Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2011, Article ID 649563, 11 pages
doi:10.1155/2011/649563

Research Article

Dynamic Context-Aware and Limited Resources-Aware Service Adaptation for Pervasive Computing

Moeiz Miraoui,[1] Chakib Tadj,[1] Jaouhar Fattahi,[2] and Chokri Ben Amar[3]

[1] *LATIS Laboratory, École de Technologie Supérieure, Université du Québec, 1100, Rue Notre-Dame Ouest, Montréal, QC, Canada H3C 1K3*
[2] *LSFM Laboratory, Université Laval, 2325, Rue de l'Université, Québec city, QC, Canada G1V 0A6*
[3] *REGIM Laboratory, École Nationale d'Ingénieurs de Sfax, Université de Sfax, Route de Soukra, B.P. W, 3038 Sfax, Tunisia*

Correspondence should be addressed to Moeiz Miraoui, moeizmiraoui@gmail.com

Received 25 August 2011; Accepted 15 November 2011

Academic Editor: Letha Hughes Etzkorn

A pervasive computing system (PCS) requires that devices be context aware in order to provide proactively adapted services according to the current context. Because of the highly dynamic environment of a PCS, the service adaptation task must be performed during device operation. Most of the proposed approaches do not deal with the problem in depth, because they are either not really context aware or the problem itself is not thought to be dynamic. Devices in a PCS are generally hand-held, that is, they have limited resources, and so, in the effort to make them more reliable, the service adaptation must take into account this constraint. In this paper, we propose a dynamic service adaptation approach for a device operating in a PCS that is both context aware and limited resources aware. The approach is then modeled using colored Petri Nets and simulated using the CPN Tools, an important step toward its validation.

1. Introduction

The main goal of a PCS is to provide adapted services proactively (i.e., without explicit intervention of the user) to both users and applications, according to the current context. By "provide adapted services," we mean that the services must be delivered according to the current context, which requires a good understanding and use of that context. One important characteristic of a PCS environment is highly dynamic change, caused by the mobility of users and devices. In order to support the user in his everyday tasks, devices must become more autonomic, requiring minimum or no human intervention. Several approaches have been proposed for service adaptation, some of them using the classic strategy of specifying a set of rules for each service and associating them with every possible context configuration. The disadvantage of such approaches is that the rules are set out before putting the system into operation (static, rather than dynamic application). In addition, the developer has to predict all possible context configurations, which may not be evident at the time. Other approaches are based on a weak specification of context elements, and this has a great impact on the adaptation task. Moreover, the existing service adaptation methods are not considered to be context-aware, and context awareness is a key feature of a PCS. Even existing context-aware service adaptation approaches are superficial and do not deal with the issue in depth, because they are either specific to a particular domain or based on an inappropriate definition of context. Moreover, related work does not consider the very limited resources of a device operating in a PCS (which is, in general, hand-held), like battery charge, memory, processing capacity, and so forth, whereas service adaptation in a PCS must take into account these constraints. With the aim of overcoming the problems and limitations of these other methods, we propose a dynamic approach for service adaptation in a PCS, which is both context aware, based on our definition of context in a previous work [1], and limited resources-aware. The adaptation system is modeled using the colored Petri Net formalism [2] and simulated using the CPN Tools [3] based on typical scenarios of a device operating in a PCS.

The rest of this paper is organized as follows: Section 2 presents the related work in service adaptation. Section 3 presents the principles of context and context awareness in a PCS. Section 4 presents our approach of dynamic context-aware and limited resources-aware service adaptation. Section 5 describes the modeling and simulation of the proposed approach. Section 6 presents our conclusions and perspectives on our future work.

2. Related Work

In computing, there are four main kinds of adaptation: (a) content adaptation, (b) behavior adaptation (services), (c) presentation (or interface) adaptation, and (d) software component adaptation [4–11]. We will restrict our adaptation approach to context-aware service adaptation. The need for service adaptation has long been recognized, and both manual and automatic approaches to service adaptation have been proposed. Autili et al. [12] proposed a conceptual model for adaptable context-aware service in their work on the PLASTIC project. The model is based on a two-layer approach combining services and components. However, they do not give a precise definition of context according to which the adaptation is to be performed. Their model looks rather like a software component adaptation than a service adaptation. Kirsch-Pinheiro et al. [13] have proposed an interesting context-aware service selection method using graph matching. This method consists of selecting among the available compatible services, the most appropriate using similarity measures, considering the current context, and taking into account the incompleteness of context information. Their approach is based on Dey's [14] definition of context, which is fairly specific to the human-computer interaction domain. In addition, dynamic adaptation consists of executing a graph-based algorithm each time the adaptation is needed, which exhausts the resources of the devices. Yang et al. [15] worked on the Pervasive Service Platform (PSP), which is the platform that adds pervasiveness to all the services deployed in the Daidalos network (a large European research project that is developing a PCS for a mobile environment). Such an environment should help in the selection of the most appropriate services for any particular user, and employment of the most relevant devices and supporting networks. The architecture of the PSP comprises six main components: context manager, rule manager, event manager, personalization, pervasive service manager, and security and privacy manager. They defined context as the set of information that describes an identity's preferences, profiles, and current situation. The approach is based on the use of rules and policies for making the composition decisions. It is concerned with how personalization is involved in service composition and decomposition, and how personalization adapts services to user context and requirements in this dynamic process. Although the proposed approach is very interesting and highly promising, particularly in the dynamic selection and composition of the required service, it is not based on a clear and adequate definition of context (not really context-aware) and does not take into account the limited resources of devices in a PCS. Cao et al. [16] proposed a context-aware service adaptation method based on designing a policy selection mechanism using fuzzy logic for formulating the service and introducing some fitness functions in order to select the policy with the highest fitness level. In their approach, the authors did not specify the elements of the context according to which the adaptation is used. Moreover, the method requires a great deal of processing. Choi [17] proposed, in their work involving the adoption of the workflow model into ubiquitous computing environments for context-aware and autonomous services, a context-aware workflow system, which can apply changes in the user's service demand or situation information to an ongoing workflow without a break in its operation. The proposed system represents contexts described in a workflow as an RDF-based DI (Document Instance) tree. When a change in a user's situation information happens, the system can dynamically reconstruct a workflow by modifying only the subtrees affected by the change. This means that the system does not obstruct the flow of an earlier ongoing context-aware service. The service adaptation approach seems quite? smart and can be easily applied, according to the user's situation (context); however, it is not based on a clear and concise definition of context (general definition) and also presupposes that devices (in particular, hand-held) can provide a service at any time without considering their limited resources. Houssos et al. [18] introduced a conceptual model for context-aware adaptive services based on Dey's [14] definition of context and adding a categorization of context entities in three domains. Their adaptation method is purely static, which contradicts the nature of a PCS and also the definition of context, like Dey's, as previously mentioned. There are other service adaptation approaches, but most of them suffer from the same limitation as the approaches cited in this section. Hirschfeld and Kawamura [19] combined aspect-oriented programming with computational reflection and late binding to adapt services and service platforms when changes actually require that this be done, as late as possible and preferably without disruption of service. They proposed a layered architectural platform with its constituents built on top of one another, which makes it possible to both implement their basic service logic and adapt this service logic to additional requirements and unforeseen circumstances if necessary. The adaptation is a dynamic one, but context awareness is not really addressed and it is focused rather on content adaptation.

In our previous work [20], we proposed a dynamic context-aware service adaptation approach in a PCS, however the proposed approach did not take into account the limited resources of hand-held devices in a PCS. An interesting work is made by Hamadi and Benatallah [21]consisting of a Petri net-based algebra for composing and modeling web services, which is helpful for verification and properties and the detection of inconsistencies both within and between services. They proposed a web services modeling tool rather than a dynamic service adaptation approach. Hansen et al. [22] introduced Flamenco/CPN tool (a part of the Hydra middleware [23]), which addresses middleware and application challenges in self-managed PCS. The approach combined high-level Petri Nets with the capability of distributed communication among nets. The tool is built in correspondence with the Kramer and Magee [24] reference model for self-managed systems and with event-based communication using the publish/subscribe paradigm. The autonomic system is modeled and implemented as communicating, high-level Petri Nets.

Both [21, 22] did not deal with neither context awareness nor limited resources of hand-held devices, which are key concepts for PCS.

3. Context Awareness

A PCS is composed of computer devices communicating and collaborating to provide proactively adapted services to both users and applications according to the current context. This reveals the importance of the concept of context, which requires a good understanding and a precise establishment of its components. Context is defined as the information needed to interpret something, but this definition is very general and does not provide a good understanding. Moreover, it is not useful for computing [17]. Many researchers have proposed definitions of context, some of which were based on listing contextual information (localization, nearby people, time, date, etc.) like those proposed by [18, 19, 25]. Others were based on providing more formal definitions in order to abstract the term, like that proposed by Dey [14]. Most of these definitions were specific to a particular domain, such as human-computer interaction and localization systems. In our previous work [1], we have made a survey of existing definitions of context and proposed a service-oriented definition of context for a PCS as follows: "Any information that triggers a service or changes the quality (form or mode) of a service if its value changes." This definition is sufficiently abstract and helps to limit the set of contextual information. In the same way and based on the concept of service, we have provided the following definition of context awareness: "A system is said to be context aware if it can automatically change the form of its services or provide a service in response to the change in the value of information or a set of information that characterizes those services." Unlike other definitions of context awareness [17, 26–28], our definition explains awareness more generically, as a reaction of the system to modifications to information values in terms of triggering a service or changing its form independently of the application. We have also classified contextual information in two categories: (a) triggering information, a change in value of which causes automatic release of a service and (b) form-changing information, a change in value of which causes the change in a service's form. We believe that this categorization is more expressive: it is simple, because it contains just two classes, and it is complete, because it covers all aspects of context, which facilitates the task of adaptation.

4. Dynamic Service Adaptation

Dynamic service adaptation consists of providing both manually and automatically triggered services according to the current context and by taking into account the limited

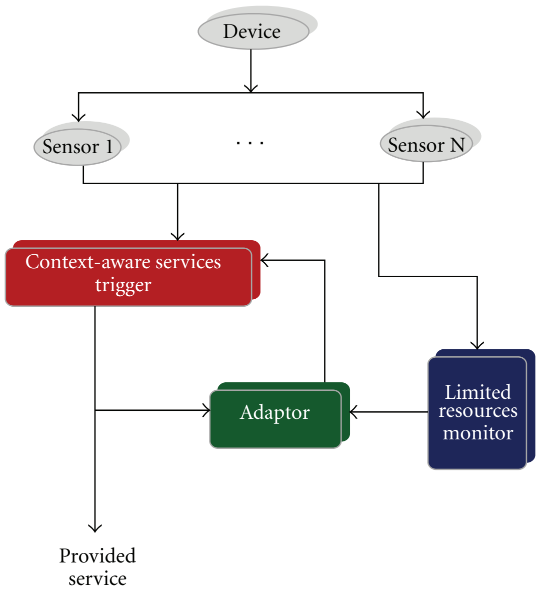

Figure 1: Global architecture of the adaptation system.

resources of the device. This adaptation must be applied in a transparent way to the user and during operation of the system (dynamic). The main idea is to provide services without exhausting the limited resources of the device. As soon as a service has been triggered (manually or automatically) and may exhaust a limited resource, the system seeks another configuration of the services provided, which prevents the exhaustion of the limited resource, in turn increasing the device's operational lifetime. The proposed adaptation system is made up of three basic components: a context-aware services trigger, a limited resources monitor, and the adaptor, as shown in Figure 1.

In describing our approach, we will detail each component of the architecture of the adaptation system and the communication flow between them.

4.1. Context-Aware Service Trigger. One of most important components of the adaptation system architecture is the context-aware services trigger (CST). This component dynamically collects contextual information using the device's sensors and is based on our previous definition of context in a PCS. The developer's task consists of determining, for each device, the set of services it can provide, and, for each service, the set of forms (or modes) in which the service can be provided. For example, a laptop computer provides an internet connection in two forms: wired and wireless.

There are two categories of contextual information: (a) triggering information, a change in value of which causes the automatic release of a service and (b) form-changing information, a change in value of which causes a change in a service's form (Figure 2 and Table 1).

There are two types of service: services triggered manually by the user and services triggered automatically according to the current context. To differentiate between these two service types, the CST uses a control bit, denoted K

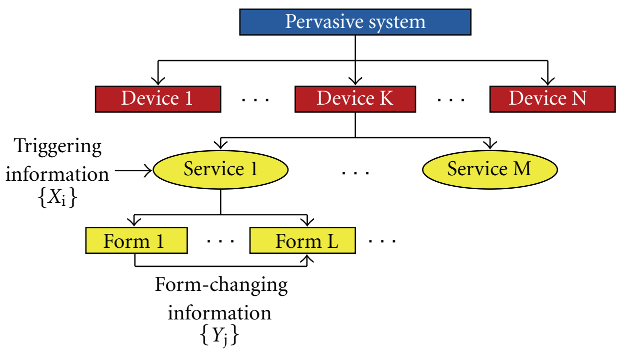

FIGURE 2: Contextual information classes: a change in the values of $\{x_i\}$ will trigger service 1. A change in the values of $\{y_j\}$ will change the form of the service 1.

TABLE 1: Contextual information classes

Services	Triggering information	Forms	Form-changing information
Service i	$\{X_i\}$	Form 1 Form 2	$\{Y_j\}$

($K = 0$ for the automatically triggered service and $K = 1$ for the manually triggered service). The CST uses another control bit, denoted S, to memorize the current state of each service ($S = 0$ for the currently inactive service and $S = 1$ for the currently active service). The CST also uses another control bit, denoted A, for each form of service to indicate whether or not this form is currently active ($A = 1$ means that the form is currently active and $A = 0$ means that it is currently inactive). This bit is automatically marked 0 for all the forms of a service if this latter is currently inactive. Moreover, the CST contains, for each form of a service, the set of the device's limited resources affected (consumed) and indicates which limited resources have been consumed if a service is provided in a particular form. Figure 3 shows the structure of the CST.

A service will be automatically triggered when some context information of this service takes some values, in other words when the service triggering logical expression is evaluated to be true. The triggering logical expression is a logical combination of a subset of the service context element values (e.g., if $((a = x)$ AND $(b = y))$, then the service will be triggered knowing that a and b belong to the contextual information set service). When triggered, the service is provided in its default form or in another form if the form-changing logical expression (which has a similar structure to the service triggering logical expression) of that form is evaluated to be true.

The structure of the CST can be formulated according to the following notation.

(i) Set of contextual information

$C = \{C_1, C_2, \ldots, C_n\}$;

C_i can be the location of a person, the speed of an Internet connection, date, hour, and so forth.

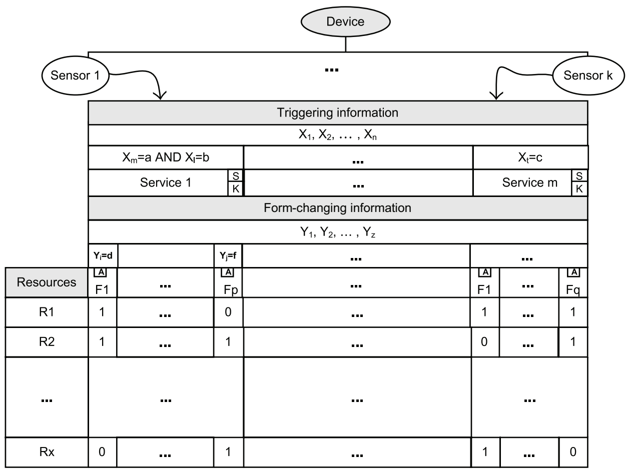

FIGURE 3: Structure of the CST.

(ii) Set of limited resources of a device

$R = \{R_1, R_2, \ldots, R_p\}$;

R_q, $q = 1, \ldots, p$, can be the charge level of a battery, Internet connection, capacity of an integrated memory, and so forth.

(iii) Set of values of contextual information

$V = \{(C_i, D_i)\}\ i = 1, \ldots, n$;

D_i is a value associated with a context element.

Example 1 (localization, university). (i) Set of utilization ratios (%) of the limited resources $T = \{(R_q, P_q)\}\ q = 1, \ldots, p$; P_i is a percentage of use of a limited resource.

Example 2 (internet connection, 80). (i) Set of services provided by a device $S = \{S_1, S_2, \ldots, S_k\}$.

(ii) Set of service forms i; $SF_i = \{F_{i1}, F_{i2}, \ldots, F_{ixi}\}$; x_i is the set of forms of the service i; F_{ij}: the form j of the service i, $j = 1, \ldots, x_i$ and $i = 1, \ldots, k$. (iii) Each service uses a set of contextual information. $SC_i = \{C_y\}$, $y = 1, \ldots, m$, $m \leq n$.

(iv) Each service S_i uses a set of limited resources. $SR_i = \{R_z\}$, $z = 1, \ldots, v$, $v \leq p$.

(v) Each service Si has a logical triggering expression $SE_i, i = 1, \ldots, k$. SE_i is a logical combination (AND, OR, NOT) of $V_i = \{(C_i, D_i)\}$ where $C_i \in SC_i$.

(vi) Each form j of a service i has a form-changing logical expression FE_{ij}, $j = 1, \ldots, x_i$ and $i = 1, \ldots, k$. FE_{ij} is a logical combination (AND, OR, NOT) of $V_i = \{(C_i, D_i)\}$, where $C_i \in SC_i$.

(vii) Each form j of a service i uses a set of limited resources $FR_{ij} \sqsubseteq SR_i$, $j = 1, \ldots, x_i$ and $i = 1, \ldots, k$.

(viii) Each form j of a service i has a state according to

$F_{ij}\ j = 1, \ldots, x_i$ and $i = 1, \ldots, k$,

$$\text{State}\begin{cases} 0 & \text{if the form is inactive,} \\ 1 & \text{if the form is active.} \end{cases} \tag{1}$$

(ix) Each service S_i, $i = 1, \ldots, k$, has a state according to (2)

$$\text{State}\begin{cases} 1 & \text{if the active service is triggered automatically,} \\ 0 & \text{if the service is inactive,} \\ 2 & \text{if the active service is triggered manually.} \end{cases} \tag{2}$$

If a service is active, then the state of one and only one of its forms will be marked 1 or 2.

(x) A service is represented by $< S_i, \{F_{ij}, FE_{ij}, FR_{ij}, \text{state}\}, SC_i, SE_i >$ where $j = 1, \ldots, x_i$ and $i = 1, \ldots, k$.

The default state of the CST is asleep, in order to reduce resource consumption. Once a context element (triggering information or form changing information) changes its value, the CST wakes up and reacts by triggering a service or by changing the form of a service.

4.2. Limited Resources Monitor. A PCS is composed of a set of computer devices where each device provides a set of services. Each service has several forms, and one of them is the default form of that service. The adaptation task consists of automatically triggering a service or changing its form according to the current context and by taking in account the available resources. Most devices in a PCS are hand-held and mobile (e.g., a cellular phone, a laptop computer, etc.). We can classify the resources of such devices into two categories: unlimited and limited. We can define the unlimited resources as a category which is less likely to have an availability problem once used (e.g., the hard disk of a desktop machine). By contrast, limited resources can be defined as those that should be employed carefully because their availability decreases rapidly (e.g., charge level of a cellular phone battery, the bandwidth of an Internet connection, etc.).

It is the adaptation system developer that considers whether a resource is limited or unlimited for each device and according to the application domain. Unlimited resource allocation is not considered a problem compared to the allocation of limited resources. Consequently, we will restrict ourselves to addressing limited resources here. We will suppose that each device has its own sensors, which provide a measure of the availability of its limited resources (e.g., the battery charge of a cellular phone). The adaptation task consists in large part of providing a service in a form which affects (consumes) as little as possible the limited resources of a device to avoid exhausting them. That is why we include a limited resources monitor (LRM) as the second component in our adaptation system architecture, which continuously provides the availability ratio of each limited resource for each device. For example, a cellular phone LRM can be represented as shown in Table 2. In this example, we consider the battery charge and the integrated memory as limited resources.

Table 2: Example of an LRM

Device	Limited resources	Availability percentage
Cellular phone	Battery charge	70%
	Integrated memory	50%
	...	...

4.3. Adaptor. The adaptor is the third component of the architecture. Its principal task is to dynamically adapt the services provided according to the current context and the availability ratio of the limited resources. The adaptor has access to the contents of the CST to identify the set of currently active services, their forms, and their types (automatically triggered service or manually triggered service). It is also in communication with the LRM, using the model subscription/notification, in order to be notified each time the available quantity of a limited resource becomes null (the resource has been completely exhausted). The default state of the adaptor is asleep, in order to reduce the consumption of resources. Once notified by the LRM, it becomes active. Manually triggered services have priority over those that are started automatically, because they are triggered by the user, which means that they constitute an immediate need on the part of the user. The main aim of context awareness in a PCS is to satisfy the user's needs initially. The system must put all the necessary limited resources into providing these manually services triggered: this is the principal task of the adaptor. We can distinguish between two operational situations, depending on the currently active service types:

(i) Situation 1: all the currently active services are triggered automatically (according to the current context);

(ii) Situation 2: one (or several) of the currently active services is (are) triggered manually by the user.

Strategy 1. All the currently active services are triggered automatically, according to the current context. In this case, all the services have the same priority degree. The adaptor executes the following steps:

For each resource entirely-used R_i, we have the following steps.

Step 1. Seek the forms of currently active services that employ the resource R_i by accessing the contents of the CST and by using the control bits $K, S,$ and A.

Step 2. Limit the set of currently active services that can be provided in another form which does not employ the resource R_i. If that is not possible, randomly choose an active service and deactivate it, and then go to Step 3.

Step 3. Randomly choose one of these services, and randomly choose one of its other forms which does not use the resource R_i Send a command to the CST in order to update its contents according to the actions of the adaptor.

Once the adaptor has been notified by the LRM by a message indicating which limited resource has become entirely used up (availability becomes null), it wakes up and reacts according to the following two strategies and the current situation.

Strategy 2. In this case, there are one or more currently active services triggered manually by the user. The system must then provide all the necessary limited resources to satisfy the needs of the user. The adaptor executes the following steps:

For each entirely-used resource R_i, we have the following steps.

Step 1. Seek the forms of currently active services that use the resource R_i by accessing the contents of the CST and by using the control bits $K, S,$ and A.

Step 2. Limit the set of currently active services (only those triggered automatically) to those that can be provided in another form which do not employ the resource R_i. If this is not possible, then seek the currently active forms of services (triggered manually) that employ the resource R_i.

Step 3. Choose the services that can be provided in another form which does not use the resource R_i, propose a service to the user, the form of which should be changed by the system change, and go to Step 4.

Step 4. Randomly choose one of these services, and randomly choose one of its other forms which does not use the resource R_i. Send a command to the CST in order to update its contents according to the actions of the adaptor.

The adaptor operates as a regulator in a closed loop for the CST, where LRM information is used to adjust its output.

5. Modeling and Simulation

In this part, we model and simulate the two possible situations of the limited resources-aware adaptation approach, which is sensitive to those resources. The steps of each strategy are modeled using the colored Petri Net formalism and simulated using the CPN Tools.

5.1. Strategy 1: All Currently Active Services Are Triggered Automatically

5.1.1. Modeling. Figure 4 shows the simplified hierarchical model of the adaptation system, which contains three steps. Figure 5, Figure 6, and Figure 7 show the detailed models of Steps 1, 2, and 3, respectively. The set of declarations used for the construction of the model of situation 1 is given by Algorithm 1 as follows.

5.1.2. Scenario Simulation and Results. To validate our model of situation 1, we employed a typical scenario, the details of which are as follows.

A computer device has six limited resources, one of them having a 100% utilization ratio (resource “b”). We represent this information by the couple: (resource name, utilization ratio (%)). Let us consider the following example: (“a”,50), (“b”,100), (“c”,10), (“d”,35), (“e”,80), and (“f”,45).

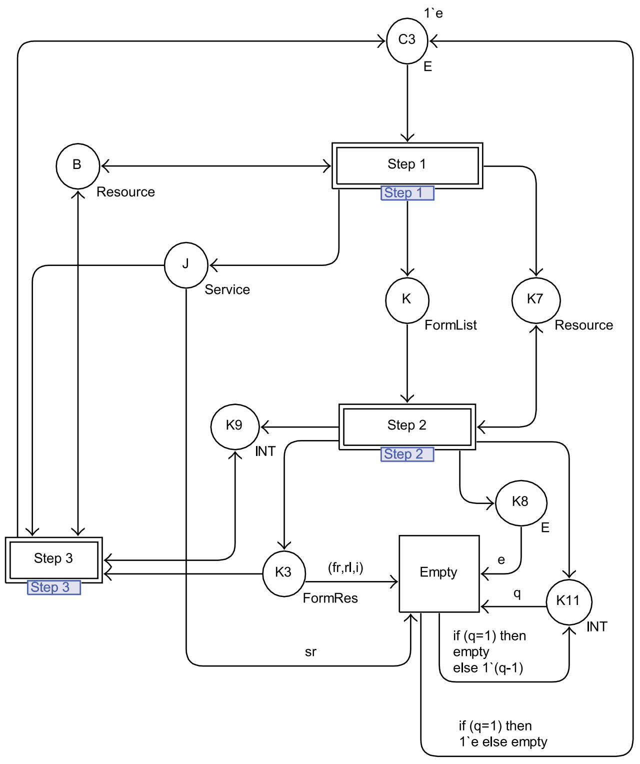

FIGURE 4: Hierarchical model of the adaptation system of situation 1.

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
val NS=5;
colset Resource=STRING;
colset E=with e;
colset ResList=list Resource;
colset Form=STRING;
colset ResAmt=product Resource∗INT;
colset Service=STRING;
colset SerStat=product Service∗ResList∗INT;
colset FormRes=product Form∗ResList∗INT;
colset FormList=list FormRes;
colset SerFormList=product Service∗FormList;
colset SerForm=product Service∗FormRes;
var r: Resource;
var q,p,i,n:INT;
var s,sr:Service;
var y:ResList;
var fl:FormList;
var fr:Form;
var rl:ResList;
```

ALGORITHM 1: Set of declarations used for constructing the model of situation 1.

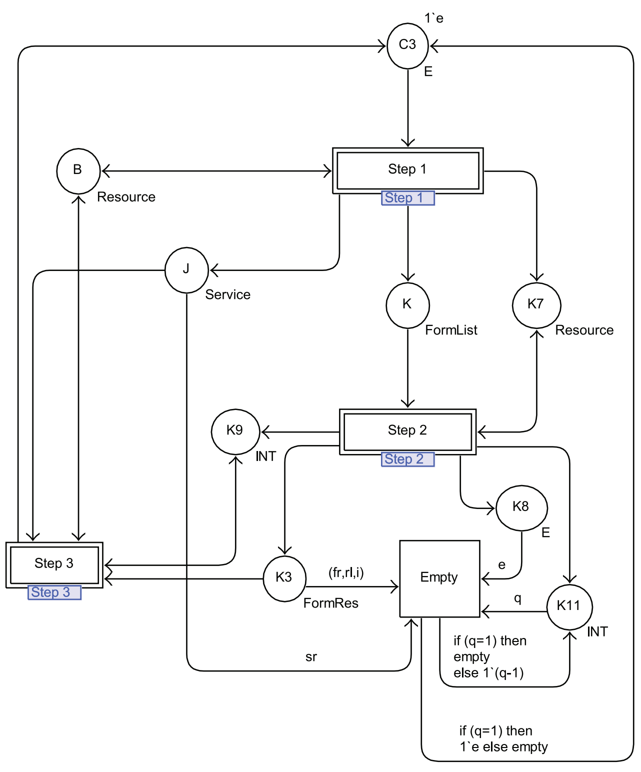

Figure 5: Model of situation 1 (Step 1).

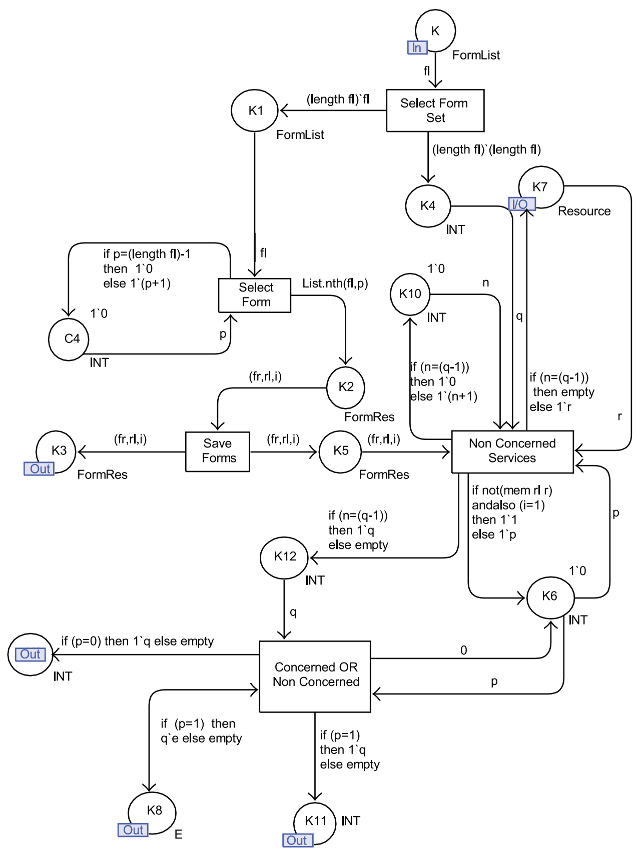

Figure 7: Model of situation 1 (Step 3).

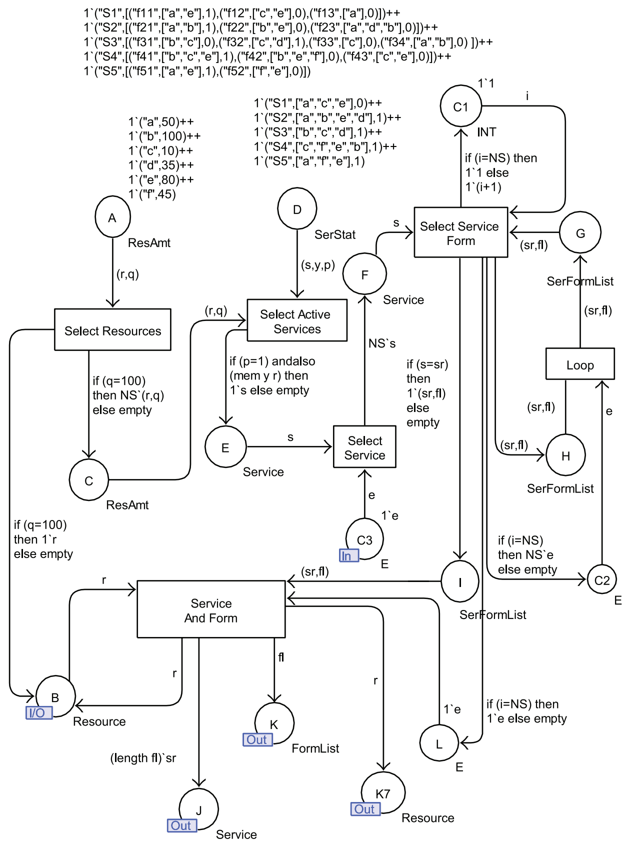

Figure 6: Model of situation 1 (Step 2).

Five services provided by the computer device, four of which are active (triggered automatically according to the current context). Each service uses a set of limited resources. We represent this information by the triplet:

(Service name, [list of limited resources used], state), where

$$\text{State}\begin{cases}0 & \text{if the service inactive,}\\ 1 & \text{if the service is active.}\end{cases} \tag{3}$$

Let us consider the following situation:

("S1",["a", "c", "e"],0),

("S2",["a", "b", "e", "d"],1),

("S3",["b", "c", "d"],1),

("S4",["c", "f", "e", "b"],1),

("S5",["a", "f", "e"],1).

Each service has a set of forms under which it is provided, and each form uses a set of limited resources. If a service is currently active, then the state of one of its forms is marked according to (1). We define the triplet in this manner:

(Service name, [[(form name, [list of used resources], state)]]).

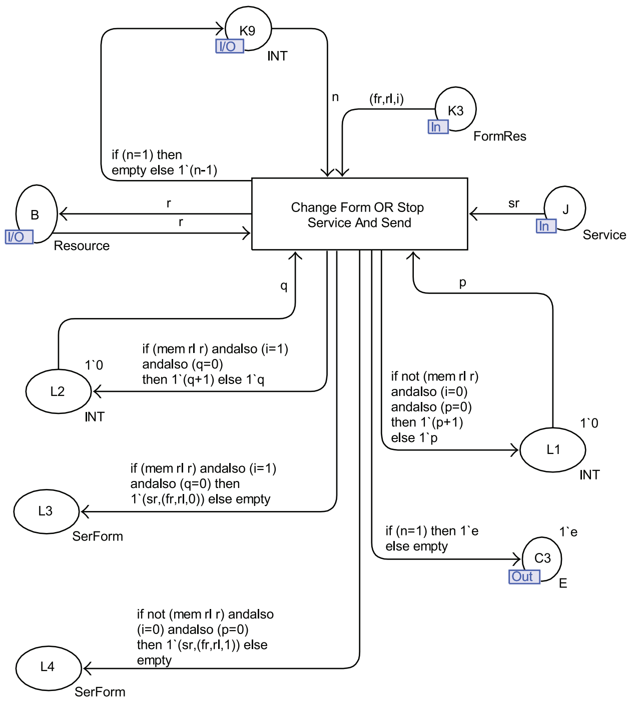

Figure 8: Simplified hierarchical model of the adaptation system, situation 2.

Let us consider the following situation:

("S1",[("F11",["a", "e"],0), ("F12",["c", "e"],0), ("F13",["a"],0)]),

("S2",[("F21",["a", "b"],1), ("F22",["b", "e"],0), ("F23",["a", "d", "b"],0)]),

("S3",[("F31",["b", "c"],0), ("F32",["c", "d"],1), ("F33",["c"],0), ("F34",["a", "b"],0)]),

("S4",[("F41",["b", "c", "e"],1), ("F42",["b", "e", "f"],0), ("F43",["c", "e"],0)]),

("S5",[("F51",["a", "e"],1), ("F52",["f", "e"],0)]).

In this scenario, the two services (S2 and S4) have an active form that uses the exhausted resource (b). After running the simulation, we obtained a new active form (F43: form 3 of service 4), which does not use the exhausted resource. The service S2 cannot be provided in another form, since all its forms use the exhausted resource.

We made several modifications to the scenario parameters (number of services, number of limited resources exhausted, list of limited resources used by each service, number of forms of each service, number of active services, etc.), and each time we obtained the expected results.

5.2. Strategy 2: Currently Active Services Are Triggered Both Manually and Automatically

5.2.1. Modeling. Figure 8 shows the simplified hierarchical model of the adaptation system, which contains four steps. Figure 9, Figure 10, Figure 11, and Figure 12 show the detailed models of Steps 1, 2, 3, and 4, respectively. The set of

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
val NS=5;
colset Resource=STRING;
colset E=with e;
colset ResList=list Resource;
colset Form=STRING;
colset ResAmt=product Resource∗INT;
colset Service=STRING;
colset SerStat=product Service∗ResList∗INT;
colset FormRes=product Form∗ResList∗INT;
colset FormList=list FormRes;
colset SerFormList=product
Service∗FormList;
colset SerForm=product Service∗FormRes;
colset SerNum=product Service∗INT;
colset SerFormRes=product
Service∗FormRes∗Resource;
var r: Resource;
var k,q,p,i,n,m:INT;
var s,sr:Service;
var y:ResList;
var fl:FormList;
var fr:Form;
var rl:ResList;
```

Algorithm 2: Set of declarations used for constructing the model of situation 2.

declarations used for constructing the model of situation 2 is given by Algorithm 2.

5.2.2. Scenario Simulation and Results . To validate our model of situation 2, we employed a typical scenario, the details of which are as follows.

A computer device has six limited resources, with one of them having a 100% utilization ratio (resource "b"). We represent this information by the couplet:

(resource name, utilization ratio (%))

Let us consider the following example:

("a",50), ("b",100), ("c",10), ("d",35), ("e",80), ("f",45)

Five services provided by the computer device, three of which are active (triggered automatically according to the current context) and two of which are active and triggered manually by the user. Each service uses a set of limited resources. We represent this information by the following triplet:

(Service name, [list of limited resources used], state), where the state is marked according to (2).

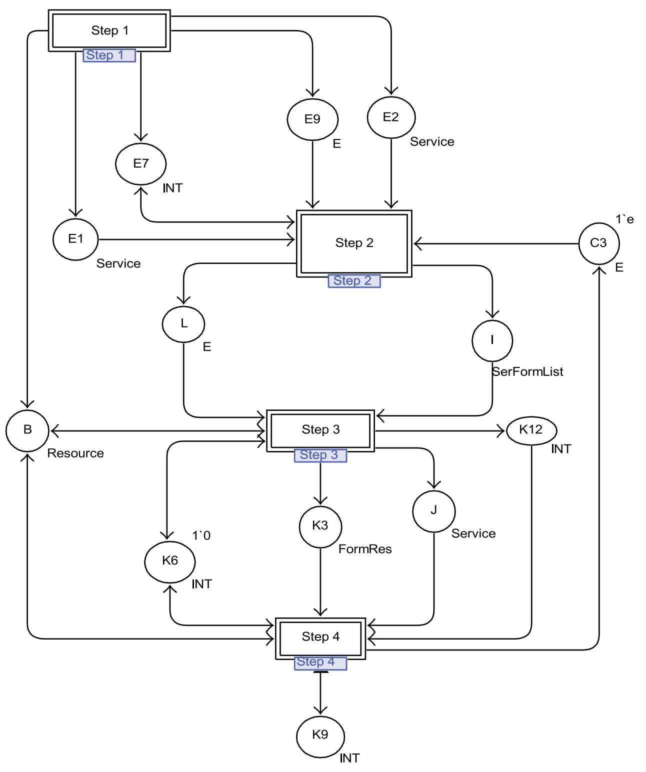

FIGURE 9: Model of situation 2 (Step 1).

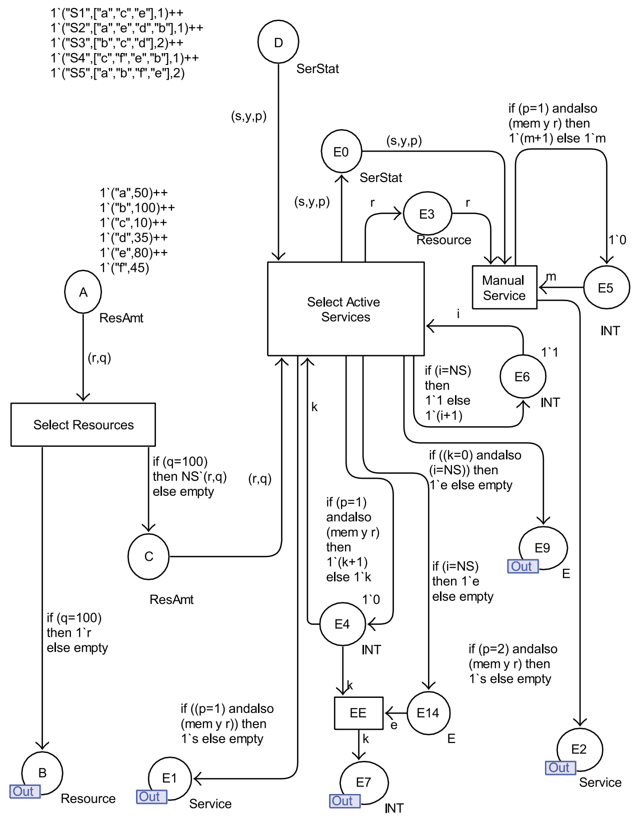
1`("S1",["a","c","e"],1)++
1`("S2",["a","e","d","b"],1)++
1`("S3",["b","c","d"],2)++
1`("S4",["c","f","e","b"],1)++
1`("S5",["a","b","f","e"],2)
D
SerStat
(s,y,p)
E0
(s,y,p)
SerStat
if (p=1) andalso (mem y r) then 1`(m+1) else 1`m
1`("a",50)++
1`("b",100)++
1`("c",10)++
1`("d",35)++
1`("e",80)++
1`("f",45)
(s,y,p)
r
E3
r
Resource
1`0
A
ResAmt
Select Active Services
Manual Service
m
E5
INT
i
1`1
(r,q)
if (i=NS) then 1`1 else 1`(i+1)
E6
INT
Select Resources
k
if (q=100) then NS`(r,q) else empty
(r,q)
if ((k=0) andalso (i=NS)) then 1`e else empty
if (p=1) andalso (mem y r) then 1`(k+1) else 1`k
E9
Out
E
C
if (i=NS) then 1`e else empty
ResAmt
1`0
E4
INT
if (q=100) then 1`r else empty
if (p=2) andalso (mem y r) then 1`s else empty
k
EE
e
E14
E
if ((p=1) andalso (mem y r)) then 1`s else empty
k
E2
Out
Service
B
Out
Resource
E1
Out
Service
E7
Out
INT

FIGURE 10: Model of situation 2 (Step 2).

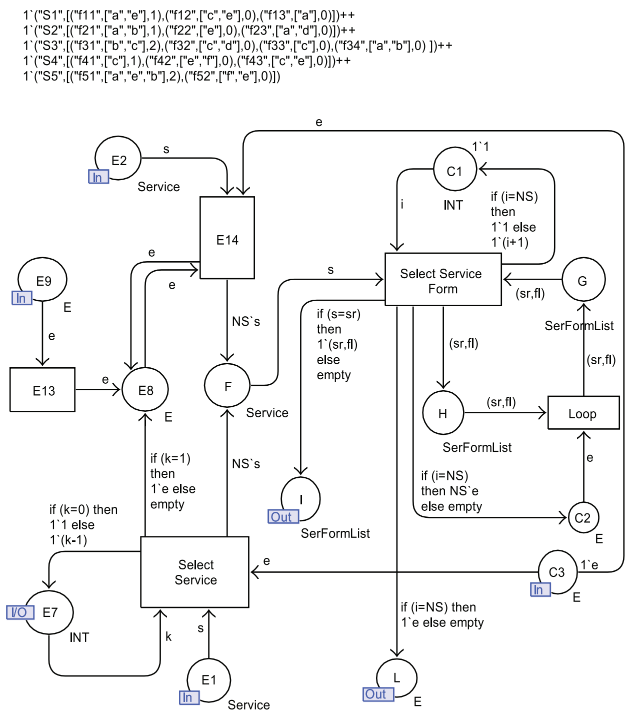
1`("S1",[("f11",["a","e"],1),("f12",["c","e"],0),("f13",["a"],0)])++
1`("S2",[("f21",["a","b"],1),("f22",["e"],0),("f23",["a","d"],0)])++
1`("S3",[("f31",["b","c"],2),("f32",["c","d"],0),("f33",["c"],0),("f34",["a","b"],0)])++
1`("S4",[("f41",["c"],1),("f42",["e","f"],0),("f43",["c","e"],0)])++
1`("S5",[("f51",["a","e","b"],2),("f52",["f","e"],0)])
e
E2
In
s
Service
1`1
C1
i
INT
if (i=NS) then 1`1 else 1`(i+1)
E14
E9
In
E
e
e
e
s
Select Service Form
G
(sr,fl)
SerFormList
NS`s
if (s=sr) then 1`(sr,fl) else empty
(sr,fl)
(sr,fl)
E13
e
E8
E
F
Service
H
(sr,fl)
Loop
SerFormList
e
if (k=1) then 1`e else empty
NS`s
if (i=NS) then NS`e else empty
if (k=0) then 1`1 else 1`(k-1)
I
Out
SerFormList
C2
E
Select Service
e
C3
In
1`e
E
I/O
E7
INT
k
s
if (i=NS) then 1`e else empty
E1
In
Service
L
Out
E

FIGURE 11: Model of situation 2 (Step 3).

Let us consider the following situation:

(“S1”,[“a”, “c”, “e”],1),

(“S2”,[“a”, “b”, “e”, “d”],1),

(“S3”,[“b”, “c”, “d”],2),

(“S4”,[“c”, “f”, “e”, “b”],1),

(“S5”,[“a”, “b”, “f”, “e”],2).

Each service has a set of forms under which it is provided, and each form uses a set of limited resources. If a service is currently active, then the state of one of its forms is marked according to (2). We define the triplet in this manner:

(Service name, [[(form name, [list of used resources], state)]]).

Let us consider the following situation:

(“S1”,[(“F11”,[“a”, “e”],1), (“F12”,[“c”, “e”],0), (“F13”,[“a”],0)]),

(“S2”,[(“F21”,[“a”, “b”],1), (“F22”,[“e”],0), (“F23”,[“a”, “d”],0)]),

(“S3”,[(“F31”,[“b”, “c”],2), (“F32”,[“c”, “d”],0),(“F33”,[“c”],0), (“F34”,[“a”, “b”],0)]),

(“S4”,[(“F41”,[“c”],1), (“F42”,[“e”, “f”],0), (“F43”,[“c”, “e”],0)]),

(“S5”,[(“F51”,[“a”, “e”, “b”],2), (“F52”,[“f”, “e”],0)]).

In this scenario, there are three services (S2, S3, and S5), the active form of which utilizes the exhausted resource (b). The service S2 is triggered automatically according to the current context, while the services S3 and S5 are triggered

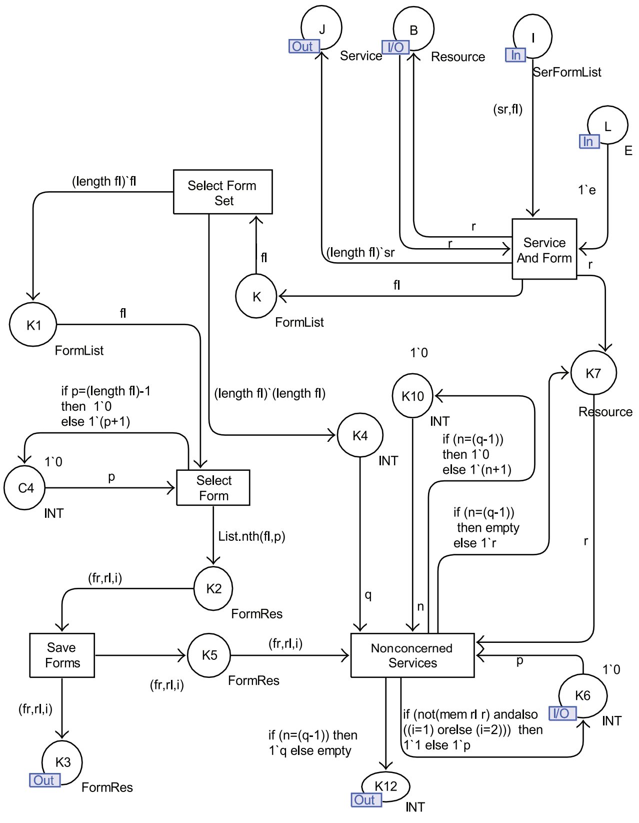

Figure 12: Model of situation 2 (Step 4).

manually by the user. After executing the simulation, we obtained a new active form (F23: form 3 of the service 2), which does not use the exhausted resource. As mentioned above for strategy 2, the system must initially seek a form that does not use the exhausted resource among the automatically triggered services. If it does not find the suitable form among these services, then it disables the active form that uses the exhausted resource of the automatically triggered service. Otherwise, it goes to the services triggered manually to perform the same treatment.

We made several modifications to the scenario parameters (services, exhausted limited resources, list of limited resources used by each service, forms for each service, automatically triggered active services, number of active services triggered manually, etc.), and each time we obtained the expected results that the system must provide.

6. Conclusion

Devices operating in a PCS must provide proactively adapted services to both users and applications. The adaptation task must be performed according to the current context (context aware) and by considering the limited resources in PCS devices, which are generally hand-held. Existing approaches to service adaptation suffer from three main limitations: (a) they are not thought to be really context-aware, (b) they are not dynamic, that is, the adaptation is not applied during operation of the system, and (c) they are not suitable for devices with limited resources. In this paper, we have proposed a dynamic context-aware and limited resource-aware service adaptation system for devices in a PCS. The adaptation system was modeled using the formalism of colored Petri Nets and simulated using the CPN Tools. Future work would involve implementing the adaptation system for one device first, and then implementing it for a set a devices operating in a PCS.

Acknowledgment

The authors acknowledge the funding of the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] M. Miraoui and C. Tadj, “A service oriented definition of context for pervasive computing,” in *Proceedings of the 16th International Conference on Computing*, IEEE Computer Society Press, Mexico City, Mexico, November 2007.

[2] K. Jensen, L. M. Kristensen, and L. Wells, “Coloured petri nets and CPN Tools for modelling and validation of concurrent systems,” *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3-4, pp. 213–254, 2007.

[3] Home page of CPN Tools, http://wiki.daimi.au.dk/cpntools/cpntools.wiki.

[4] M. Aksit and Z. Choukair, “Dynamic, adaptive and reconfigurable systems overview and prospective vision,” in *Proceedings of the International Conference on Distributed Computing Systems Workshops (ICDCSW ’03)*, pp. 84–92, Providence, RI, USA, 2003.

[5] J. Keeney and V. Cahill, “Chisel: a policy-driven, context-aware, dynamic adaptation framework,” in *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks*, pp. 3–13, IEEE, Lake Como, Italy, 2003.

[6] M. T. Segara and F. André, “A framework for dynamic adaptation in wireless environments,” in *Proceedings of the Technology of Object Oriented Languages and Systems (TOOLS 33)*, St. Malo, France, 2000.

[7] D. Narayanan, J. Flinn, and M. Satyanarayanan, “Using history to improve mobile application adaptation,” in *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications*, Monterey, Calif, USA, 2000.

[8] C. Efstratiou, K. Cheverst, N. Davies, and A. Friday, “An architecture for the effective support of adaptive context-aware applications,” in *Proceedings of the 2nd International Conference on Mobile Data Management (MDM ’01)*, pp. 15–26, Hong Kong, 2001.

[9] M. Fayad and M. P. Cline, “Aspects of software adaptability,” *Communications of the ACM*, vol. 39, no. 10, pp. 58–59, 1996.

[10] G. South, A. P. Lenaghan, and R. R. Malyan, “Using reflection for service adaptation in mobile clients (t4),” Tech. Rep., Kingston University, UK, 2000.

[11] M. Yarvis, P. Reiher, and G. J. Popek, “Conductor: a framework for distributed adaptation,” in *Proceedings of the 7th Workshop on Hot Topics in Operating Systems (HotOS-VII)*, pp. 44–49, March 1999.

[12] M. Autili, V. Cortellessa, A. di Marco, and P. Inverardi, “A conceptual model for adaptable context-aware services,” in *Proceedings of the Web Services Modeling and Testing (WS-MaTe ’06)*, Palermo, Sicily, Italy, June 2006.

[13] M. Kirsch-Pinheiro, Y. Vanrompay, and Y. Berbers, "Context-aware service selection using graph matching," in *Proceedings of the 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC '08)*, Dublin, Ireland, November 2008, at ECOWS 2008.

[14] K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, pp. 4–7, 2001.

[15] Y. Yang, F. Mahon, M. H. Williams, and T. Pfeifer, "Context-aware dynamic personalised service re-composition in a pervasive service environment," in *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC '06)*, pp. 724–735, Wuhan, China, September 2006.

[16] J. Cao, N. Xing, A. T. S. Chan, Y. Feng, and B. Jin, "Service adaptation using fuzzy theory in context-aware mobile computing middleware," in *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '05)*, pp. 496–501, Hong Kong, August 2005.

[17] J. Choi, Y. Cho, K. Shin, and J. Choi, "A context-aware workflow system for dynamic service adaptation," in *Proceedings of the International Conference Computational Science and its Applications (ICCSA '07)*, pp. 335–345, 2007.

[18] N. Houssos, S. Pantazis, and A. Alonistioti, "Generic adaptation mechanism for the support of context-aware service provision in 3G networks," in *Proceedings of the 4th IEEE International Conference on Mobile Wireless Communication Networks (MWCN '02)*, Stockholm, Sweden, September 2002.

[19] R. Hirschfeld and K. Kawamura, "Dynamic service adaptation," in *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW '04)*, vol. 2, pp. 290–297, 2004.

[20] M. Miraoui, C. Tadj, and C. B. Amar, "Dynamic context-aware service adaptation in a pervasive computing system," in *Proceedings of the 3rd International Conference on Mobile Ubiquitous Computing, Systems, Services, and Technologies (UBICOMM '09)*, pp. 77–82, October 2009.

[21] R. Hamadi and B. Benatallah, "A petri net-based model for web service composition," in *Proceedings of the 14th Australasian Database Conference*, pp. 191–200, 2003.

[22] K. M. Hansen, W. Zhang, and M. Ingstrup, "Towards self-managed executable petri nets," in *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '08)*, pp. 287–296, October 2008.

[23] K. M. Hansen, W. Zhang, and G. Soares, "Ontology-enabled generation of embedded web services," in *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering*, Redwood City, San Francisco Bay, USA, July 2008.

[24] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Proceedings of the Future of Software Engineering (FoSE '07)*, pp. 259–268, IEEE Computer Society, Washington, DC, USA, 2007.

[25] G. Chen and D. Kotz, "A survey of context-aware mobile computing research," Tech. Rep., Department of Computer Science, Dartmouth College, 2000.

[26] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, no. 5, pp. 22–32, 1994.

[27] P. J. Brown, J. D. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," *IEEE Personal Communications*, vol. 4, no. 5, pp. 58–64, 1997.

[28] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pp. 85–90, December 1994.