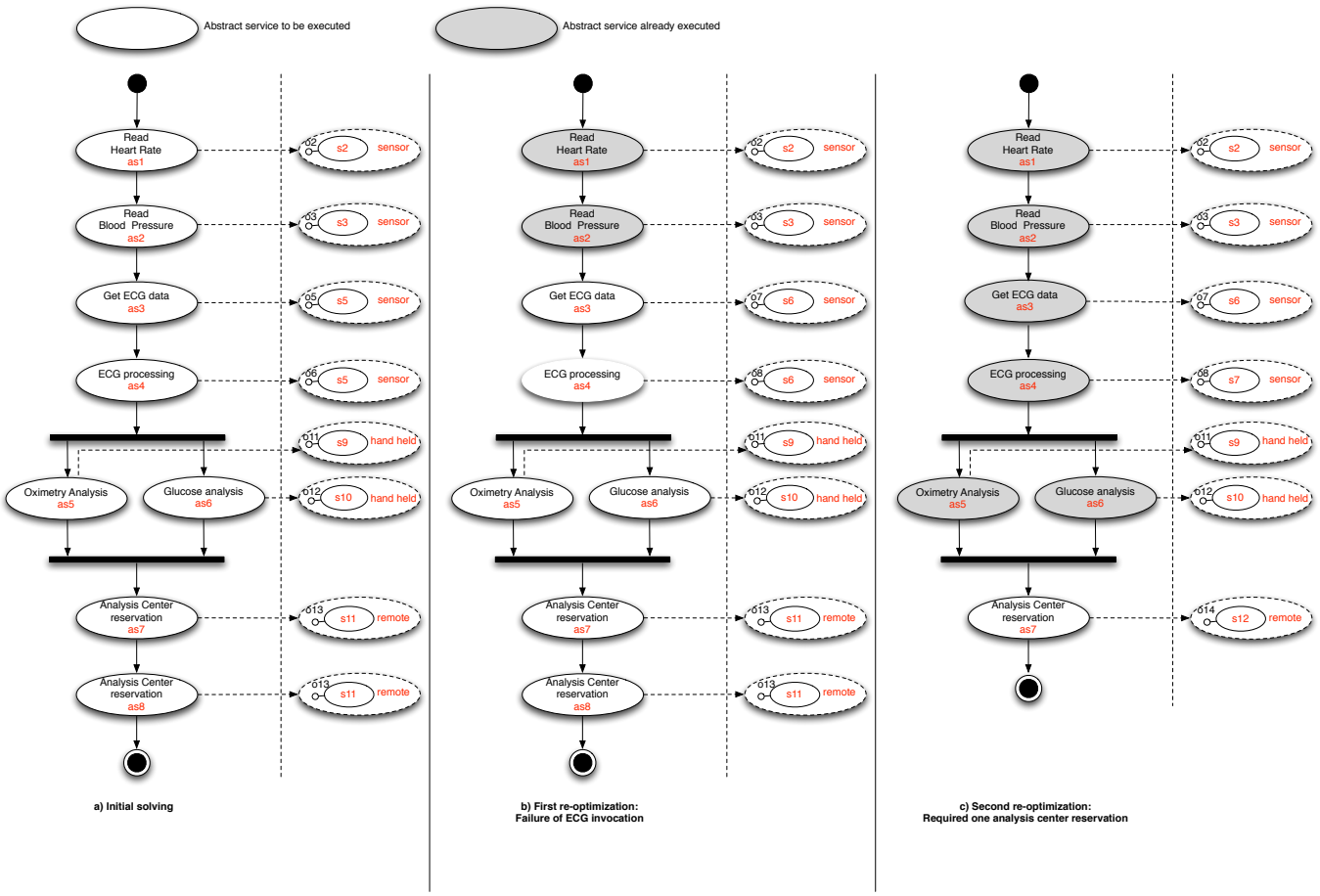

Fig. 4. Case study execution

pervasive and situation-aware applications based on the concepts developed in the area of Service-Oriented Computing (SOC). Although SOC has been originally proposed mainly for Internet-wide, business-to-business scenarios, it is now increasingly proposed as an effective approach to systematic design of pervasive systems [32], [33]. In particular, industrial proposals like Jini [34], Universal Plug and Play [35], and OSGI [36] together with a large body of academic research (e.g., [37]–[43]) address the issues related with implementing Service–Oriented Architectures (SOAs) in generic pervasive environments involving embedded and mobile devices. More recently, several academic researchers have matured the idea of using SOAs in WSNs and WSANs to reduce the complexity in building applications involving such technologies. This idea was explored by developing different middleware systems to let individual sensor nodes act as service providers [15], [44]–[49].

In SOA systems, building applications through the composition of available services is a key point [3]. Current research approaches can be classified into two main categories: composition by planning and business process optimization [50]. The former approach, proposed by the Semantic Web and AI communities, investigates the problem of synthesizing a complex behaviour from an explicit goal and a set of candidate services which contribute to a partial solution of the complex problem. In the latter case [4]–[6], [51], complex applications are specified as workflow processes and the best available set of services are dynamically selected at run-time by solving an optimization problem. The Semantic Web and AI approach is very flexible since an application schema is built automatically or semi-automatically from a high level specification of the required functionality [52]–[55], but it is usually computation intensive and, from the QoS point of view, only sub-optimal solutions can be identified [52].

In process optimization, vice versa, the application schema is given and the optimum mapping of activities to component services candidate for their execution is identified. Process optimization has its roots in workflow scheduling problems where the mapping of tasks to resources has to be identified such that some temporal or resource constraints (i.e., agents which can support tasks executions) are met [56].

The literature has provided *three generations* of solutions. First generation solutions implemented *local* approaches [4], [6], [57] that select services one at a time by associating the running abstract activity to the best candidate service which supports its execution. Local approaches are very simple (the optimum solution can be identified by polynomial time algorithms), but they can guarantee only local QoS constraints [58]. Recently very efficient solutions have been provided within the Web Service challenge [59]–[62], which, however are able

only to identify the set of candidate services minimizing the response time or maximizing the process throughput.

Second generation solutions proposed *global* approaches [4], [9], [19], [51], [63], [64]. The set of services which satisfy the process constraints and user preferences for the whole application are identified before executing the process. In this way, QoS constraints can predicate at a global level. Second generation techniques are based on the solution of NP-hard optimization problems. In [65], the complexity of some variants of the global process optimization problem is analyzed, while an overview of heuristic techniques can be found in [64]. Global approaches have been proposed for the first time in [4], where the process optimization problem has been formalized as a MILP problem, solved by integer linear programming solvers. Some recent proposals face the process optimization problem by implementing genetic algorithms [9], [19], [63]. Genetic algorithms are more flexible than MILP approaches, since they allow considering also non-linear composition rules for the evaluation of the process QoS, but are less computationally efficient. In current implementations some execution time is wasted by generating also non-feasible solutions. More recently, in [57] process optimization has been modeled as a multiple choice multiple dimension knapsack problem and as a graph constrained optimum path problem and efficient heuristic techniques have been proposed. An efficient recursive branch and bound algorithm has also been proposed by [66].

Second generation solutions, requiring the solution of NP-hard problems, introduce a significant overhead in the system. To reduce optimization complexity, a number of solutions have been proposed which guarantee global constraints only for the critical path [4] (i.e., the path which corresponds to the highest execution time), or reduce loops to a single task [9], [19]. Another drawback of second generation solutions is that, if the end-user introduces severe QoS constraints for the process execution, i.e., limited resources which set the problem close to unfeasibility conditions (e.g., limited budget or stringent execution time limit), no solutions could be identified and the process execution fails [9], [19].

Third generation techniques [5], [6], [67] try to overcome the limitations of the previous approaches. In particular, the work described in [6] focuses on the execution of processes under severe QoS constraints. Negotiation is exploited if a feasible solution cannot be identified, to bargain QoS parameters with Service Providers offering services, reducing process invocation failures. The proposed approach has been proven particularly efficient for large process instances. A hybrid global/local approach has been proposed in [5] with the aim of reducing optimization complexity and allowing also a decentralized implementation of the optimization process. Recently in [67], a novel approach based on local search with the aim to maximize the QoS under probabilistic constraints has been proposed. The goal is to provide the solution which returns the best quality level q^*, such that the probability that the actual quality received by the end-user falls below $\overline{q}$ is within a prescribed threshold.

A related and complementary problem is addressed by [68], which focuses on the interdependencies of dynamic binding policies adopted by different compositions, which may be running concurrently. Since multiple bindings to the currently best performing concrete service may degrade its performance, different alternative strategies, including probabilistic and co-operative strategies, are explored and compared.

All of the above mentioned approaches only focus on the selection of available remote services and do not take into account the option of locally deploying and running services.

This paper is based on the work presented in [6], which has been extended in order to: (i) consider the local execution of services on the devices available in the pervasive environment, (ii) modelling network interaction more in more detail, (iii) implementing the optimization of multiple quality criteria incrementally, and (iv) taking into account energy constraints explicitly.

VIII. Conclusions and future work

This paper proposed a new approach to dynamic service selection and allocation, which optimizes the QoS exhibited by an application described as an abstract workflow. The abstract workflow orchestrates a number of abstract services that must be bound to single operations of concrete services by a selection procedure that performs the QoS optimization. Optimization includes deciding—whenever possible—whether a service should be executed remotely or whether it is preferable to deploy and execute it locally. The proposed framework for dynamic service composition aims at supporting situation-aware computing, where decisions about optimal service selection and allocation must be made dynamically.

Our future work will put this framework in practice in a number of case studies that are currently undertaken [11]. Further refinements of the optimization approach will be explored, including support of direct communication between services that are co-located on the same device, avoiding interaction and communication with the Application Manager. Finally, the optimization of multiple process instances competing in the same situational environment will be also considered.

Acknowledgment

This research has been partially funded by the European Commission, Programme IDEAS-ERC, Project 227977-SMScom. Authors thank Dr. Bernardetta Addis for many fruitful discussions on linearization of bi-linear constraints.

References

[1] F. Adelstein, S. Gupta, G. R. III, and L. Schwiebert, *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hill, 2004.

[2] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engg.*, vol. 15, pp. 313–341, December 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1459074.1459084

[3] K. Nakamura and M. Aoyama, "Value-based dynamic composition of web services," in *APSEC*, 2006, pp. 139–146.

[4] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnamam, and H. Chang, "QoS-aware middleware for web services composition," *IEEE Trans. on Software Engineering*, vol. 30, no. 5, May 2004.

[5] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *WWW2009 Proc.*, 2009.

[6] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. on Software Engineering*, vol. 33, no. 6, pp. 369–384, June 2007.

[7] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: a Research Roadmap," *Int. J. Cooperative Inf. Syst*, vol. 17, no. 2, pp. 223–255, 2008.

[8] A. Carzaniga, G. P. Picco, and G. Vigna, "Designing distributed applications with mobile code paradigms," in *Proceedings of the 19th international conference on Software engineering*, ser. ICSE '97. New York, NY, USA: ACM, 1997, pp. 22–32. [Online]. Available: http://doi.acm.org/10.1145/253228.253236

[9] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani, "A framework for qos-aware binding and re-binding of composite web services," *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754–1769, 2008.

[10] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. J. Wiley, May 2007.

[11] "Smscom porject web site." http://deepse.ws.dei.polimi.it/smscom/index.html.

[12] OASIS, "Standard WS-BPEL 2.0." [Online]. Available: http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf

[13] OASIS, "Uddi oasis standard," http://uddi.xml.org/.

[14] P. Plebani and B. Pernici, "URBE: Web Service Retrieval Based on Similarity Evaluation," *IEEE Transactions on Knowledge and Data Engineering.*, pp. 1629–1642, 2009.

[15] G. Cugola and A. Margara, "SLIM: Service Location and Invocation Middleware for Mobile Wireless Sensor and Actuator Networks," *International Journal of Systems and Service-Oriented Engineering*, pp. 60–74, 2010.

[16] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *SOSP 2001 Proc.*, 2001.

[17] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca, "Context information for knowledge reshaping," *Int. J. Web Eng. Technol.*, vol. 5, no. 1, pp. 88–103, 2009.

[18] S. Chandrasekaran, J. A. Miller, G. Silver, I. B. Arpinar, and A. P. Sheth, "Performance Analysis and Simulation of Composite Web Services," *Electronic Market: The Intl. Journal of Electronic Commerce and Business Media*, vol. 13, no. 2, pp. 120–132, 2003.

[19] G. Canfora, M. Penta, R. Esposito, and M. L. Villani, "QoS-Aware Replanning of Composite Web Services," in *ICWS 2005 Proc.*, 2005, orlando.

[20] L. Wolsey, *Integer Programming*. John Wiley and Sons, 1998.

[21] L. Zhang and D. Ardagna, "SLA Based Profit Optimization in Autonomic Computing Systems," in *ICSOC 2004 Proceedings*, New York, 2004, pp. 173–182.

[22] S. Litchfield, "How to: Know how much power each component of your smartphone uses," Website, 2009, http://www.allaboutsymbian.com/features/item/How_to_Know_how_much_power_each_component_of_your_smartphone_uses.php.

[23] B. Latre, P. D. Mil, I. Moerman, N. V. Dierdonck, B. Dhoedt, and P. Demeester, "Maximum Throughput and Minimum Delay in IEEE 802.15.4," *Mobile Ad-hoc and Sensor Networks*, 2005.

[24] C. P. from Texas Instruments, "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," Texas Instruments, Tech. Rep., 2006.

[25] R. E. Klabunde, "Cardiovascular physiology concepts," Website, 2007, http://www.cvphysiology.com/Arrhythmias/A009.htm.

[26] "Find the energy contained in standard battery sizes," Website, http://www.allaboutbatteries.com/Energy-tables.html.

[27] C. Antonopoulos, A. Prayati, T. Stoyanova, C. Koulamas, and G. Papadopoulos, "Experimental Evaluation of a WSN Platform Power Consumption," *Parallel and Distributed Processing Symposium, International*, vol. 0, pp. 1–8, 2009.

[28] C. Perkins, Ed., *Ad Hoc Networking*. Addison Wesley, 2000.

[29] C.-K. Toh, *Ad hoc mobile wireless networks*. Prentice Hall Inc., 2002.

[30] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. J. Wiley, 2005.

[31] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Networks*, vol. 2, no. 4, 2004.

[32] S. Kalasapur, M. Kumar, and B. Shirazi, "Evaluating service oriented architectures (soa) in pervasive computing," in *Proc. of the 4th Annual IEEE International Conference on Pervasive Computing and Communications*. Pisa, Italy: IEEE Press., March 2006.

[33] S. Helal, *The Landscape of Pervasive Computing Standards*. Morgan and Claypool, 2010.

[34] K. Arnold, R. Scheifler, J. Waldo, B. O'Sullivan, and A. Wollrath, *The JINI Specification*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[35] "Universal plug and play (upnp)," http://www.upnp.org/.

[36] O. Alliance, http://www.osgi.org/, October 2005.

[37] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "Service composition for mobile environments," *Mobile Networks and Applications*, vol. 10, no. 4, August 2005.

[38] U. Bellur and N. Narendra, "Towards service orientation in pervasive computing systems," in *Int. Conf. on Information Technology: Coding and Computing*, April 2005.

[39] S. Kalasapur, M. Kumar, and B. Shirazi, "Dynamic service composition in pervasive computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 7, July 2007.

[40] A. Bottaro, A. Gerodolle, and P. Lalanda, "Pervasive service composition in the home network," in *21st Int. Conf. on Advanced Networking and Application*, Niagara Falls, Ontario, Canada, May 2007.

[41] B. M. Sonia, N. Georgantasa, and V. Issarny, "Cocoa: Conversation-based service composition in pervasive computing environments with qos support," *Journal of Systems and Software*, vol. 80, no. 12, December 2007.

[42] N. Ibrahim and F. Le Mouel, "A survey on service composition middleware in pervasive environments," *Int. Journal of Computer Sciences Issues*, vol. 1, August 2009.

[43] J. Zhou, J. Riekki, and J. Sun, "Pervasive service computing toward accommodating service coordination and collaboration," in *4th Int. Conf. on Frontier of Computer Science and Technology*, Shanghai, China, December 2009.

[44] E. Avilés-López and J. A. García-Macías, "Tinysoa: a service-oriented architecture for wireless sensor networks," *Service Oriented Computing and Applications*, vol. 3, no. 2, pp. 99–108, 2009.

[45] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, "An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks," in *33rd IEEE Conference on Local Computer Networks, 2008. LCN 2008.*, October 2008, pp. 740–747.

[46] F. C. Delicato, F. P. P. F. Pires, L. Pirmez, and L. F. Carmo, "A flexible web service based architecture for wireless sensor networks," in *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2003, p. 730.

[47] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, and J. Sztipanovits, "Oasis: A programming framework for service-oriented sensor networks," in *Communication Systems Software and Middleware, 2007. COMSWARE 2007. 2nd International Conference on*, Jan. 2007, pp. 1–8.

[48] D. I. Tapia, J. A. Fraile, S. Rodríguez, J. F. de Paz, and J. Bajo, "Wireless sensor networks in home care," in *IWANN '09: Proceedings of the 10th International Work-Conference on Artificial Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 1106–1112.

[49] L. Gurgen, C. Roncancio, C. Labbe, A. Bottaro, and V. Olive, "Sstreamware: a service oriented middleware for heterogeneous sensor data management," in *Proc. of the 5th international conference on Pervasive services*. Sorrento, Italy: ACM, 2008.

[50] B. Srivastava and J. Koehler, "Web service composition — current solutions and open problems," in *ICAPS 2003 Proc.*, 2003.

[51] A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma, "METEOR-S web service annotation framework," in *WWW 2004 Proc.*, 2004.

[52] A. Lazovik, M. Aiello, and M. Papazoglou, "Planning and Monitoring the Execution of Web Service Requests," *Journal on Digital Libraries*, pp. 235–246, 2006.

[53] A. Marconi, M. Pistore, and P. Traverso, "Automated Composition of Web Services: the ASTRO Approach," in *IEEE Data Eng. Bull*, vol. 31, no. 3, 2008, pp. 23–26.

[54] L. A. G. da Costa, P. F. Pires, and M. Mattoso, "Automatic Composition of Web Services with Contingency Plans," in *ICWS 2004 Workshop Proc.*, 2004, san Diego.

[55] V. Agarwal, K. Dasgupta, N. Karnik, A. Kumar, A. Kundu, S. Mittal, and B. Srivastava, "A service creation environment based on end to end composition of web services," in *WWW 2005 Proc.*, 2005, pp. 128–137.

[56] P. Senkul and I. H. Toroslu, "An architecture for workflow scheduling under resource allocation constraints," *Inf. Syst.*, vol. 30, no. 5, pp. 399–422, 2005.

[57] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Trans. Web*, vol. 1, no. 1, pp. 1–26, 2007.

[58] Z. Maamar, Q. Z. Sheng, and B. Benatallah, "Interleaving web services composition and execution using software agents and delegation," in *WSABE 2003*, 2003, melbourne.

[59] "Web service challenge," http://www.wschallenge.org/.

[60] Z. Huang, W. Jiang, S. Hu, and Z. Liu, "Effective pruning algorithm for qos-aware service composition," in *CEC*, 2009, pp. 519–522.

[61] Y. Yan, B. Xu, Z. Gu, and S. Luo, "A qos-driven approach for semantic service composition," in *CEC*, 2009, pp. 523–526.
[62] S.-C. Oh, J.-Y. Lee, S.-H. Cheong, S.-M. Lim, M.-W. Kim, S.-S. Lee, J.-B. Park, S.-D. Noh, and M. M. Sohn, "Wspr*: Web-service planner augmented with a* algorithm," in *CEC*, 2009, pp. 515–518.
[63] D. B. Claro, P. Albers, and J. K. Hao, "Selecting Web Services for Optimal Composition," in *ICWS 2005 Workshop Proc.*, 2005, orlando.
[64] M. C. Jaeger, G. Muhl, and S. Golze, "QoS-aware composition of web services: An evaluation of selection algorithms," in *COOPIS 2005 Proc.*, 2005, cyprus.
[65] P. A. Bonatti and P. Festa, "On optimal service selection," in *WWW 2005 Proc.*, 2005.
[66] C. Wan, C. Ullrich, L. Chen, R. Huang, J. Luo, and Z. Shi, "On solving qos-aware service selection problem with service composition," in *GCC '08 Proc.*, 2008.
[67] Q. Liang, X. Wu, and H. C. Lau, "Optimizing Service Systems Based on Application-Level QoS," *IEEE Transactions on Services Computing*, vol. 2, pp. 108–121, 2009.
[68] C. Ghezzi, A. Motta, V. P. L. Manna, and G. Tamburrelli, "Qos driven dynamic binding in-the-many," in *QoSA '10: Proceedings of the 6th International Conference on the Quality of Software Architectures*, Prague, Czech Republic, 2010.

Chiara Sandionigi received the bachelor and master degrees in Computer Engineering from Politecnico di Milano, where she is presently a PhD student at the Dipartimento di Elettronica e Informazione. Her PhD activity is supported by the European Space Agency. Her research interests are related to the design of embedded systems, with focus on reconfigurability and reliability-aware properties, and on the optimization of service based systems.

Danilo Ardagna received the Ph.D. degree in computer engineering in 2004 from Politecnico di Milano, from which he graduated in December 2000. Now he is an Assistant Professor at the Dipartimento di Elettronica e Informazione, at Politecnico di Milano. His work focuses on the design, prototype and evaluation of optimization algorithms for resource management and planning of Service Oriented and autonomic computing systems.

Gianpaolo Cugola received his Dr.Eng. degree in Electronic Engineering from Politecnico di Milano. In 1998 he received the Prize for Engineering and Technology from the Dimitri N. Chorafas Foundation for his Ph.D. thesis on Software Development Environments. He is currently Associate Professor at Politecnico di Milano where he teaches several courses in the area of Computer Science. His research interests are in the area of Software Engineering and Distributed Systems. In particular, his current research focuses on middleware technology for largely distributed and highly reconfigurable distributed applications with a special attention to the issue of Content Based Routing as the basic mechanism to develop advanced middleware services like publish/subscribe and data sharing.

Carlo Ghezzi is a Professor and Chair of Software Engineering at Politecnico di Milano. He is an ACM Fellow, an IEEE Fellow, and a member of the Italian Academy of Sciences. He was awarded the ACM SIGSOFT Distinguished Service Award. He has been General Chair and a Program Co-Chair of the IEEE International Conference on Software Engineering, Program Chair of the European Software Engineering Conference, General Co-Chair of the International Conference on Service Oriented Computing. He has been the Editor in Cheif of the ACM Transaction on Software Engineering and Methodology and currently is an Associate Editor of IEEE Transactions on Software Engineering, Communications of the ACM, Science of Computer Programming, Service Oriented Computing and Applications, and Computing. His research has been focusing on different facets of software engineering and programming languages. Currently, he is active in the area of design mnethodologies for evolvable and distributed software architectures for ubiquitous and pervasive computer applications.