A Formal Framework for Community Computing

Byung Y. Sung, Mohan Kumar, Behrooz Shirazi, and Swaroop Kalasapur

Department of Computer Science and Engineering,
University of Texas at Arlington,
P.O. Box 19015, Arlington, TX 76019-0015, USA
{sung, kumar, shirazi, kalasa}@cse.uta.edu

March 14, 2003

Abstract

In this paper we introduce a novel framework for pervasive computing called PICO (Pervasive Information Community Organization). PICO allows distributed computing and communication capabilities at higher and lower levels of abstraction, resulting in a significant improvement in current models of information systems for dynamic applications such as telemedicine, manufacturing, and crisis management. In PICO, sensors, devices, and actuators are used at low levels to collect environmental information and to influence the environment. Software agents working on behalf of the low level entities form logical communities to provide application-specific services. The major contribution of this paper is the introduction of the community computing concept as a framework for collaboration among software agents. In this paper we present formal models for representing physical devices, software agents, and computing communities. Additionally, we describe a prototype proof of concept implementation to validate the proposed framework.

1. Introduction

The proliferation of smart devices, embedded systems, and wearable computers yield to pervasive computing by extending the notion of distributed systems and mobile computing, such that computing environments saturated with computing and communication devices are invisible to human users, and can be used effectively. The realization of pervasive computing demands automated, continual unobtrusive services, and proactive real-time collaborations among devices and software agents in the environment. Current communication and collaboration models are essentially defined in an offline fashion and largely application-environment specific, and lead to visibility, inflexibility, and inefficiency in any alteration.

PICO (Pervasive Information Community Organization) is being defined as a framework for enhancing Pervasive Computing environments. A PICO application consists of a set of embedded hardware devices and sensors, called camileuns (context-aware, mobile,
intelligent, learned, ubiquitous nodes), and their associated software agents, called delegents (intelligent delegates). Any physical device that possesses a CPU, memory, communication ability and a subset of the above attributes can serve as a camileun. A delegent is a logical entity created by the programmer, user, application or another delegent. Delegents perform goal-oriented tasks on behalf of their associated camileuns and may also work in communities sharing information and resources with other sibling delegents. The major contribution of the PICO project is the introduction of a novel concept, called community computing, and using it as a framework for collaboration among delegents. The community computing concept of PICO provides the necessary platform to enable effective communication and collaboration among heterogeneous hardware and software entities. Pervasive computing challenges being addressed in the PICO project include: (i) handling dynamically changing information, (ii) adapting to changing situations, and (iii) providing scalability in terms of number of users, devices, and data sizes.

In section 2, the research related pervasive computing is presented. In section 3, we describe PICO framework with extensive discussion on camileun, delegent and their formal models. The layered architecture of PICO is also presented. In section 4, we describe PICO community and explain how it is formed and operated. In section 5, we present Parking Service Community prototype as a proof of concept and conclude the paper in 6.

2. Pervasive Computing Research Initiatives

The pervasive computing idea was initially put forward by Mark Weiser [1]. Recently, pervasive computing has been an active area of research attracting substantial attention by several research groups. In situated computing [2], situations of use and context play a central role in the use of computers. Interfacing mobile devices and environment-based applications are the main themes in situated computing and sentient computing [3] paradigms which are both dependent on sensory inputs from the environment. The concept of sentient computing is to create an interface that extends throughout the environment. The contextual computing paradigm [4] at Georgia Tech employs video inputs for location tracking and management. The Oxygen project [5] at MIT focuses on eight environment-enabled technologies: H21, E21, N21 and five others aimed at improving the user experiences such as speech, intelligent knowledge access, collaboration, automation of everyday tasks and adaptation of machines to the user needs. The Oxygen project is a large all encompassing research initiative whose current emphasis is on improving human-computer interface, PICO’s objective is to enhance the existing, as well as future, Internet services to improve human quality of life through pervasive computing.

The Portalano project [6] at the University of Washington seeks to create a test-bed for investigation into the emerging field of Invisible Computing. Portalano’s emphasis is on data-centric routing to facilitate automatic ‘smart’ data migration among applications rather than Internet service enhancement. The Aura project at CMU is another exciting pervasive computing project which encompasses techniques and concepts such as task-driven computing, resource exploitation, the use of agents [7], and ubiquitous service provisioning. Aura is based on well-proven legacy systems such as Odessey [8] and Coda [9]. Researchers at IBM are developing a model for writing application front-ends for their
Platform-Independent Model for Applications (PIMA) project [10]. The PIMA environment allows a developer to create platform-independent front-ends to multiple heterogeneous device platforms. In the Endeavour project at the University of California Berkeley, scalable adaptable information utility is the key issue [11]. Nomadic data access is facilitated by pervasive storage of information in the net to enable users to acquire information anywhere anytime. Unlike proactive delegent communities in PICO, the components of Endeavor react to changes in the environment.

PICO’s uniqueness, simplicity and versatility are derived from proactive community computing and modular hardware and software tools that can be assembled dynamically and automatically to suit any application and/or user requirements. PICO differs from all the existing projects we are aware of. In PICO, delegents constantly learn about a person (device or the environment), work diligently to assist the person in every task in an unobtrusive way. Communities are dynamically formed around a user to provide needed services transparently.

3. The PICO Framework

A PICO application consists of a set of embedded hardware devices and sensors, called camileuns (context-aware, mobile, intelligent, learned, ubiquitous nodes), and their associated software agents, called delegents (intelligent delegates). Any physical device that possesses a CPU, memory, communication ability and a subset of the above attributes can serve as a camileun. A delegent is a logical entity created by the programmer, user, application or another delegent. Delegents perform goal-oriented tasks on behalf of their associated camileuns and may also work in communities sharing information and resources with other sibling delegents. In this section, we will describe the details of camileun and delegent.

3.1. Camileun

Pervasive computing deals with many devices with different functionality to provide a service to a user. Not much work has been done in this direction, but it is necessary to capture the hardware and software components of a device and their characteristic in a systemic manner. The captured information could be used by any other components in the pervasive applications. In PICO framework, a camileun is the one that captures the functional entities of a device. A camileun is an abstracted logical representation of a device and provide a link between a device and delegent(s). A camileun describe a device by the tuple of $C = \langle C_{id}, F, H \rangle$ where $C_{id}$ is the camileun identifier, $F$ is the set of functionalities, and $H$ is the set of system characteristics. The set of functionalities of a camileun, $F$, classifies devices into sensors, effectors, processors and communicators.

- **Sensor**: A sensor receives an input and generates an electronic signal (information) to be consumed by other devices, such as effectors, processors, communicators. The examples of sensors are physical sensors, keyboards, mice, etc.
- **Effector**: An effector is a device used to produce a desired change in an object (environment) in response to input from a sensor, processor, or communicator. The
examples of effectors are displays, disks, mechanical actuators, etc.

- **Processor**: A processor is a hardware component and/or a software program that translates/manipulates an input into a required form. The hardware components of this category include micro-processor units in a device, and the software components include driver modules and s/w applications installed on devices.
- **Communicator**: A communicator is a hardware component and/or a software program that transmits or receives information to/from other devices. This category includes physical network device such as wired/wireless LAN devices and their accompanying software.

![General schematic diagram of a functional entity.](image)

While the above four categories define the nature of functions a device possess, the set $H$ represents system characteristics and describes system aspects of a functional entity, such as type of input/output, and characteristics of internal operation. In order to explain the $H$, let us look at the generalized schematics shown in figure 1. Each functional entity requires certain type of input, processes it, and produces output. For example, a contact sensor gets electric current as an input, processes it and produces binary data as an output. To fully describe a functional entity (sensor, effector, processor, and communicator), we need to define types of input/output and characteristics of internal logic. The input and output of a functional entity is defined in a form accepted and generated by the device. The form can be in an analog electrical signal, a boolean, a text, a image, or an audio/video sequence, etc. An example of an effector is an automatic door lock that receives a binary input, analyzes the input to see if it is valid, and send a signal to the physical door to open or close depending on the input. The characteristics of internal logic includes processing speed (CPU power), amount of memory, and I/O capabilities, etc.

The concept of camileun and its formal representation provide a base for building delegents described in the section 3.2, and facilitate a system designer a way of systematic modeling of a device.

### 3.2. Delegent

Camileuns are the abstracted representation of devices in PICO framework. It includes meta-data described in section 3.1. A delegent is an intelligent delegate that works diligently on behalf of a camileun or a user. A delegent provides encapsulation, interface, delegation, adaptation and manageability for the camileun it is associated with. The delegent responds to sensory inputs, events in the community and events within itself, and takes appropriate actions based on a set of rules. Delegents work in a community environment where they interact with other delegents and their environments.
A delegent *encapsulates* one or more functional units of a camileun. A simple delegent encapsulates only one functional unit, whereas a complex delegent encapsulates two or more functional units. The details of simple and complex delegents will be described in the section 3.3. Note that a complex delegent is not allowed to take functional units from multiple camileuns. Intuitively, it will cause the delegent unnecessary burden and complexity of maintaining and utilizing information on multiple camileuns. Further, the concept of community provides a way of such encapsulations among multiple camileuns.

Delegents *delegate* one or more functionalities of their associated camileuns to one or more communities. A delegent represents the camileun to a community where camileun function(s) are required for community operation. It is this attribute that makes camileuns adapt to different situations much like real chameleons.

A delegent provides a common *interface* to communicate or collaborate with other delegents. Community is a collection of one or more delegents working together towards a common goal. Communication and collaboration are one of essential to operations of community.

Delegents make camileuns not only *adaptive* to their surrounding environments, but also *condition* them to overcome uneven smartness of various collaborating camileuns. When a camileun is in the environment with poor network resource, the delegent for the camileun will initiate collaborative adaptivity, which requires help from other delegents to cache the network packets destined to it.

3.2.1. Modeling of delegents

For modeling purpose, a delegent is described as \( D = d_{id}, M, R, S \). \( d_{id} \) is the identifier, \( M \) is the set of modules, \( R \) is the set of operational rules and \( S \) is the set of services provided by the delegent.

\( M \) can be formally defined as \( M = \{ m_i \mid m_i = \mu(f_i), f_i \in F \} \) where \( m_i \) is a module, \( \mu \) represents the implementation of \( f_i \), and \( F \) is the set of functionalities of the camileun associated with the delegent. \( M \) consists of required modules to carry out \( S \) and the modules are derived from the \( F \) of the camileun it’s associated with.

\( R \) is the set of operational rules and \( S \) is the set of services [todo:should be one service] provided by the delegent. The \( R \) defines how a delegent responds to the events when it is in a certain state. The states a delegent can be in are *sleep, dormant, active, mobile, or states* specific to the service of the delegent. Each of the states associates with an event such as *wakeup, activate, roam, return, wait/fail, and flush*. The figure 2 shows how transitions among these states take place according to an event.
3.3. Relationship between $S$, $M$ of delegent, and $F$ of camileun

Camileuns posses one or more functionalities and delegents delegate the functionalities to PICO communities. In this section, we describe how to map camileuns’ functionalities to delegents.

3.3.1. One-to-one Mapping

The simplest way of mapping is the one-to-one mapping. One of camileun’s function is mapped to a single delegent. In other words, the delegent, called a simple delegent, is associated with only one of camileun’s function. The advantage of this mapping is that 1) it is easy to maintain camileun’s resources unlike the complex delegent which will be discussed later, and 2) it is easy for the delegent to migrate to another camileun since it requires only one function from that camileun.

The disadvantage of this mapping is that the number of delegents is proportional to the number of functions a camileun can carry out. The number of camileun’s functions depends on how each function of the camileun is defined. The number of camileun’s function
becomes unnecessarily large if the function is defined too finely. This will result in large number of delegents which in turn cause unnecessary duplication of delegent and increase overhead on the camileun. On the other hand, the number of camileun’s function is unfairly small if the function is defined in a coarse way. The coarse functional definition of a camileun causes low utilization of a camileun’s resources because of unnecessary sharing between two or more delegents. If a camileun function includes one or more sub-functions, the delegent is forced to include all of the sub-functions of a camileun function regardless of the usefulness of the sub-functions. Some of these issues are the basic problem of system decomposition, which is beyond the scope of this paper [reference(system decomposition)].

3.3.2. One-to-many Mapping

In this mapping, a delegent, called a complex delegent, is mapped to two or more functional units of a camileun. The main purpose of having this mapping is to provide delegent level service composition. Complex delegents compose services by combining a set of functional units from a camileun. They are able to carry out the compound tasks whereas simple delegents are limited to functionalities they are associated with. Complex delegents use one or more functional units of camileuns. Sets of functional units that the delegent uses can either overlap or be disjoint. Note that mapped functional units for a complex delegent are from only one camileun. We are not allowing a complex delegent to take functional units from multiple camileuns. This inter-camileun mapping causes difficulties in managing camileun resources. Further, this can be supported by our concept of community.

3.4. Layered Architecture of PICO

PICO is made up of five layers as shown in figure 4. The camileun (physical) layer consists of the computing and communication hardware, communication network, operating system, and associated drivers. The next layer comprises PICO-compliance software for delegent-camileun interface for adaptation of existing hardware devices to the PICO environment. This layer also consists of communication APIs and modules, and location-, energy-aware mechanisms. Recently, a plethora of wireless devices have been deployed in a wide variety of situations: pocket PCs, cell phones, 2-way pagers, Bluetooth devices, 3G multimedia terminals, and many others. Most of these devices have processors and their focus is on providing processing support to perform single functions. These devices can be adapted for use in heterogeneous PICO environments. Delegent supporting layer responsible for creation of delegents by dispatching appropriate delegent, managerial function including service discovery, support mobility of native and foreign delegents, migration, authentication for foreign delegents to execute on the camileun. This layer is also coordinating delegent communication by doing network address translation for delegent, binding delegent address with it’s native camileun. The delegent layer (DL) comprises of delegents created for carrying out various tasks and serve in communities. At the community layer, the creation, merger and disintegration of communities take place. The functionality, mission, interest, and goals of a community are also defined in the community layer. All layers except the lowest layer are also responsible for providing mobility services, resource discovery,
authentication, QoS, and resource management activities such as network monitoring, congestion control, and energy saving.

![Layered Architecture of PICO](image)

**Figure 4. Layered Architecture of PICO**

## 4. Community

A camileun is an abstraction of a device’s capabilities, and a delegent provides software encapsulation of a device. A community in PICO is defined as an entity consisting of one or more delegents working towards achieving a common goal. It provides a framework for collaboration and coordination among delegents.

For modeling purpose, a community is described as $\langle p_{id}, U, G, E \rangle$. $\langle p_{id} \rangle$ is the identifier, where $P$ is the set of delegents that are members of the community, $G$ is the set of community goals, and $E$ represents community characteristics. The $E$ includes information of the leader of community and any other community management and operation related factors such as the rate of periodic vital message exchange, servicing model etc.

During the life of the community, a community is formed, is in the operation and maintenance phase, and is dismantled. While in operation phase, the community carries out its goal, serves its service and maintains its members consistently.

The community creational model explains how a community is formed; the community service model is describe how the members of community serve a user to accomplish the goal of a community and Community management and operation will be discussed in the next sections.

### 4.1. Community creational model

A community in PICO is created either statically or dynamically. The former is called *service provider community* and the later is called *dynamic community*. Service provider communities are created to provision services in various applications. The mission or goal of service provider communities and the rules for creation of such communities are pre-defined. Dynamic communities are created in response to exceptions. Thus, the goal of the community is a result of a series of events. The scope of this paper is limited to service provider community.

The goal of a community plays critical role on selecting member(s). To fulfill the goal, a
community need to have at least one or more delegents that possess functionalities required by the community. A community can have one of kind delegents to provide the same benefit of clusters of resources. The leader of community coordinates the activities of community such as scheduling and planning of resources.

4.2. Community management

Community management includes a mechanism for community creation, and membership management. Service provider community is statically created, but the members of the community are dynamically changing during the operation. Community is composed of one or more delegents which are working toward goal of the community. Typically, the leader of a community is residing on relatively powerful system such as home or office server. A community leader is responsible for community membership management and coordination of community tasks. Since the goal of service provider community is defined a priori and created statically, it is safe to assume that the leader of a service provider community is also predefined.

The core of community membership management is community membership maintenance (CMM) protocol. The CMM protocol basically does two things, 1) it allows a delegent to voluntarily join or leave the community and 2) it maintains the status information of member delegents in a community. Like any other group management protocols, the CMM reflects community membership changes accurately and in a timely manner as they happen. The inaccurate member status information of community will lead to delay or failure of community service.

5. A Prototype of PICO

As a proof of the PICO community concept, we now describe a prototype, built to serve drivers find a parking space in places like downtowns, shopping malls and universities. We call the resulting service as Parking service community. The prototype assumes the existence of a pervasive infrastructure, wherein we have abundant computing devices embedded into our environment. The drivers of the vehicles are equipped with devices like GPS systems, cell phones, PDAs, etc, with communication capabilities. The infrastructure itself has the capability of identifying the user in its vicinity and authenticating that particular user. Our premises are not very unrealistic considering the kind of devices flooding the market today.

The Parking service community has been implemented as a service provider community. It consists of parking service and map service delegents. The leader of the Parking service community is well known to the world. The leader maintains membership and status of the community members. The delegents residing on the user’s devices are aware of this leader’s presence. Also, the delegents monitoring different parking lots have information about this leader. Figure 5.1 shows the community structure of the prototype. There are a number of sensors in each of the parking lots, monitoring each parking space. These sensors are monitored by the delegent responsible of that particular parking lot ($D_{member}$).

When an active user comes into the vicinity of the parking service, the parking information is provided in a proactive manner. This information contains the names of different parking lots
and the number of vacant spaces in them. When the user decides to park, s/he has an option of either directly reserving a space at a parking lot of choice, or requesting the service for the nearest parking space. The service returns the result with directions to the nearest parking lot, with a vacant space. Depending on the capabilities of the user device, the result is either an image or directions to the parking lot in text, both from the current location of the user.

We now will give the formal notations for the community components. These notations, the process of extracting or abstracting them is a gradual process, and we are currently experimenting with XML and other formal ways of representing them. Nevertheless, the following general representation does justice to the abstraction and serves the purpose.

5.1. Parking Service Community

The parking service community composes of leader, map, members, and user delegents. The formal representation of the community is as follows.

\[ P = \langle UTA_{\text{Parking}}, \{ D_{\text{leader}}, D_{\text{map}}, D_{\text{member}}, D_{\text{user}} \}, \text{Parking}_\text{Service}, \{ D_{\text{leader}}, \text{update\_rate} = 10/min \} \rangle \]

As we discussed earlier, the leader is in charge of community maintenance and operation; map delegent is responsible of generating directions and map of the location; the member delegent represents individual parking lots; and user delegent represents user. The membership of user delegent in the community is on demand.

5.2. Camileuns of Parking Service Community

We have made use of four types of devices to build the prototype which are abstracted as camileuns. The sensors collecting status of parking lots, the computers on which the delegents in the community run and the user device through which s/he makes the request. Considering one by one, they are represented as follows

- \( C_1 = \langle \text{sensor}_{(n,m)}, \{\text{sensor}\}, \{\text{in}=\text{on/off}, \text{out}=\text{boolean}, \text{rate}=10/\text{min}\} \rangle \)
- \( C_2 = \langle \text{parking\_computer}_{m}, \{\text{communicator, processor}\}, \{\text{in/out}=[\text{IR, Bluetooth, TCP/IP}], \text{collect\_sensor\_status}\} \rangle \)
- \( C_3 = \langle \text{server}, \{\text{communicator, processor}\}, \{\text{in/out}=[\text{IR, Bluetooth, TCP/IP}]\} \rangle \)
- \( C_4 = \langle \text{Jim’s PDA}, \{\text{sensor, communicator, Effector}\}, \{[\text{Touchscreen}, [\text{in/out}=[\text{IR, Bluetooth, 802.11b}], \text{HTML Display}]\} \rangle \)

5.3. Delegents of Parking Service Community

The delegents used in building the parking service prototype are of mainly three types. \( D_{\text{member}} \) maintains information about a parking lot. \( D_{\text{leader}} \) coordinates with all the members and also acts as a contact point for the service. \( D_{\text{map}} \) is very similar to \( D_{\text{leader}} \) in its structure, but provides a different service. The delegent \( D_{\text{user}} \) resides on the user device and has the capability to put forward the request to \( D_{\text{leader}} \). These delegents can be represented as
Since this is a service provider community, all the components are under the control of the designer. The above mentioned tuples guide the designer in building these components. Another interesting observation that we have developed is the way in which the operational rules for the delegents can be generated. While designing the delegents, we can extract these rules from the state diagram for the delegent. Figure 5.2 gives an example of the process, which maps on to the operational rules for \( D_{\text{leader}} \). The same process is adapted to fill all other delegent rules.

6. Conclusions

In this paper, we have introduced PICO as a novel framework for pervasive computing. Camileuns capture functionalities of devices in a systemic manner. Delegents represent camileuns or users in pervasive computing environment and community provide a way of collaboration and coordination. With these components, PICO provides a framework for realizing pervasive computing.

Reference


