iShadow: Yet Another Pervasive Computing Environment

Daqiang Zhang, Hu Guan, Jingyu Zhou, Feilong Tang, Minyi Guo
Department of Computer Science
Shanghai Jiao Tong University
Shanghai 200240, P. R. China

Abstract

Previous architectures of pervasive computing are customized for specific types of applications. In this paper, we propose a new architecture named iShadow, which facilitates the design and implementation of generic applications in pervasive computing environment. iShadow gracefully integrates physical spaces and human attention, and provides fundamental and flexible support to construct pervasive applications rapidly. Significant differences of iShadow from previous works are lightweight user-shadow model, scalable resource discovery and potent context inference mechanism. Our prototypes demonstrate that the iShadow architecture is robust, feasible and effective for pervasive applications.

1. Introduction

Pervasive computing [25] has posed many significant challenges to software engineering. The essence of pervasive computing is creating an environment that seamlessly integrates computation and human attention. A number of representative architectures or applications for demonstrating pervasive computing, including Oxygen [16], Aura [19], Endeavour [22], Gaia [17] and Easyliving [3], have addressed different aspects in pervasive computing [18]. It is unclear that all these solutions could be successful when attempting to cover all pervasive computing devices, ranging from laptops and sensors to mobile phones. These devices are heterogeneous in terms of platforms, power, storage capacity and user interfaces. With increasing heterogeneity, developing applications that runs across all platforms will result in exceedingly complex even inextricable programming issues. There are many unsolved issues due to lack of the standards and guidelines for pervasive computing [18, 8]. Although Satyanarayanan [18] enumerated four research thrusts of pervasive computing, these research concerns are highly abstract for implementation. To articulate the risks, we characterize pervasive computing by four challenges, which have guided our architecture development.

The first challenge comes from Context Awareness. Pervasive computing visions computation, devices and services as ubiquitous, whether with users or embedded into a specific environments. The result of this vision requires that the architecture is aware of environmental context and user context. The architecture is also responsible for tackling the context variation timely and notifies high-level applications or modules, because the context varies with user movement.

Another challenge arises from Semantic Coherence. Intuitively, the representation and reasoning of context are required to incorporate semantics and keep coherence. Ontology is an ideal representation for context for two reasons: 1) It has been recommended as international standard. 2) It has powerful software support, while other attributed-based and tuple-based representations are still lack of semantics and standards.

The third challenge springs from Cognitive Continuity. In daily work and life, users experience diverse spaces such as office, meeting rooms, home and public areas. User mobility may raise the user distraction and cognitive discreteness, reducing the quality and effectiveness of pervasive software. Consequently, the architecture is expected to offer a systematic, continuous and scalable strategy to make the context representation and user recognition consistent.

The fourth challenge is derived from Sentient Software. Pervasive computing environment seriously suffers from different devices capacity, energy consumption and constant varying context. In support of heterogeneous devices as well as various user preferences, the architecture is supposed to be adaptive and intelligent in terms of dynamical configuration and reconfiguration.

In this paper, we are developing a pervasive computing architecture named Intelligent Shadow (iShadow) to provide minimal but flexible system support for adaptive pervasive applications. iShadow denotes that our system accompanies the user like a shadow, and offers personalized services to the user according to the user context. Specifically, iShadow concentrates on the general architecture with flexible low-level support to the pervasive applications through
gracefully integrating human attention with computation. To summarize, our main contributions are: 1) We propose a lightweight user-shadow model to cater for the context awareness and track the user anytime, anywhere. Moreover, the user-shadow can be tailored to the context and user preferences. 2) We provide scalable, distributed resource discovery mechanism, which is based on hash structure with two synchronization methods. 3) We offer a potent context inference mechanism. iShadow employs the ontology to represent context and Bayesian networks for causal reasoning that ontology does not cope well with.

The rest of this paper is organized as follows. Section 2 overviews the iShadow architecture. Section 3 discusses the lightweight user-shadow model. Section 4 highlights the design of scalable resource discovery. Section 5 explains the context inference mechanism. Section 6 reports case study and Section 7 concludes the paper with our future work.

2. iShadow Architecture Overview

![Figure 1. iShadow architecture](image)

Figure 1 illustrates the hierarchical structure of iShadow, consisting of devices layer, middleware layer and application layers. The devices layer provides a means for users to access services in pervasive computing environments. Computing devices include PC, cell phone and PDA; sensor devices mainly contain the devices to sense the physical environment; network devices make sure the network is connected. These devices differ in terms computation capability, memory and human interface, leading to the heterogeneity in pervasive computing.

The middleware layer is crucial to iShadow, which consists of resource management, user management, context awareness, message management, service management and data management, where resource management and user management are two essential modules. We restrict the communication among middleware modules by means of security control bus. Broad consensus of design and development has been reached from the existing architectures of pervasive computing environment [14, 6, 7, 2]. For instance, data management, service management and message management are almost implemented in a parallel manner, whereas context representation using ontology is much efficient [5, 24]. We designed the iShadow architecture similar to the previous work. Meanwhile, our architecture has gradually developed three distinctive characteristics: lightweight user-shadow model in user tracking module, scalable resource discovery in resource discovery module and potent context inference mechanism in context awareness module.

The application layer of iShadow refers to various applications, relying on the services provided by middleware layer. In a sense, any application on specific smart spaces can be rapid prototyped by calling the lower level service. We cooperate with international collaborators to research on key issues of pervasive computing by building smart campus, smart car and smart office using the iShadow architecture. Note that we focus on the salient characteristics of iShadow and omit the remainder modules of iShadow that have been extensively studied Gaia, Easyliving and Aura [19].

3 Lightweight User-Shadow Model

As an essential component of iShadow, user management module includes user tracking, user-shadow and user recognition. We address the following requirements for user-shadow: (i) Deep context-awareness and semantic coherence. It is supposed to always record and recognize individual’s preference, personality and the relationship with environments. (ii) Cognitive continuity. It should move with the individual anywhere and anytime, and provide cognitive consistency for the system. (iii) Sentient software. It ought to be capable of being tailored to different context and environment.

iShadow takes these requirements into account, and separates its function as three aspects. The first aspect is user’s basic information, such as user identification, user preference, location, access permission, owned resources and their functions for specific context. The second aspect is resource requirements that a task may need like memory and network requirements. The third aspect of user-shadow describes its dependencies. It specifies the functionalities that user shadow model depends on, the dependencies for the execution of a model or the situations that the model can be used.

Fig. 2 illustrates an interesting story that a user moves from the dorm to the lab when he is watching movies. The description of user-shadow model at dorm is shown as Algorithm 1. When the user moves into the lab, the context including changes of devices and services will be sensed by resource discovery and user tracking module. Then,
the user-shadow in the lab updates the corresponding items listed as Figure 2(a). The device and the resolution attribute in user task will be changed to PC and 1280X1024, respectively. Latency will also be modified to one second because of 100 Mbps bandwidth available.

However, a drawback is that programmers or users are required to explicitly specify the requirements of user-shadow such as memory usage and task priority, because iShadow cannot parse the resource requirements from the present context when user moves to another environment. For instance, it is unable to confirm whether user like 1280X1024 or 240X320 resolution. Factually, it is almost impossible to catch the ephemeral user behavior, but observations that most people’s behavior and preference in routine life and most human tasks are predictable. In this sense, we are attempting to build an infrastructure that conforms more closely to user behaviors. After the update of user-shadow, sentient software will take over context changes, and make some adaptation.

4. Scalable Resource Discovery

Resource discovery guarantees the client applications automatically locate the required devices and services in a timely and accurate manner. Given the targeted scenarios, requirements for resource discovery are: 1) It is supposed to manage comprehensively the devices and services in terms of dynamism. In pervasive computing environment, the users and their devices frequent join or leave the scenarios. Consequently, resource discovery must keep track of these changes and effectively manage the devices pertained to a specific user or device. 2) It should provide expressive resource description and be capable of choosing a reasonable solution when dealing with many results returned from resource queries. 3) It should work well with respect to scalability and availability throughout large-scale systems.

To satisfy these requirements, we designed a scalable resource discovery mechanism based on hash structure, which includes resource registration, resource query and resource management. Resource registration is in charge of the registration of all devices and services and is implemented based on hash table and XML. Resource query is searching the results that satisfy the users’ requests. Because of the well-designed registration structure, resource query performs efficiently and accurately. Resource update and synchronization are also carried out by resource discovery.

4.1. Resource Description

We choose XML document as the resource description language, because XML document is taken as an exchange format by ISO, for their straightforwardly usable over the internet, human-legible and reasonable representation and interpretability with both SGML and HTML [23]. We classify resource into device and service categories with a set of attributes and constraints.

Algorithm 2 illustrates the code of resource description of device TestBed (PC) and Winword (Microsoft Office Word) service. TestBed has a number of attributes including type, cpu, load, free memory, free disk and latency.
Each attribute is defined as \(<\text{origin, min, max, avg}>\), where \(\text{origin, min, max, and avg}\) denote the origin indicator parameter, the minimum, the maximum and the average values during the discovery phase, respectively. For instance, the record of \(<\text{cpu load}>3G, 0.32, 0.86, 0.57</\text{cpu load}>\) describes that TestBed’s CPU is 3.0Ghz, the minimum, maximum and average CPU overhead in the discovering process is 0.32, 0.86 and 0.57. Furthermore, TestBed also has some constraints for potential applications that may access TestBed. OS and network are two constraints, which indicate the applications must access EPCC before using TestBed with at least 1.2 Mbps bandwidth. For other types of resources, their constraints definition are similar to that of devices. Winword has string attributes \(<\text{type}>, <\text{version}>, and <\text{provider}>\), and also has complex string attributes \(<\text{textedit}>, and <\text{spellcheck}>\), denoting that textedit and spellcheck functions of Winword 2007 are 1.2 and 1.5 times powerful than that of Winword 2003, respectively.

### 4.1.1 Resource Registration

Resource query has great impact on the performance of pervasive systems, because devices and services are heterogeneous and continuously varying. We choose hash tables for resource storage and registration. Compared with resource storage of existing architectures of pervasive computing, our approach has two merits. One is that it has less probability of confliction because of three level hashing structure as well as scalable and robust hash function. The other merit is that it is efficient for lookup operation.

In our hashing scheme, the first bit of hash key is type selection, with 0 and 1 representing device and service, respectively. The rest of hash key is the real key for device or service. For instance, 0561 and 1705 are the hash keys for TestBed device and Winword service, respectively. To explain how the hash keys are created, we introduce a formal description for hash function, which is three tuple \(<C, F, T>\).

- \(C\) is a key, \(|C|\) denotes the cardinality of \(C\), and \(c_i\) is the \(i\)th character of \(C\),
- \(F\) is mapping functions \(f_1, f_2\) and \(f_3\),
- \(T\) is a hash table with slots numbered 0, 1, \(\ldots\), \((|T| - 1)\), and \(|T|\) represents the cardinality of \(T\).

Where \(f_1, f_2\) and \(f_3\) are defined as follows:

\[
\begin{align*}
    f_1 &= 3 * c_1 + c_i \\
    f_2 &= c_{i-1} + c_i, (i \geq 2) \\
    f_3 &= \text{mod}(key, 2^{16} - 1)
\end{align*}
\]

The function that creates hash keys is given as Algorithm 3. Suppose that the device key is ‘dca’. Then, the key \((4, 3, 1)\) for ‘dca’ through calculating ‘\(c_i - \text{‘a’} + 1\)’. Apply Equation 2, we obtain the final hash key.

\[
T = \text{mod}(\sum (4, f_1(c), f_2(c), f_1(a), f_2(a)), 2^{16} - 1)
\]
where

\[ f_1(c) = 3x c_1 + c_2 = 3x4 + 3 = 15, \]
\[ f_2(c) = c_1 + c_2 = 4 + 3 = 7, \]
\[ f_1(a) = 3x c_1 + c_4 = 3x4 + 1 = 13, \]
\[ f_2(a) = c_2 + c_3 = 3 + 1 = 4. \]

Finally, we obtain the hash key for device ‘dca’ as 043.

After creating the hash keys, the system creates an index item to register the resource, as well as stores the related XML files. The whole index is deployed in a web server and distributed to each agent that is a software module, assisting the system to query, update and synchronize resource. The index structure is two-level. First level index divides the 65535 into 256 items and each item stores 256 sub items in its intervals. This hash structure guarantees the efficient query.

4.2. Resource Query

Resource query is easily implemented with XQuery. However, query result is usually not unique because different devices or services may provide a similar function. Therefore, we score each candidate to find the best one. We calculate scores for all constraints. Considering that a user would like to watch movie remotely by EPCC LAN or Dorm LAN. This scenario raises a requirement that the network speed is at least 1 Mbps. If the access speed of EPCC LAN and Dorm LAN are 2 Mbps and 0.5 Mbps, the scores will be 2/(2+0.5)=0.8 and 0.5/(2+0.5)=0.2, respectively. In practice, a request usually has many requirements. As a result, we use weighted sum of the constraints to judge what service or device should be selected. Currently, we have defined some rules for common applications in advance such as ‘text-editing’, ‘video-watching’ and ‘music-listening’.

4.3. Resource Discovery

Resource discovery is to manage resource in pervasive computing where users frequently join or leave specific scenarios with their devices accessing in and out of the networks. Many efforts have been made to keep the resource consistency. One commonly used approach is that the resource manager periodically verifies the connectivity. However, this approach may incur heavy communication overload. Furthermore, clients are just the passive receivers, which results in many unsuccessfully communication with the surrounded services. To overcome this problem, we propose a hybrid mechanism of resource management, which including two levels synchronizing mechanisms. One level is agent update with \( \mu \) as refresh interval; the other level is the update among the agents and resource with \( \lambda \) as refresh interval.

Figure 3 illustrates the refresh process. The refresh interval \( \mu \) is greater than \( \lambda \), indicating the slower update frequency. Each agent is responsible for resources that directly communicate with it and keeps their states updating every \( \lambda \) intervals. A resource advertises itself within \( \lambda \) period and the adjacent agent receives its message and confirms that it is still in the network. Otherwise, the resource is recognized as a leaver. Then, the agent propagates the change to appropriate agents. Correspondingly, when a resource joins the network, the adjacent agent will detect the newcomer within \( \lambda \) period and notifies the related agents to update the resource state timely.

![Figure 3. Resource update and synchronization](image)

5. Potent Context Inference Mechanism

The ultimate goal of iShadow is to reduce user distraction to underlying technologies and provide personalized services for users. However, the context-awareness system faces many challenges and cannot always precisely identify the context. One of the challenges comes from the inaccurate data. The data collected from wired and wireless sensors tends to be very noisy and inconsistent, which leads to pervasive systems uncertain and probabilistic. For instance, Bluetooth, GPS, infrared, RFID and wireless sensors are incapable of accurately positioning the object, although they have achieved widespread success in industry. Another challenge is the context itself, although many efforts have been made on various aspects of context [1, 4, 21], there is still no common context definition and model.

5.1. Ontology-based Reasoning

To make our software and services adaptive, we present context reasoning mechanism into iShadow, involving two policies: ontology-based reasoning and Bayesian networks reasoning. Ontology-based reasoning is easy
to implement that ontology incorporates the semantics into the context expression and provides powerful software support such as Jena [10] and Protege [20]. Table 1 displays partial rules of reasoning rules used in iShadow. Furthermore, we also employ some user-defined rules to assist the system in reasoning dynamic user context (e.g., preference and activity).

<table>
<thead>
<tr>
<th>Relation</th>
<th>Deduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transitive Property</td>
<td>(P rdf:type owl:TransitiveProperty) ∧ (A P B) ∧ (B P C) ⇒ (A P C)</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>(?A rdf:subPropertyOf ?B) ∧ (B rdf:subPropertyOf C) ⇒ (?A rdf:subPropertyOf C)</td>
</tr>
<tr>
<td>subClassOf</td>
<td>(?A rdf:subClassOf ?B) ∧ (B rdf:subClassOf C) ⇒ (?A rdf:subClassOf C)</td>
</tr>
</tbody>
</table>

Table 1. Partial ontology reasoning rules

Considering the scenario that the user customizes its presence to different contacts in mobile instant messenger. We implemented common reasoning rules: event reminder and proximity-based grouping for indoor scenarios. We selected Jena as software tool to support ontology reasoning. More detail can be found in our previous work [9, 13].

In a sense, ontology-based reasoning can be considered as an instance of rule-based one. Nevertheless, rule-based reasoning can not cope well with causal dependencies. It is incapable of handling missing values and dependencies, but they often occur in pervasive computing environment because of its complexity and dynamics. To address such deficiency in ontology-based reasoning, we also implement Bayesian networks as another strategy to infer the high-level context.

5.2. Bayesian Networks Reasoning

The conceptual model of Bayesian networks in iShadow consist of person and current context parts. Person part collects user location, identification, preference and access permission in specific context, whereas the current context contains other information except those belonging to user context. An example is shown as Figure 4. Before using Bayesian networks, we employ different policies to quantify the collected information. For instance, we divide the LightStatus into five levels. Then LightStatus "1: dim" in Figure 4 can be turned into probabilistic value 0.2. From a series of context, we are able to infer that Tom is presenting pervasive computing in the meeting room at 16:30 P.M.

Bayesian networks offers the benefits of creating the structures dynamically, depending on data set collected from various contexts.

However, three issues should be considered when using Bayesian networks reasoning. First, the dynamism of structure creating of Bayesian networks require all contexts should be quantized. However, quantization of context is non-trivial, because there are so many diverse context in pervasive computing. iShadow is working on the quantization of contexts, proposing and examining some general policies to map context.

Another issue taken into consideration is the calculation of Bayesian networks, which increase exponentially [11, 15, 12]. Let $C_i$ represent the predictive context, i.e., high-level context or action, $H$ denote the person and $W$ depict the context pertaining to the workplace where the person works or lives. $H$ and $W$ consist of many sub contexts. According to Bayes rule, $P(C_i | H, W)$ is calculated as:

$$P(C_i | H, W) = \frac{P(H|C_i)P(W|C_i)P(C_i)}{\sum_{C} P(H|C)P(W|C)P(C)}$$

When the size of Bayesian network becomes large, calculating Eq. 3 becomes a bottleneck. To solve this problem, we are investigating and comparing different approaches for close estimation instead of direct calculation.

The third issue is user preference. Defining user preference globally is a challenging task, because user preference varies significantly with individuals. We use log-based algorithms and model-based algorithms to obtain user preferences. Log-based algorithms record user behaviors, including visited web sites, usage of softwares, then find out user preference utilizing data mining algorithms. Given that log-based methods may not find complete user preference, we utilize recommendations from those recommendation by friends, colleagues and family members. Specifically, we borrow model-based algorithms [26] to look for the preference that users may like through building user-item matrix and predicting user preference.
6. Case Study

The goal of our case study is to demonstrate the effectiveness of our iShadow architecture. Particularly, we try to achieve the following goals: 1) Build a smart, pervasive computing platform, which offers fundamental functions to diverse applications. 2) Provide customized user experience. User personalizes the services according to his preference, which embodies the philosophy that pervasive computing is human-centric. 3) Implement a simple prototype of intelligent shadow. By utilizing the basic services provided by iShadow, the system is capable of unobtrusively tracking the user and adjusting.

To achieve these goals, we have implemented a prototype called iCampus for typical campus scenarios. Figure 5 illustrates the iCampus architecture and three scenarios.

(a) iCampus architecture
(b) Three scenarios of our campus, covering about 2,843,421 m²

Figure 5. iCampus architecture and scenarios

The user can access iCampus networks by PC, PDA, cellphone or laptop across the campus. The prototype includes various services such as map service, photo sharing service, search service and daily work assisting service.

- **Map Service.** It is deployed in our campus, as an application of campus guide. When a user visits the campus, the map service guides the user to navigate our campus. Map service varies with the user identification. If the user is a visitor to our campus, the map service will give detailed direction and show some introduction related the user’s location. For the users who often go to the university, the map service just shows some scenery pictures and routes that the users are traversing.

- **Photo sharing service.** After a user takes some beautiful photos of the campus building, he is able to upload them to the photo sharing service, which facilitates the sharing of photos and the construction of a social network.

- **Search service.** Because of the access control and other security techniques, abundant resource of our university can’t be indexed by commercial search engines such as Google, Yahoo and Baidu. However, school resource is valuable to both students and faculties. We implemented a small-scale search engine, which only indexes the school resource. Our search engine can search according to the user preference like profession and research interest.

- **Daily work assisting service.** This is an interesting service, designed a user’s daily work. When approaching his office or lab, the user wakes up his PC by handheld devices, such as PDA or cellphone. Then, his PC will check his most used email box, update his subscribed morning paper, synchronize the calendar among PDA, PC itself and Google calendar, and notify the user the new email and appointment. When time is approaching out of work, it notifies the user the weather or the bus according to the real time situation.

We created an intelligent shadow for each user and utilized the shadow to track the user. When a user moves into or leaves a scenario, resource discovery module will become aware of the change in λ period and notify the system, which makes some adaptation correspondingly. We extract the user preferences and current contexts from map service, photo sharing service, calendar information and the user search behavior. Then, we use ontology or Bayesian networks to infer high-level contexts. Finally, the system will give the user recommendations or execute actions. As a result, we hide the computing and greatly improve user experience. Figure 6(a), 6(b), 6(c) and 6(d) are snapshots of map service, photo sharing service, search service, and notification service, respectively.

7. Conclusion

In this paper, we have discussed iShadow architecture for building applications of pervasive computing. Compared with existing architectures, iShadow is characterized by lightweight user shadow model, scalable resource discovery and potent context inference mechanism.

The design and implementation of this architecture is an ongoing effort. While some progress has been made, there are still a number of issues to be addressed. One is the self-adaptation of user-shadow. When the context varies, current user-shadow may be unable to adapt automatically, because the context awareness devices may not collect adequate context information. Another issue is service composition and decomposition for applications. We are studying ways for automated service composition for users.
8. Acknowledgements

We would like to thank Gongwei Zhang, Lincun Cao, Wei Shi, Hao Zhou and Tiankun Zhu for their contributions to the iCampus project.

This work is supported in part by 863 Program of China (Grant No. 2006AA01Z172, 2006AA01Z199 and 2008AA01Z106), National Natural Science Foundation of China (Grant No. 60533040, 60725208, 90612018 and 60773089), and Shanghai Pujiang Program (Grant No. 07pj14049).

References


