Modeling User Intention in Pervasive Service Environments

Pascal Bihler, Lionel Brunie, and Vasile-Marian Scuturici

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

Abstract. The introduction of pervasive computing environments in everyday life will not just be a big step for users, but also for application designers. The well defined interaction interfaces will make place for other, more intuitive ways of interaction. It is the challenge for a pervasive system middleware to capture and model the user intention in a smart way and to solve ambiguousness in the user's expression of a pervasive action. This paper introduces the Pervasive Service Action Query Language (PsaQL), a language to formalize the description of a user intention using composed pervasive services. The work describes a way of translating the user intention into an executable action and propose algorithms performing this translation. Considerations to implement this process are given within the scope of PerSE, a pervasive service environment developed by our research group, together with general evaluation metrics for such algorithms.

1 Introduction

"Pervasive" or "Ubiquitous" Computing – as Mark Weiser calls it in his trendsetting paper "The Computer for the 21st Century" [1] – left its cache and starts to become everyday reality. A network of omnipresent, highly embedded computing machines (possibly with limited power and communication resources) 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 do not 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. Today, most machines react to more or less formal commands given by the user, e. g. switching the television channel if the user presses some button on his remote control at the beginning of the advertising block. In the future, knowing the user's dislike for this kind of information service, a proactive television could immediately switch the channel, mute the screen and sound for the duration

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

of the advertisement or present background information about the interrupted broadcasting from the Internet.

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 (voice command or similar) 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 presents the first step towards interpreting a user intention and describes in a formal way a corresponding action using service composition. The proposed approach is integrated in the PerSE environment.

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 introduce an intuitive formal language which can work as an intermediate step between the user's expression of an action intention and the system internal action representation. The translation process between the user's and the machine's interpretation is presented. This is followed by some benchmarks guidelines 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 present the pervasive service environment PerSE developed by our research group. This allows us to introduce in an informal way the key concepts of this paper that will be formally defined in the next sections.

Pervasive service environments support the interaction of independent services collaborating to perform an intended action. Examples for such services are a filesystem interface, a translating service, a laser pointer acting as an input device or a video projector to visualize information. 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*.

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 basic architecture of PerSE is designed like this: Each *service* is managed by a base-application called *PerseBase*, corresponding to a device included in the pervasive environment. The PerseBase, which is connected to other PerseBases, 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.

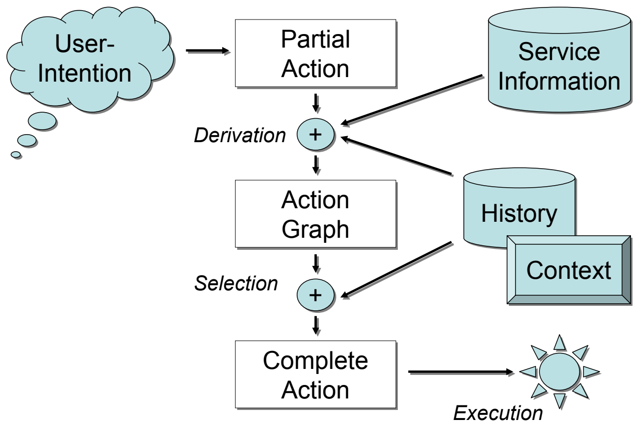

Fig. 1. A user intention is transformed into a complete action by selecting the optimal action from all possible complete actions (action graph) matching the user intention, using the knowledge about available services, the context information and the execution history.

The environment must select the best action as answer to a user intention with respect of the constraints of the whole system (i. e. data transmission speed, reliability, . . .). 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. The challenge of deriving this complete action from a partial user input is studied in this work. Detailed information about this process can be found in Sect. 3.

The process of transforming a user intention into an executable series of commands for the pervasive environment can easily be understood with the following example, which will accompany us through the rest of this paper:

A person enters a room and wants to display some presentation about his vacation from his personal notebook. With today technologies and softwares, he has to connect his computer to the local network and copy the file containing the presentation to the local server connected to the video projector. Then he has to find the appropriate program on this machine to present his slides. Alternatively, 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 runs a PerseBase as well and communicates with the local resources via Bluetooth or*

WLAN, interprets the user command as a partial action with a service delivering a presentation*, the attribute* sunrise *for this service, and a connection with a service called* projector*. Using the information about the local available services, context information (for instance the room the user has entered) and the action history (maybe the user has already ran a request like this in the past, so if this action is still valid, it can be reused), the PerseBase constructs a graph of all possible (and reasonable) service combinations, each representing a complete action matching the given partial action (see Fig. 1). From these combinations, an algorithm (see Sect. 4) 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

Even if the user interface once will be hidden completely into the background, there will still be the need of expressing a partial action in a formal way. This representation can be used directly to express an intention or as an exchange format used by parts of the PerSE middleware system. Therefore we decided to define a formal language, PsaQL (Pervasive Service Action Query Language). PsaQL plays a similar role in the PerSE-enabled pervasive environment as SQL [2] does for accessing relational database management systems. PsaQL also looks similar to SQL on the first glance (BNF notation):

```
<partial_action> ::= USE <action_part> [<ext_action>]
<ext_action> ::= WITH <action_part> [<ext_action>]
<action_part> ::= <attr_constr_def>[ FOR <service_constr_def>]
                                   [ ON <base_constr_def>] |
                  <service_constr_def>[ ON <base_constr_def>] |
                  <base_constr_def>
<base_constr_def> ::= BASE <base_constraint>[ AS <name>]
<base_constraint> ::= <name> | LIKE "<partial_name>"
<service_constr_def> ::= SERVICE <service_constraint>[ AS <name>]
<service_constraint> ::=  <name> | LIKE  "<partial_name>"
<attr_constr_def> ::= (<name> | LIKE "<partial_name>")[ AS <name>]
<name>  ::= {<'a'-'z', 'A'-'Z', '0'-'9','_'>}
<partial_name>  ::= {<'a'-'z', 'A'-'Z', '0'-'9','^','$',
                     '(',')','[',']','.','+','*','?', ...>}
```

Our example (see Sect. 2) can be expressed in PsaQL as:[2]

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

[2] In this case the user already has the knowledge about the current device names. If the user is not sure about these parameters, his request could be expressed using the LIKE-statement, which currently advises the algorithm in charge to interpret the associated character chain as a regular expression.

4 From User Intention to User Satisfaction

We present in this section an algorithm to translate a partial action into a complete action (see Fig. 1). A *partial action* is a formal representation of a user intention whereas a *complete action* is a connected graph of services, representing the best combination of services to satisfy the user intention. The translation process consists of three steps:

1. Translate the user-input (given for instance in PsaQL) into an internal model of a partial action
2. Expand the partial action to an *action graph* using the service description database, the action history, the context information and heuristic strategies
3. Select the best solution as complete action

With some mathematical formalization, we can define step two and three of this algorithm:[3]

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 $b = \perp$.*[4] *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.*[5] *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*

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

[3] To simplify the model, we do not consider attributes in the following section, they can be easily added lately.
[4] $\perp$ represents "undefined".
[5] 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 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*

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

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

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

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*:*[6]

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

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:*

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

An algorithm using these definitions and declarations to derive the best solution for a given partial action p would be straight-forward, but unfortunately with an exponential complexity. This is not applicable in larger pervasive service environments, so we developed an heuristic approach, solving the query in polynomial time (Alg. 1), whereas $H(p, E, \gamma)$ represents 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. The approach bases on the following coloring of the nodes, defining a partition of E:[7]

- *Black nodes* ϵ_B: $\forall \epsilon_B = (\beta, \sigma) : \exists (\beta, \sigma) \in p$
- *Red nodes* ϵ_R: $\forall \epsilon_R = (\beta, \sigma) : \exists (\beta, \perp) \in p \vee \exists (\perp, \sigma) \in p$
- *White nodes* ϵ_W: all other nodes

The solution we want to find includes all black nodes, some of the red nodes (so that there is for every $e \in P$ an appropriate node in the solution) and maybe some white nodes. To find the "best" red nodes, we use the following heuristic:

First we calculate using the Shortest Path Algorithm of Dijkstra [3, p. 204] for each red node ϵ_R the distance to all black nodes ϵ_B by using a modified cost function $C_I((\epsilon_R, \epsilon_B), \gamma)$. Beginning with the red node of the shortest distance,

[6] In some rare cases (when $\forall (b, s) \in p :\ b = \perp$) it can happen, that A_p^E is not unique, but this does not imply the algorithm presented below.

[7] Each node wears just one color, a hierarchy is given as: *Black* > *Red* > *White*, at least one black node is required.

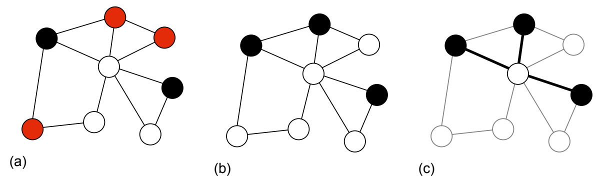

Fig. 2. In this example, the partial action p contains three elements. Two of them define exactly a corresponding (black) node in the action graph and one matches to three different (red) nodes (a). From the red nodes, the one with the nearest distance to a black node is selected and colored black, the others are colored white (b), and the Minimum Cost Spanning Tree containing all black nodes is calculated (c).

we examine all red nodes in order of their distance: If there is a corresponding query part e in the partial action p, which is not already covered by any black node, we color the node black. We repeat this, until every part of p is covered with a black node. The remaining red nodes are colored white. Then, using the Minimum Cost Spanning Tree Algorithm [3, p. 208] we extract from $(E, I(E))$ the graph containing all black nodes as a complete action (see Fig. 2).

5 Evaluation Metrics for a Pervasive Service Environment

To achieve the goal of seamless integration of the pervasive computing system into the user's everyday life environment, it is important that the user percept nearly no delay between expressing his intention and receiving the result of it by the executed action. This leads to the goal of short *execution time* for any action algorithm. This is directly related with the *user satisfaction*, the most important aspect which is nevertheless difficult to measure. The execution time of the transforming process from a partial action to a complete action is just little connected with the complexity of the algorithm, in a pervasive network with a couple of services data transmission costs are much more important. This defines the challenge of minimizing the *network data exchange*, viz minimizing the *size of transferred service descriptions* and the *transmission distance*. All these parameters should not blow the execution of the translating process up when increasing the number of known services and the length of the requests, that means *scalability* as a development goal. Other important issues, when implementing the algorithms on a smart device like a PDA or a wristwatch are the *minimization of CPU-usage* to save energy resources as well as minimizing the *memory usage* to save little memory resources.

6 Related Work

To execute pervasive applications in a "user-aware" way has been worked out in [4] and [5]. Similar as PerSE does, El-Kathib et al. design in [6] a platform trying

Algorithm 1 Heuristic 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:    list $F$  // black nodes
6:    list $R$  // red nodes
7:    for each $\epsilon = (\beta, \sigma) \in E$
8:       if $\exists e = (b.s) \in p : (b = \beta) \wedge (s = \sigma)$
9:          push($F, \epsilon$)
10:      else if $\exists e = (b.s) \in p : ((b = \beta) \wedge s = \perp) \vee ((s = \sigma) \wedge b = \perp)$
11:         push($R, \epsilon$)
12:   array $D$  // calculate the minimal distance
13:   for each $\epsilon_f \in F$
14:      for each $\epsilon_r \in R$
15:         $d$ = length(dijkstra($(E, I(E))$,$\epsilon_f$,$\epsilon_r$))
16:         if $d < D[\epsilon_r]$
17:            $D[\epsilon_r] = d$
      // elements of partial action covered only by red nodes:
18:   $p' = \{e = (b, s) \in p \mid (b = (\perp) \vee (s = \perp) \wedge (\nexists \epsilon_f = (\beta, \sigma) \in F : (b' = \beta) \vee (s' = \sigma))\}$
19:   sort($R$,$D$)  // sort red nodes by distance
20:   for each $\epsilon_R = (\beta, \sigma) \in R$
21:      if $\exists e = (b, s) \in p' : (b = \beta) \vee (s = \sigma)$
22:         push($F, \epsilon_r$)  // color the node black
            // removed satisfied parts from the partial action:
23:         $p' = p' - \{e' = (b', s') \in p' \mid (b' = \beta) \vee (s' = \sigma)\}$
24:   $c_p^E$ = MCST($(E, I(E))$,$F$)  // calculate Min. Cost Spanning Tree on black nodes
```

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. Where it is currently optimized for fixed network structures like available in the Internet, the platform is already resilient against service failures and hereby maybe as well applicable in a pervasive context.

Other user-oriented systems for managing pervasive environments have been developed, for instance Gaia [7] or Aura [8]. 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.

In [9], M. Valle et al. introduce a system for dynamic 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.

C. Linnhoff-Popien et al. present in [10] a language to describe service request in computer networks. This language specifies in details the semantic of a requested service but has not the intention to support service enchainment as multistep action in a pervasive service environment. The Human-Markup-Language (HumanML) [11] tries to offer an XML-based description for human interacting, but it does not focus on using services in a pervasive environment. Therefore it cannot be used as a replacement for PsaQL, but nevertheless act as user input for creating the partial action.

Another widely explored field of research inspiring the development of this work are Semantic Web Services, as presented for instance by [12]. Ontologies as defined by OWL-S [13] 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], [15], [16], and [17]. The general challenge of matching several demands semantically on web resources based on their descriptions is treated by [18].

A pervasive environment will be characterized by the availability of application context information [19]. PerSE will use the contextual information to build an appropriate action graph and to select the best solution as complete action. Earlier approaches of introducing context based adaption into pervasive environments are presented in [20] and [17]. When one wants to use public available services in a pervasive network seamlessly, the security is one of the main points. First attempts are made by introducing smart authentication into PerSE like worked out in [21].

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. It introduced PsaQL, a language to express user intention in a pervasive service environment. We outlined algorithms extending this partial action to an executable graph of services using the service descriptions, the context information and the execution history. The next step will be an implementation of the translation service as prototype and the evaluation of this prototype in an appropriate environment.

Following this research work, a couple of issues remain open to be developed in future time. Capturing the user intention using PsaQL will not stay the last conclusion of wisdom, more seamless ways like voice or gesture recognition will be developed. Independent of the kind of expression the user selects to communicate his intention, PsaQL will be an easy interface between the intention capturing module and the module developing the complete action to execute.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American (September 1991) 94–104
2. Date, C.J.: A guide to the SQL standard. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1986)
3. Manber, U.: Introduction to Algorithms: A Creative Approach. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1989)
4. Sousa, J.P., Garlan, D.: Improving user-awareness by factoring it out of applications. In: UbiSys'03 - System Support for Ubiquitous Comp. Workshop. (2003)
5. Saif, U., Pham, H., Paluska, J.M., Waterman, J., Terman, C., Ward, S.: A case for goal-oriented programming semantics. In: UbiSys'03 - System Support for Ubiquitous Computing Workshop. (2003)
6. El-Khatib, K., v. Bochmann, G., Saddik, A.E.: A qos-based framework for distributed content adaptation. In: Quality of Service in Heterogeneous Wired/Wireless Networks, QSHINE 2004. (2004) 308–312
7. Román, M., Hess, C., Cerqueira, R., Ranganat, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing (2002) 74–83
8. Garlan, D., Siewiorek, D., Smailagic, A., Steenkiste, P.: Project aura: Toward distraction-free pervasive computing. IEEE Pervasive Computing **1** (2002) 22–31
9. Vallée, M., Ramparany, F., Vercouter, L.: Composition flexible de services d'objets communicants. In: UBIMOB 05. (2005)
10. Popien, C., Meyer, B.: A service request description language. In: FORTE'94. Chapman & Hall, Bern (1994) 14–32
11. Best, K.F.: Oasis standards work. Markup Lang. **3** (2001) 241–249
12. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems **16** (2001) 46–53
13. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. online: http://www.w3.org/TR/2004/REC-owl-features-20040210/ (2004)
14. Staab, S., van der Aalst, W.M.P., Benjamins, V.R., Sheth, A.P., Miller, J.A., Bussler, C., Maedche, A., Fensel, D., Gannon, D.: Web services: Been there, done that? IEEE Intelligent Systems **18** (2003) 72–85
15. Sheshagiri, M., desJardins, M., Finin, T.: A planner for composing services described in DAML-S. In: Proceedings of ICAPS'03 Workshop on Planning for Web Services. (2003)
16. Sycara, K.P., Paolucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. J. Web Sem. **1** (2003) 27–46
17. Vukovic, M., Robinson, P.: Adaptive, planning-based, web service composition for context awareness. In: Second Int. Conference on Pervasive Computing. (2004)
18. Noia, T.D., Sciascio, E.D., Donini, F.M., Mongiello, M.: A system for principled matchmaking in an electronic marketplace. In: WWW. (2003) 321–330
19. Ma, J., Yang, L.T., Apduhan, B.O., Hunag, R., Barolli, L., Takizawa, M.: 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. Troubador Publishing Ltd. (2005) 53–68
20. Ranganathan, A., Campbell, R.H.: An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing **7** (2003) 353–364
21. Saadi, R., Pieson, J.M., Brunie, L.: APC: Access pass certificate. distrust certification model for large access in pervasive environment. In: IPCS'05. (2005)