Expressing and Interpreting User Intention in Pervasive Service Environments

Pascal Bihler Vasile-Marian Scuturici Lionel Brunie
Laboratoire LIRIS – UMR 5205, INSA de Lyon
7 avenue Jean Capelle, F-69621 Villeurbanne cedex, France
{pascal.bihler, marian.scuturici, lionel.brunie}@insa-lyon.fr

Abstract

The introduction of pervasive computing environments enforce new ways of human-machine-interaction. In a pervasive service environment, the system middleware should take care of capturing the user's expression of an action intention, solving ambiguousness in this expression, and executing the final pervasive action. After recalling the Pervasive Service Action Query Language (PsaQL), a language to formalize the user's intention of composing pervasive services, this paper presents the next steps of intention treatment in a pervasive service environment: A mathematical model is given, which helps to express the algorithms performing translation of the user intention into an executable action. To implement such algorithms, a suitable object-oriented model representing actions is introduced. In the scope of PERSE*, a pervasive service environment developed by our research group, a prototype has been developed and first benchmark results are presented in this paper.*

1 Introduction

Pervasive Computing is going to become everyday reality for a large part of our society. A network of omnipresent, highly embedded computing machines seems reasonable in the upcoming century in a manner that one even does not recognize the presence of these computers anymore. In the way, one perceives (or don't perceives anymore) his/her[1] intelligent environment, the way one interacts with it has to change as well. The shift is clearly defined from teaching the user how to interact with a computer to teach the computer how to interact with a user.

Computing devices will move from reactive actions to proactive ones. In our understanding, "being proactive" means for a computing system "understanding the user's intention", "learning from its action history", and "proposing an action" or "acting". These two first aspects touch the two ways of deriving the action intention, either from an explicit user directive or by comparing the context information with the action history. In a pervasive environment bringing different services on different machines together, it should not be the concern of the service developer to care about the user intention interpretation, but rather he should rely on a well defined service interface and concentrate on doing the service work well.

At the LIRIS laboratory in Lyon, we develop a pervasive service environment platform called PERSE. This paper continues the way of interpreting a user intention in a pervasive service environment and deriving in a formal way a corresponding action using service composition, first published in [1]. The proposed approach is integrated in the PERSE environment and benchmark test have been performed.

The rest of this paper is organized as follows: The challenges and constraints of a pervasive service environment are presented by introducing the main aspects of PERSE in Sect. 2. In the following Sect. 3, the process of handling the user intention in a pervasive service environment is presented. We recall the formal language PsaQL and introduce a formal description of the translation process between the user's and the machine's interpretation, along with corresponding algorithms. This is followed by some benchmarks guidelines and prototype results in Sect. 5, leading to a short overview of related work, a conclusion, and a set of open questions.

2 PerSE: A Pervasive Service Environment

In this section we introduce in an informal way the key concepts of this paper by presenting the pervasive service environment PERSE developed by our research group.

Pervasive service environments support the interaction of independent services collaborating to perform an intended action. It is the task of the middleware to connect the services in a pertinent and efficient way. We call this combination of services *complete action*.

[1] In the following, we will just use the male form, but the female form is intentionally included.

2-9525435-0

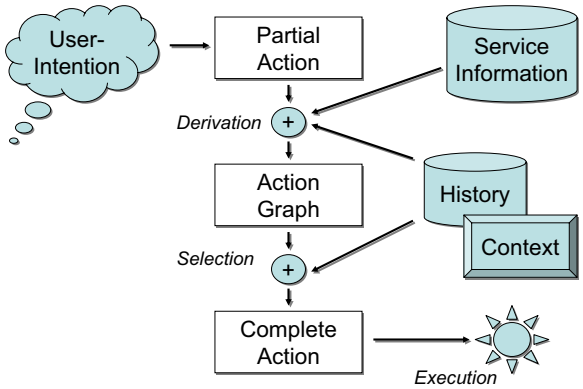

Figure 1. A user intention is transformed into a complete action by selecting the optimal action from all possible complete actions (action graph) matching the user intention, using the knowledge about available services, the context information and the execution history.

PERSE models such a system and offers the interfaces needed to use the managed services without worrying about the limits of the pervasive environment. The system consists of inter-connected bases and services: Each *service* is managed by a base-application called *PerseBase*, corresponding to a device included in the pervasive environment. The PerseBase is responsible for managing the services it proposes and is capable to construct and start complete actions. These complete actions are modeled in PERSE as connected graphs of services.

The environment must select the best action as answer to a user intention with respect of the constraints of the whole system. To describe a user intention, we introduce the concept of a *partial action*, that means a description of the action containing (more or less) exactly defined the data source and the data sink and maybe some steps between. The PerseBase in charge has to derive a complete action from this partial action and the knowledge it has about the available services in the network. This process is sketched in Fig. 1 and can be easily understood with the following example:

A person enters a room and wants to display some presentation about his vacation from his personal notebook. Today, he has to connect his computer to the local network and copy the presentation-file to the local server connected to the video projector, or he can deconnect the projector from the local machine and connect it to his own mobile computer, adjust the screen settings and so forth. A PERSE*-enabled, smart classroom, does not provide the ability to deconnect the local projector cable or to access the room server, because these entities are "invisibly" embedded into the room arrangement. Instead, the user expresses his desire by saying or typing "Show the sunrise presentation on the projector". His notebook, which communicates with the local resources via Bluetooth or WLAN, interprets the user command as a partial action. Using the information about the local available services, context information and the action history, the PerseBase constructs a graph of all possible (and reasonable) service combinations, each representing a complete action matching the given partial action. From these combinations, an algorithm (see Sect. 4.1) selects the "best" one depending on a given cost-function, e. g. the volume of transferred data. Finally, this complete action is executed: the presentation is displayed on the projector.*

3 Modeling User Intention - PsaQL

When someone wants to use a computing system, most of the time he has a very concrete idea of what he wants to do, even if he does not exactly know how to do it. The action which the system should execute seems very clear for him. Unfortunately, this action is not complete that is it lacks some information to be executable.

It is the challenge for the embedded user-intention-interpretation algorithm to create, based on this little information from the user, a complete action which suits best the user intention, maximizing his level of satisfaction with the system. We introduced in [1] a formal language to express partial actions in a system-independent and intuitive way: PsaQL (Pervasive Service Action Query Language). PsaQL plays a similar role in the PERSE-enabled pervasive environment as SQL [3] does for accessing relational database management systems. Therefore, PsaQL looks similar to SQL at the first glance.

Our example (see Sect. 2), expressed in PsaQL:

```
USE sunrise.ppt
   ON BASE notebook
 WITH SERVICE projector
```

In this case the user already has the knowledge, that his presentation is stored in a file named "sunrise.ppt" on his computer accessible as "notebook", and that the video projector is named "projector". If the user is not sure about these parameters, his request could use the "LIKE"-statement (currently treated as regular expression):

Based on this description of a partial action, an action graph of all possible solutions is created and finally, a complete action is selected (see Fig. 1), which can be executed by the pervasive environment. A prototype to demonstrate the process of creating a complete action based on a partial action can be found at `http://liris.wh4f.de/perse`.

4 From User Intention to User Satisfaction

4.1 Translating a Partial Action into a Complete Action

We present in this section an algorithm to translate a partial action into a complete action (see Fig. 1). The main ideas of the mathematical formalization are provided here:[2]

Definition 1 (Partial Action) *Let B be the set of bases, S the set of services and $S(b)$ the set of available services on a base b, where $S(b)$ is equal to S when $b = \perp$.*[3] *A* partial action *p, the formal expression of a user intention, can be modeled as a list of 2-tuples:*

$$p = (e_1, \cdots, e_n) \mid e_i = (b_i, s_i) \text{ with } b_i \in \{\perp\} \cup B; s_i \in \{\perp\} \cup S(b_i) \quad \forall i : (b_i \neq \perp) \vee (s_i \neq \perp) \tag{1}$$

Definition 2 (Service Graph) *Let $E = \{\epsilon = (b, s) \mid b \in B, s \in S(b)\}$ be the set of all available services in a pervasive service environment and $I(E) \subseteq E \times E$ the set of all possible service interactions within this environment.*[4] *Then we can define a service graph in a part $\mathcal{E}$ of the pervasive service environment as*

$$G_{\mathcal{E}} = (\mathcal{E}, \mathcal{I});\ \mathcal{E} \subseteq E;\ \mathcal{I} \subseteq I(\mathcal{E}) \tag{2}$$

The set Γ_E of all valid service graphs in E is defined as:

$$\Gamma_E = \{(\mathcal{E}, \mathcal{I}) \mid (\mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E}))\} \tag{3}$$

Definition 3 (Connected Service Graph) *A service graph $g = (\mathcal{E}, \mathcal{I});\ \mathcal{I} \subseteq I(\mathcal{E})$ is called* connected *iff*

$$\forall \epsilon_0, \epsilon_n \in \mathcal{E},\ \epsilon_0 \neq \epsilon_n :\ (\epsilon_0, \epsilon_n) \in \mathcal{I} \vee (\exists \epsilon_1, \ldots, \epsilon_{n-1} : (\epsilon_i, \epsilon_{i+1}) \in \mathcal{I}; (i = 0, 1, \ldots, n-1)) \tag{4}$$

Definition 4 (Solution) *A graph $g_p^E = (\mathcal{E}, \mathcal{I});\ \mathcal{E} \subseteq E, \mathcal{I} \subseteq I(\mathcal{E})$ is called* solution *for a given partial action p in a pervasive service environment E iff*

$$g_p^E \in \Gamma_E \tag{5}$$

$$g_p^E \text{ is connected} \tag{6}$$

$$\forall e = (b, s) \in p\ \exists \epsilon = (\beta, \sigma) \in \mathcal{E} \text{ with } \begin{cases} \beta = b & \text{if } s = \perp, \\ \sigma = s & \text{if } b = \perp, \\ (\beta = b) \wedge (\sigma = s) & \text{otherwise.} \end{cases} \tag{7}$$

[2] To simplify the model, we do not consider attributes in the following section, they can be easily added lately.

[3] $\perp$ represents "undefined".

[4] The interoperability between services is not studied here. We assume that $(\epsilon_1, \epsilon_2) \in I(E) \Leftrightarrow ((\epsilon_1, \epsilon_2) \in E \times E) \wedge (\epsilon_1$ is interoperable with $\epsilon_2)$.

Definition 5 (Action Graph) *Let $\mathcal{S}_p^E$ be the set of all solutions for p in a pervasive service environment E. An* action graph *$A_p^E \in \Gamma_E$ of a partial action p in the pervasive service environment E is a connected service graph containing all solutions for p:*[5]

$$A_p^E = \Big(\bigcup_{(\mathcal{E}, \mathcal{I}) \in \mathcal{S}_p^E} \mathcal{E},\ \bigcup_{(\mathcal{E}, \mathcal{I}) \in \mathcal{S}_p^E} \mathcal{I} \Big) \tag{8}$$

Definition 6 (Complete Action) *Let $C(g, \gamma)$ be the function calculating the cost of a solution g when it is executed in a context γ. Then the* complete action *c_p^E for a given partial action p in a service environment E is defined as:*

$$C(c_p^E, \gamma) = \min\{C(g_p^E, \gamma) \mid g_p^E \in \mathcal{S}_p^E\} \tag{9}$$

Let $H(p, E, \gamma)$ be the function fetching from the execution history the complete action for a given partial action p and a pervasive service environment E in a context γ. Then, with these definitions and declarations, we proposed in [1] an algorithm using an heuristic approach to solve the translation in polynomial time. To receive stable benchmark results, we designed a second, exhaustive algorithm (Algorithm 1), transforming a user intention p into a complete action c_p^E.[6] Starting with an action graph (representing a solution for p), this algorithm removes step by step the connections (see line 11 of algorithm 1). When a service graph derived by this has no longer any successors which are solutions (line 16), this service graph is a possible candidate for the complete action (line 17). The candidate with the minimal glolbal cost is selected as solution of the algorithm.

The complexity of this algorithm is exponential, but it finds always the optimal action for a given partial action. For small pervasive service environments (small number of bases and services), this algorithm calculates in a reasonable time the best possible complete action for a given partial action.

4.2 Complete Action Representation

To represent the action graph and a complete action, we need a suitable data structure. Each used service is connected with other services, therefore we model the action graph as well as its thinned out version, the complete action, as a set of channels, connecting output- and input-ports of the participating services.

[5] In some rare cases (when $\forall (b, s) \in p :\ b = \perp$) it can happen, that A_p^E is not unique. If so, algorithm 1 (see below) has to be executed for each $(A_p^E)_i$.

[6] A_p^E can be directly derived from $(E, I(E))$.

Algorithm 1 Exhaustive Translation Algorithm

1: $c = H(p, E, \gamma)$ *// try to find a solution in history*
2: **if** $c \neq \perp$
3: $c_p^E = c$
4: **else**
5: fifo A, fifo S
6: push(A,A_p^E)
7: **while** A $\neq \perp$
8: $(\mathcal{E}, \mathcal{I})$ = g = shift(A) *// take a service graph from the beginning of the list*
9: is_leaf = **true**
10: **for each** $\iota \in \mathcal{I}$
11: $\mathcal{I}' = \mathcal{I} - \{\iota\}$ *// remove connection ι from $\mathcal{I}$*
12: $\mathcal{E}' = \bigcup_{(\epsilon_1,\epsilon_2)\in\mathcal{I}'} \{(\epsilon_1, \epsilon_2)\}$ *// keep connected elements*
13: **if** $g' = (\mathcal{E}', \mathcal{I}')$ is solution for p
14: push(A,g') *// remember this successor*
15: is_leaf = **false**
16: **if** is_leaf
17: push(S,g) *// g is a solution candidate*
18: g = shift(S)
19: **for each** $g' \in$ S *// find minimal solution*
20: **if** $C(g', \gamma) < C(g, \gamma)$
21: $g = g'$
22: $c_p^E = g$

To describe a complete action, we use the object model sketched in Fig. 2. The utilized classes are:

- *Address* - Combination of a type (e. g. `"ipv4"`) and a valid address-string (e. g. `"172.20.0.9:8080"`)
- *Base* - Represents a PerseBase, has one or more *Addresses* and manages *Services*
- *Service* - A service published on a *Base* proposing *Ports* for data input and output
- *Port* - Representation for data-flow input and output of a *Service*. Contains information about the type of data-flows accepted/produced and the direction of the data-flow (in/out)
- *Element* - Combination of a *Service* with adaptation information (*Attributes*)
- *Attribute* - A string transmitted to a service to adapt it to the specific needs of a PERSE-action
- *Stub* - Combination of an *Element* with a service-*Port*
- *Channel* - The two *Stubs* forming a data-flow between two services
- *Action* - A collection of *Channels*

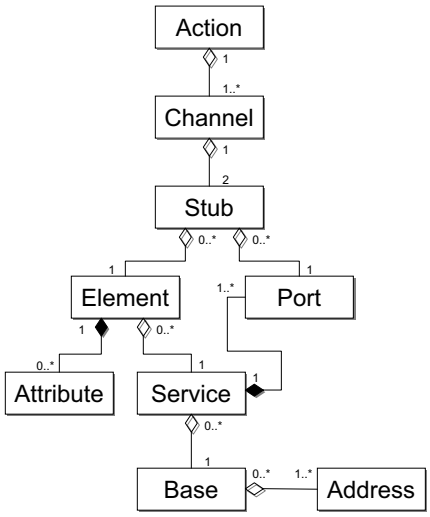

Figure 2. Class diagram for the internal representation of action graphs and complete actions.

5 Implementation and Evaluation Issues

5.1 Service Description Management

The objects which form an action graph are created based on a database connected to the system performing the translation process.[7] This database contains descriptions of available PerseBases, services and the description of these services. Where such a database initially just contains information about local available services, each PerseBase supports a HTTP-based interface to access this information. We have designed a PerseBase-API supporting the following requests:

- *allObjects* – Returns a list of all known neighbor-bases (including the requested PerseBase itself) as well as their connection information (IP-address for instance) and other meta-data
- *services* – Returns a list of local available services accompanied with meta data like name, keywords etc.
- *serviceDescription* – Returns a detailed description of a given service containing information about ports, treatable data types and so on

The results of this interface requests are transmitted using specific XML schemas. The interoperability of services is determined using the `accepted_data_types` information of a service description.

[7] Normally, this will be a PerseBase

5.2 Evaluation Metrics

In summary the following domains for measuring algorithms translating a user intention into a complete action have been identified in [1]:

- *User satisfaction* as the most important goal of algorithm design in a pervasive environment.
- *Execution time* as the factor having the biggest impact on the user satisfaction.
- *Network data transmission* as having an important impact on the execution time. Beside the *network speed* which cannot be influenced by the design of the translating algorithm, the *size of the transferred service descriptions* and the *transmission distance* constitute this value.
- *Scalability* of the algorithm when increasing the *number of available services* or the *length of the query* (see chapter 5.3).
- *Pervasive Computing Constraints* as an *accountable usage of CPU and memory resources*.

5.3 Benchmark Results

To test our prototype implementation of a simple transforming algorithm, we evaluated the volume of the transferred data when varying the number of available PerseBases from 10 to 1000 in steps of 10 bases. To examine the *scalability* of the solution, we introduced a *scalability factor* which calculates the average distance covered by transmitted data:

$$f_{scal} = \frac{\sum_{msg} size(msg) * distance(msg)}{\sum_{msg} size(msg)}$$

A first implementation ("Algorithm A") fetches the total available service description at the beginning and performs the creation of a complete action on base of these descriptions. This implementation corresponds in its benchmark results to the exhaustive algorithm presented in section 4.1. In a second implementation ("Algorithm B") we have tried to minimize the amount of transferred data. To reach this goal, we introduced some constraints for the implementation of this prototype. The selection of the complete action from the action graph is done on-the-fly and the implementation fetches at the same time the service descriptions from the network, just until the algorithm found a solution. Another restriction of this second implementation is the composition of services by linear concatenation.

The test environments have been created with a uniform distribution of n PerseBases in a region of fixed size. The connecting network has been created using the PLNGen algorithm, developed for this testcase and modeling pervasive networks[8]. The connections between the simulated bases try to model a heterogeneous pervasive network with short distance and long distance connections. We analyzed queries of the type "`USE BASE` x `WITH BASE` y`...`" with a length varying from two to ten enchained entities.

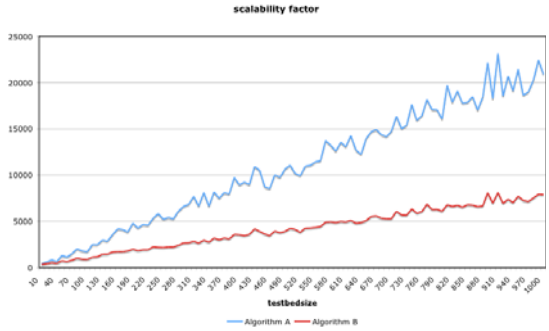

Figure 3. The progression of the scalability factor as introduced in the text when varying the testbed size.

The results of our first test runs (see Fig. 3) show that the currently implemented algorithm reacts very sensitively to an increasing of the number of available PerseBases. It is at most the aspect of the increasing distance between the requesting and the service-providing PerseBase raising the value exponentially.

6 Related Work

The idea to execute pervasive applications in a "user-aware" way has been worked out for instance in [13]. Similar as PERSE does, El-Kathib et al. design in [4] a platform trying to maximize user satisfaction while adapting the content of multimedia data with a dynamically estimated path of enchained transcoders. His solution also relies on graph base composition, where he does not present a formal language to define the adaptation requests.

Other user-oriented systems for managing pervasive environments have been developed, for instance Gaia [12] or Aura [5]. Gaia proposes a programming language to construct executable tasks based on the interoperability of services, whereas Aura tries to avoid any interaction with the user and does not present a model for user intention.

The Ninja Environment [6] introduced the idea of composing services on distributed adaptation paths, whereas [2] added path optimization considerations for content adaptation. In [15], M. Valle et al. introduce a system for dy-

[8] see `http://liris.wh4f.de/pln_gen_algo.html`

namic service composition in intelligent environments: they are following a related way as PERSE does, from a partial action (what they call *abstract plan*) through a composition algorithm to a concrete action, in their words *detailed plan*. They have worked out well the mechanisms for service descriptions and service composition, while they do not present a formal way to express intuitive and computer-interpretable user intention in form of a partial action. They rely on there part on a library of predefined abstract plans, which can be interpreted as a kind of predefined execution history, but this history is not taken directly into account when composing services.

Another widely explored field of research inspiring the development of this work are Semantic Web Services, as presented for instance by [9]. Ontologies as defined by OWL-S [8] can help to create correspondent and valid action graphs and maximize the user satisfaction with a calculated solution. Semantic composition of Web Services is as well introduced by [14] among others.

A pervasive environment will be characterized by the availability of application context information [7]. PERSE will use the contextual information to build an appropriate action graph and to select the best solution as complete action. Earlier approaches introducing context based adaption into pervasive environments are presented in [11] and [10].

7 Conclusion and Open Issues

This paper has presented a strategy and a methodology to take the user intention into account when composing service-based actions in a pervasive service environment. We outlined an exhaustive algorithm extending this partial action to an executable graph of services (complete action) using the service descriptions, the context information and the execution history.

We presented an object-oriented model of complete actions. Finally, some metrics supposed to model the user satisfaction and environment scalability has been worked out and performance tests have been applied to a prototype implementation.

In future, the implemented algorithm will be further developed to take the execution history into account. This is intended to lower significantly the execution time of the translation algorithm, as the typical actions in a pervasive service environment are frequently reexecuted. We estimate that about 90 % of the actions can completely or partially base on already executed actions stored in the history, and just a small amount of 10 % has to be completely new calculated. As well, security will be important in further PERSE-development. The current presumption that all services can be used by everyone cannot be hold in real world application, a system of authentication and authorization based on roles and access control lists will be implemented.

References

[1] P. Bihler, L. Brunie, and V.-M. Scuturici. Modeling user intention in pervasive service environments. In *Proceedings of the IFIP-EUC'2005*, (to appear) 2005.

[2] S. Buchholz and T. Buchholz. Adaptive content networking. In *ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies*, pages 213–219. Trinity College Dublin, 2003.

[3] C. J. Date. *A guide to the SQL standard*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[4] K. El-Khatib, G. v. Bochmann, and A. E. Saddik. A qos-based framework for distributed content adaptation. In *Quality of Service in Heterogeneous Wired/Wireless Networks, QSHINE 2004*, pages 308–312, 18-20 Oct. 2004.

[5] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*, 1(2):22–31, 2002.

[6] S. D. Gribble, M. Welsh, R. von Behren, E. A. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R. H. Katz, Z. M. Mao, S. Ross, B. Zhao, and R. C. Holte. The ninja architecture for robust internet-scale systems and services. *Computer Networks*, 35(4):473–497, March 2001.

[7] J. Ma, L. T. Yang, B. O. Apduhan, R. Hunag, L. Barolli, and M. Takizawa. Towards a smart world and ubiquitous intelligence: A walktrough from smart things to smart hyperspaces and ubickids. In *International Journal of Pervasive Computing and Communications*, volume 1, pages 53–68. Troubador Publishing Ltd., March 2005.

[8] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. online: http://www.w3.org/TR/2004/REC-owl-features-20040210/, 2004.

[9] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, 16(2):46–53, March 2001.

[10] S. K. Mostéfaoui. A Context-Based Services and Discovery and Composition Framework for Wireless Environments. In *Proc. of the 3rd IASTED International Conference on Wireless and Optical Communications (WOC'2003)*, pages 637–642, Banff, Canada, 14-16 July 2003.

[11] A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, December 2003.

[12] M. Román, C. Hess, R. Cerqueira, A. Ranganat, R. H. Campbell, and K. Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, pages 74–83, Oct-Dec 2002.

[13] J. P. Sousa and D. Garlan. Improving user-awareness by factoring it out of applications. In *UbiSys'03 - System Support for Ubiquitous Comp. Workshop*, October 12 2003.

[14] S. Staab, W. M. P. van der Aalst, V. R. Benjamins, A. P. Sheth, J. A. Miller, C. Bussler, A. Maedche, D. Fensel, and D. Gannon. Web services: Been there, done that? *IEEE Intelligent Systems*, 18(1):72–85, 2003.

[15] M. Vallée, F. Ramparany, and L. Vercouter. Composition flexible de services d'objets communicants. In *UBIMOB 05*, Mai 31 - June 3 2005.