Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 1481–1488

DOI: 10.15439/2015F49
ACSIS, Vol. 5

A Platform for Context-Aware Application Development: PCAD

Ufuk Celikkan
Izmir University of Economics
Dept. of Software Engineering
Sakarya Cad. 156 35330
Izmir, Turkey
Email: ufuk.celikkan@ieu.edu.tr

Kaan Kurtel
Izmir University of Economics
Dept. of Software Engineering
Sakarya Cad. 156 35330
Izmir, Turkey
Email: kaan.kurtel@ieu.edu.tr

***Abstract*— We propose a novel software platform based on the notion of context-awareness which allows rapid and easy development of context aware applications. One of the fundamental goals of the proposed platform is extensibility, allowing the platform to react to new requirements without making fundamental and substantial changes. The Context Aware Application Development Platform- PCAD- has inspired from an operating system and modeled as a layered architecture. It exhibits a plug-and-play behavior very similar to devices and device drivers found on an operating system/kernel. The platform offers functions such as data management, notification, security and privacy as services. The platform provides a public interface to application developers in their development of context-aware applications. It covers a wide range of disparate application domains such as smart city and livestock monitoring applications.**

I. Introduction

APPLICATIONS using information and communication technologies collect and process a diverse range of data using machines connected through communication networks. This phenomenon is captured in the term the *Internet of Things*. The Internet of Things is a complex interconnection of heterogeneous devices that include sensors, cameras, micro chips and RFID based products, and which generate large amount of data obtained from various domains. For this reason platforms or architectures must be designed in anticipation of the increase in the number of devices, and the varying demands of users and applications in different contexts. In order for a platform to be successful, it is imperative that it recognizes the context in which users and applications are operating, and enable service customization for a particular user. The creation of smart applications and environments then becomes a possibility by using context-aware computing, which acquires, analyzes, and interprets relevant context information, and responds to contextual changes.

Temperature, humidity, traffic congestion, road conditions, sea pollution, and river level are some examples of context information. Such context information can be used alone or in combination within context-aware applications to provide custom services in various domains, such as transportation, health and medical systems, tracking and control of environment, energy, agriculture, industry, sport events, and tourism. However, the effective use of this context information requires its efficient and effective acquisition, storage, processing and reasoning. In this way, productivity, economic output and quality of life can be increased.

This paper describes a novel, service-based software platform proposal based on the notion of context-awareness. The platform basically follows a middleware approach, which draws on the techniques taken from operating system design. The primary goal of the proposal is to offer a platform to simplify the development of context-aware applications by relieving the applications from complex context data management issues. In an open, dynamic and continuously changing environment, context data must be acquired, managed and ultimately offered to applications which will interpret the data according to the situation. The platform separates context acquisition from the application code and handles many context data management issues on behalf of the applications. The fundamental design force of the proposal is that the platform is agile, robust, and capable of reacting to new requirements without the need for fundamental and substantial changes.

The rest of the paper is organized as follows. The motivation section explains the rationale and design issues that motivated this work. Related works section includes an overview of the existing systems based on context-aware features. Section 4 presents the technical architecture of the platform. Section 5 gives a practical scenario from a smart city parking application, and finally, Section 6 draws conclusions about the proposed system.

II. Motivation

Dey et al. has listed three characteristics of a context aware system as (i) information and services are presented to a user, (ii) a service is executed, and (iii) context is linked to information for later retrieval [1], [2]. Therefore, a service

This work is supported by the Scientific and Technological Research Council of Turkey – TUBITAK under Grant 114E938.

based software infrastructure support facilitates building context aware systems as it simplifies system creation by placing common context aware computing features in the infrastructure as services. A fundamental advantage of a service-based infrastructure approach is that the resources (i.e. data, sensors, devices, services) are shared, making maintenance and evolution of the infrastructure much easier [3]. Such an infrastructure is best realized in the form of a middleware, in which the infrastructure acts as a foundation on which other systems can be built. It takes responsibility for acquisition, storage, processing and interoperability of the context data, and provides services such as alarm, notification and security to the applications.

An architecture used in building the service-based infrastructure must possess certain attributes in order to be a viable option in creating context aware systems. Notable among those attributes are extensibility, robustness, scalability, and ease of use. Extensible architectures allow the addition or removal of components without breaking or taking down the system, and facilitate the seamless deployment of new sensors, devices and services. The entire system can evolve in a step-wise fashion. Robustness requires that the infrastructure operates in the case of a malfunction and terminates peacefully when breakage is unavoidable, while scalability ensures that infrastructure is able to serve many users and collect data from a diverse range and number of sensors. The amount of administrative effort must also be minimized to maintain the efficiency of services, devices and sensors. Finally, no matter how well the architecture is designed, it must be efficient and simple to use. Efficiency is measured in terms of time and space, which are influenced by the data storage, response and real time requirements of the system.

An Operating Systems inspired approach is the most suitable proposition in designing context aware infrastructure architectures. The primary aspect of Operating System architectures is the abstraction of inner details through layering and providing a high-level interface to allow uniform access to the functions of the system contained in its layers. Two typical examples from the UNIX world are the POSIX and device driver interfaces. POSIX interface allows the applications interact with OS services such as file system, network stack and devices, and device driver interface allows the seamless integration of new devices and drivers to OS kernel.

For the reasons mentioned above, a service based middleware architecture inspired by an OS is used in the design of PCAD system. The sensors, devices, services and more importantly, the applications are separated from each other, yet interact with one another. The loose coupling of components permits plugability and reduces fragility, the two essential characteristics of an extensible and robust system.

A prototype of the platform is currently under implementation together with two sample applications that utilize the platform. One of these is a livestock monitoring application that monitors livestock physiological data and notifies the farmer and the veterinary physician of potential livestock health problems. The second application is an implementation of a smart city parking scenario described in Section 5. Both applications will allow the evaluation of the effectiveness, robustness and performance of the platform applied to diverse domains. It will also demonstrate that the platform facilitates rapid context-aware application development.

Despite being an important issue in context aware systems, we have chosen not to address context modeling. Context is defined by [2] as: “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves”. A piece of information is considered context data only if it is interpreted, otherwise, it is simply information belonging to an environment [4]. For applications and services to adapt their behavior based on the particular context, they need to be able to do reasoning. The method chosen for modeling the context has direct influence on the kind of reasoning performed [5], [6]. Therefore it is very important to choose a modeling technique that will facilitate not only easy reasoning, but also sharing of context with others. Since our primary focus is to define the platform itself and the interactions among the components, we defer the discussion of context modeling for our future work.

III. Related Work

Many context aware architectures proposed in the past have borrowed ideas and features from operating system designs. Gaia project [7] describes one such architecture which is heavily influenced by operating system design. It extends the OS concepts by incorporating context-awareness. It has a context file system that organizes data by context and a Unix-like signal event mechanism to create channels to connect context data suppliers to consumers through a handle called *ContextType*. Another architecture influenced by operating system design is Context Toolkit in which a widget model is adapted resembling the device driver model of an OS. In the widget model, widgets are software components that provide an interface to a hardware sensor. In other words a widget acts like a device driver to a sensor. Context Broker Architecture (CoBrA) [8] employs an intelligent context broker to provide a centralized context model shared by devices, agents and services. One of the responsibilities of the context broker is to ensure that user privacy is protected through policies by allowing the user to control how contextual information is shared. The issue of how to control access to shared information is of key importance in operating systems.

There are other architectures based on layered architecture model which have the goal of abstracting sensor data management from users. SOCAM, Service Oriented Context

Aware Middleware [9], project is a middleware architecture that supports the building of context aware services in pervasive environments. Context aware services, which may also be applications, can obtain context using the context middleware layer from context providers. As expected from a middleware-based architecture, in SOCAM, context aware application code is separated from context acquisition and reasoning. Another example of a centralized middleware approach is found in CASS project [10], where the middleware itself is a server responding to context management requests of mobile context aware applications. Context acquisition is done by the sensor nodes and context data is sent to the server. Storage, modeling and reasoning tasks are performed by the middleware server. Consequently CASS applications neither need to communicate with the context source directly, nor are they affected by context-based inferences and behaviors. Another layered architecture tailored towards mobile devices that addresses the shortcomings and special needs of mobile devices is the Hydrogen [11] project. One of its three layers, the Management layer is responsible for storing context information and sharing it using XML protocol. A second layer, the Adapter layer, delivers information obtained from context sources, i.e. sensors, to the Management layer. The Application layer, where the application code resides, can access context data synchronously or asynchronously.

A common benefit of these architectures is their capability to prevent coalescing of application code with the context management functions by decoupling context sensing and acquisition from the application code.

In addition to architectures and frameworks several context aware systems for different domains have been built or proposed in the literature. Context aware systems for pervasive environments are surveyed in [12] and web services based systems are studied in [13] and a classification of context aware systems is given in [14].

IV. PCAD Architecture

Context-Aware Application Development Platform-PCAD- has been inspired by operating systems design and modeled as a layered architecture. PCAD's primary goal is to offer extensible, scalable, robust, secure and general purpose software architecture for use by developers in several industrial processes and sectors. The platform particularly addresses the following design issues:

1) Extensibility
2) Security and Privacy
3) Simplicity (be applicable and pragmatic)
4) Generic (not domain specific)
5) Service Based

The platform itself adapts a middleware and blackboard [4] model, providing common services to the applications. This is similar to the conceptual separation of user and kernel services in an OS. Among the services provided by PCAD are data storage, alarm, notification, a simple rule engine and reporting facilities. One can view the platform as a layer providing kernel-like services, while the applications provide user level services and the physical sensors and their software act like devices and device drivers providing data to applications. The applications subscribe themselves to the sensors through the platform, after which they are notified by the context providers when a context data becomes available. The applications and the sensor software are bound to the platform using standard protocols (e.g., REST, web sockets, and raw socket). The architecture exhibits a plug-and-play behavior similar to the devices and their associated drivers found in operating system/kernel. The sensors can be added and removed, and applications can be registered to the newly added sensor without needing to modify the platform. The platform provides a public interface to enable the rapid and easy development of applications from various domains by application developers. It relieves the applications from implementing common services and functions, thus fulfilling one of the fundamental principles of software engineering – reusability.

A. PCAD Services

PCAD's functionality is made available to applications through services. The services of PCAD and its architectural resemblance to Operating System architecture is shown in Figure 1. The list of services available in PCAD is given in Table I and explained in detail below.

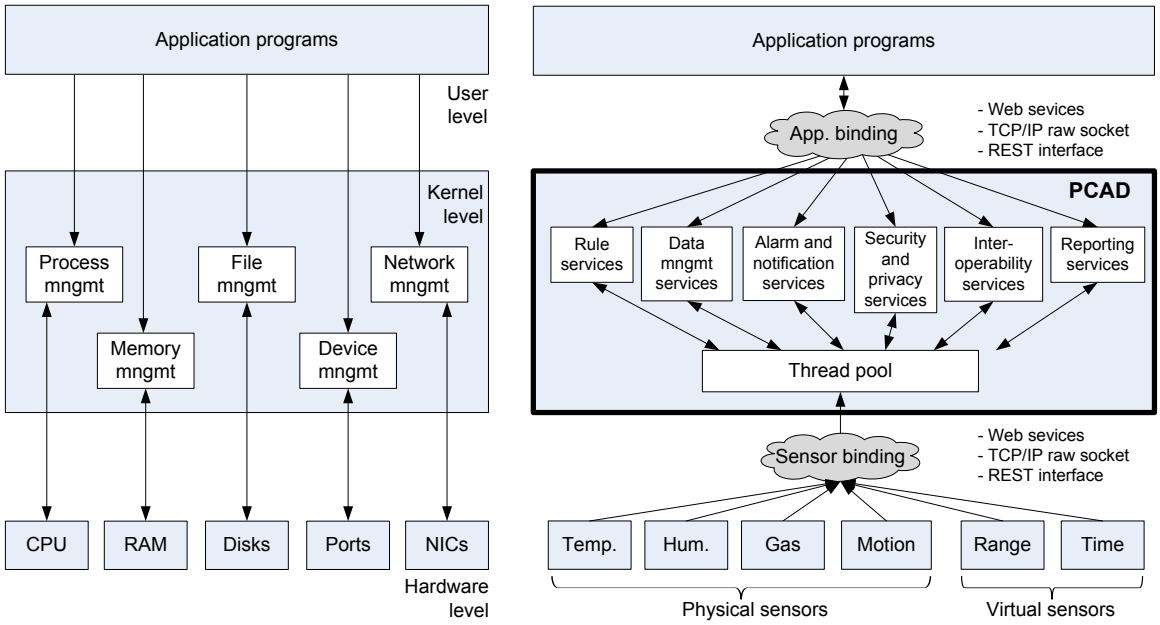
TABLE I.
PCAD SERVICES

Service name	Service description
Rule Service (RS)	Context processing
Data Management Service (DMS)	Stores context data and provides data upon request through a uniform interface. Underlying storage mechanism is transparent to the users.
Alarm and Notification Service (ANS)	Notifies the registered parties about context data. Filters data if necessary.
Reporting Services (RPS)	Generates detailed reports about context sources including data and status of the context source.
Security and Privacy Service (SPS)	Ensures only authorized parties can access to context data based on policies.
Interoperability and Communication Service (ICS)	Data exchange among similar platforms using standard communication protocols and message formats.

Rule Services

A very simple and generic syntax will be designed in the form of if-then rules. The rules will be specified in an XML file, and will be verified against the Rule XML Schema using XSD. A generic rule will be of the following format:

```
if <cond> then action
```



a) Generic OS Architecture [15] b) PCAD

Fig. 1 A Comparison of PCAD architecture and OS architecture

The condition in the rule can be a combination of conjuncts and disjuncts. When the condition evaluates to true, then the action is performed. This action could be further used to trigger other rules. Below, one such example is given:

```
if (raining and
    location="main street")
then congestion
```

When this rule is executed, firstly rain sensor data is retrieved, followed by location information. If the condition evaluates to true, which means that it is raining and the location name is *main street*, then the `congestion` rule is executed. Congestion rule simply retrieves the congestion data for the main street using the congestion sensor and returns it to the application.

The applications create a rule file in XML and hand over to PCAD for context processing. Alternatively, applications may perform their own context processing, thus opting out using the Rule service of PCAD.

Data Management Services

Data Management Services is responsible for storing of data received from multiple sources and making the data available to those requesting it. It provides a uniform interface for the transparent access of data by the applications. This interface let users of this service insert, update, delete and filter information. It is essential that the sensor data, whether collected synchronously or asynchronously, is stored without any loss. DMS could store data simply in a relational database, or it can use a cloud service. The underlying storage mechanism is transparent to the applications.

Alarm and Notifications Services

The Alarm and Notification Services (ANS) follow the paradigm of observer design pattern [16]. The applications attach themselves to a sensor of interest to acquire context information. There are two modes to query the presence of sensor data. In the asynchronous mode, ANS periodically queries the sensor and then reads and stores that data. It then immediately makes the data available to the applications that showed interest. In synchronous mode, the sensor software interrupts the ANS for a reading. Asynchronous mode requires support from the sensor software, which must notify ANS for data availability. This is generally not the conventional method of interacting with a sensor, and therefore is not available in most cases. The ANS provide sensor data to applications using either pull or push method. In the pull model, the applications receive only a minimal notification, and applications or users ask for specific details thereafter. In the push model, ANS sends applications detailed information about the sensor data.

A filtering mechanism will be employed to allow applications to register themselves to specific conditions so that they will be notified when such condition occurs. An example filter is shown in Figure 2. An application named *myApp* is only interested in data from the temperature sensor *temp001* between 9:00 AM and 12:00 PM, and only for values 39 and above.

Reporting Services

Detailed reports presenting sensor data and sensor availability shall be provided by the Reporting Services (RPS). Visualization of the sensor status and data provides a useful overview of the system, which is important because it is not easy to understand and interpret raw data. The data is presented to the user in different formats, such as tables, charts and text. It was decided to select open source reporting tools with an application programming interface. BIRT [17] is one such tool with several APIs to design, generate and render reports and charts.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration name="MyApp">
<Sensor name="TME3m" id="temp001" >
<TimeFilter start="09:00:00"end="12:00:00"
      onMatch="ACCEPT" onMismatch="DENY"/>
<ThresholdFilter value="39"
      onMatch="ACCEPT" onMismatch="DENY"/>
</Configuration>
```

Fig. 2 Example filter

Thread Pool

The platform will ensure that the context data received from the sensors are properly stored and sent to the applications in a timely manner. A pool of threads, as shown in Figure 3, are to be created and tasks are assigned to threads whenever needed. Each thread will have two sub threads; one is responsible for real time delivery of raw data to applications, and the other responsible for the storage and processing of the context data.

Real-time support

A certain class of applications imposes constraints on the sensor data delivery time. Context aware systems using direct sensor access approach are better suited for meeting the hard real time requirements. Such systems collect information directly from the sensor without any intervening components, thus, time lapse in reacting to the events conforms to the real-time requirements. However, a disadvantage of these systems is their inability to re-use components due to the tight coupling of sensor device drivers with the application [11]. In contrast, systems that are built using layered architectures are considered superior because they are extensible and allow modifications to the architecture without modifying the applications. However, meeting the hard real-time requirements is difficult in layered architectures, as data flows through several layers before it reaches the application. Therefore, layered architectures are much more suited to soft real-time requirements. Soft real-time systems can tolerate larger time latencies and they still operate even when time constraints are violated, albeit with degraded performance.

PCAD addresses the soft real-time requirement by establishing a direct pipe or channel between the sensor software and the applications, as depicted in Figure 4. A dedicated thread is allocated to the channel, allowing raw sensor data to be transmitted through this channel, while a copy of the data is processed and stored by the platform concurrently.

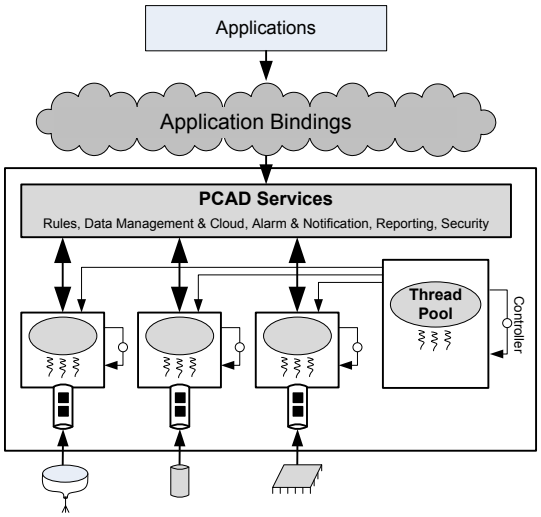


Fig. 3 Thread Pool in PCAD

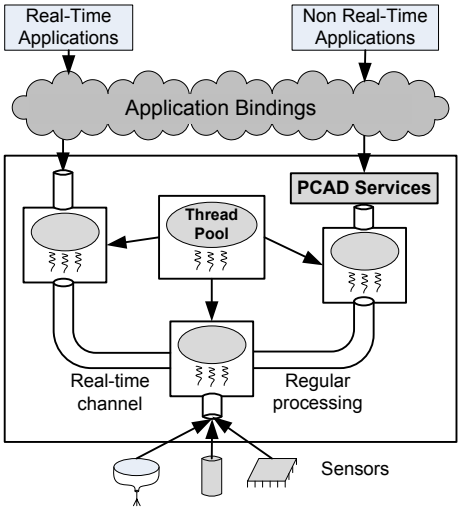


Fig. 4 Real-time support

Security and Privacy Services

Privacy is an important issue in PCAD as sensors will acquire information about the environment, and share it with different applications. As the sensor data is owned by the platform rather than the applications, it is the platform's responsibility to protect the privacy of the data.

A very basic Role Based Access Control (RBAC) [18] scheme mediates which sensor data is accessible to which application. An RBAC-like access control mechanism is preferred because it allows more complex policies to be accommodated in the future. The idea is to assign applications particular roles and assign access rights to roles, and then control which application can access sensor data based on those roles. Our use of RBAC will be much more

closely aligned with Mandatory Access Control mechanism, where the sensor data is assigned privacy levels, and the applications are given clearance levels. The set of roles will be defined by the platform based on policy configurations. For instance, a weather application does not need to access to the transportation data.

The three basic elements of access control, Subject, Object and Access Right have the following mappings in PCAD: the subject is the software processes in which the applications run. Object is our context resources. These context resources may be physical or virtual sensors. Access Right would be the read right of the context data, or querying the status of the sensor providing the data.

Tables II-A, II-B and II-C show an example of role based access control. In this example, the subject is the application and the object is the sensor. *Application1* has been assigned roles *role1* and *role2*, which gives application the right to read *sensor2* and *sensor3*, and to query the status of *sensor1*. On the other hand *application2* can only query the status of the three sensors. RBAC makes it easy to put constraints, for instance, we could limit the number of applications collecting data from a sensor by limiting the number of applications assigned to a role.

TABLE II.
A. AUTHORIZATION EXAMPLE

Authorization	Description
read	Allowed to acquire context data
query	Allowed to query the status of the context source

B. APPLICATION ROLE MAPPING

Subject	Roles		
	role 1	role 2	role 3
application 1	*	*	
application 2			*

C. ACCESS CONTROL IN PCAD

Roles	Sensors		
	sensor 1	**sensor 2**	**sensor 3**
role 1	query	read	read
role 2		read	
role 3	query	query	query

Two aspects of equal importance are the authenticity of the sensor, and protection of the confidentiality and integrity of the information it provides. The field of network and information security has sufficiently matured in the provision of security features to allow combination of symmetric and asymmetric cryptography (digital certificates) to fulfill the authentication, confidentiality and integrity requirements of the platform.

Interoperability and Communication Services

According to a recent survey, interoperability is listed as the number one quality driver for smart city applications [19]. Interoperability is a necessity since data is owned by other parties and managed by different platforms, similar to PCAD and also for making PCAD data available to other requestors. This creates a need for interoperability for exchanging context information among different applications. Interoperability and Communication Services are responsible for context data exchange using standard messaging format. As shown in Figure 5, PCAD uses XML as the data exchange format.

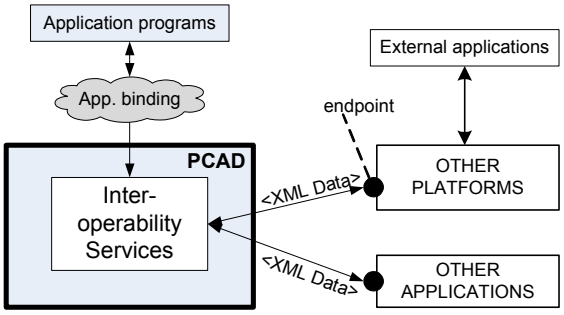


Fig. 5 Context data exchange

Applications built on top of the PCAD platform use this service to send context data to other applications. The applications can also ask the platform to receive context data from other platforms, including from another instance of the PCAD platform. In such a case, the applications must let the service know the endpoint information that has the data.

B. *Sensor Layer*

The sensor layer consists of physical and virtual sensors. Examples of physical sensors are pH, humidity and temperature. Unlike physical sensors, virtual sensors use either Application Programming Interfaces or web services to obtain data, such as GPS position or real-time camera stream [20]. In many situations, data can be made meaningful by aggregating individual context data obtained from multiple physical and virtual sensors. This process, called context aggregation, generates combined data that is more accurate and relevant than information obtained from a single source. Context aggregation function is implemented also as a virtual sensor.

The sensor software will interact with the platform by registering to the platform and sending raw data to the platform, which is further formatted and processed by the platform. Sensor layer interacts with the platform through sensor binding layer.

C. *Binding Layers*

For the purpose of allowing applications and sensor software to communicate with the platform, the platform contains two binding layers. The first is between the application software, and the platform and the second is between the sensor software and the platform. These binding layers are facades that simplify the application and the sensor software's interaction with the platform. The application

binding layer is used by the applications when they require the services of the platform, such as receiving context data, registering and deregistering for sensor data, creating reports and receiving alarms and notifications. The sensor binding layer is used by the platform to gather data from a sensor and deliver it to the registered applications, possibly multiple times. Therefore, it is possible for many applications to access the same context data simultaneously, which may not be possible with the direct access to sensor alternative.

The implementation of the binding layer employs widely used protocols. Among the protocols supported are web services, web sockets and REST.

V. A Scenario: Smart Parking

In this section, we present a scenario to demonstrate the interactions between an application PCAD and sensors. The scenario involves a context aware smart city application running on a mobile device developed for the purpose of finding a parking space based on certain constraints.

One day Mr. Jones went out of the house and boarded his car. Just he was about to approach to his workplace in downtown, he remembered that he wasted time searching for a parking place the previous week. He started the application on his mobile phone and received the coordinates and the route information about the nearest parking location within a 500 meter radius on a map. He thought, "I did a good job of using this context-aware [sic] application to access to the information provided by the smart [sic] city, otherwise I would have been late for my appointment again."

Following is the sequence of actions that take place in the application and PCAD platform:

1) User invokes the smart city context aware application on the mobile phone.
2) Smart city application starts a session with the PCAD platform.
   A communication session is opened using WebSockets. The application and PCAD now begin to communicate via the application binding layer for fulfilling user's specific request about an available parking place.
3) User requests route information about a parking place within the 500 meter radius of his workplace.
   The user's operational parameters, such as the 500 meter constraint, are translated into rules and registered with the Rule Services (RS) of CAS, using the session established before. These rules get executed by PCAD later on.
4) Smart city application sends user requests to PCAD via application binding layer.
5) PCAD periodically requests ground parking data from installed sensors, which includes sensor fingerprint (id, location, time) and parking availability.
   Since the actual parking status changes often due to usage conditions, ground sensors are asked to send parking availability information in real-time. The frequency of sensor query is a configurable parameter. The scenario is an example of soft real time application because the latency of getting availability information is not critical to the operation of the application. PCAD collect these data via the sensor binding layer and notifies the application via application binding layer using alarm and notification services (ANS). The real time channel is used to send the data to the application. PCAD enforces data privacy by only providing parking data to the application.
6) Smart city application determines the user's location.
7) Smart city application finds optimal parking space for the user.
   Smart city application acquires user's location information from mobile device, and parking availability and parking location information from PCAD. The application then offers an optimal path using this information together with other constraints, such as shortest path, traffic signs or number of traffic lights etc. The application either gets some of this constraint information from PCAD or PCAD can aggregate this information from other sources, including other platforms using the interoperability and communicating services (ICS). This part directly points all other smart city applications working together.
8) Smart city application presents the information including, coordinates and map to user.

This scenario shows how PCAD services are utilized by the application, and how the actions are performed by the actors of the system. Figure 6 shows the steps of interaction described above. The figure illustrates how sensor acquisition is decoupled from the application code.

This scenario can easily be adapted to other problems, such as finding a charging station for an electric car. A context-aware charge management system for electric vehicles is described in [21]. A more elaborate problem on parking lot management for charging stations is described in [22], while a parking reservation system based on telecommunication APIs is given in [23].

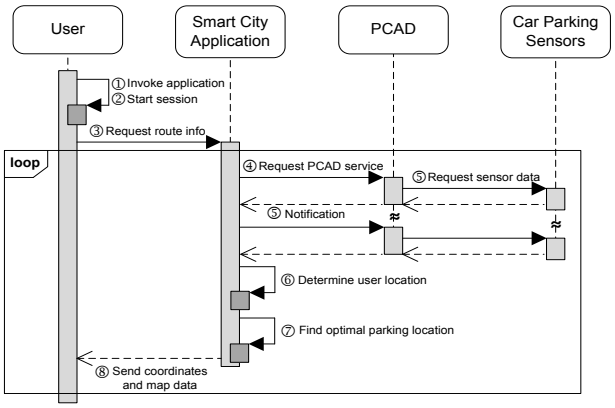


Fig. 6 Interaction diagram of the scenario

VI. Conclusions

In this paper we propose a Platform for Context-Aware Application Development -PCAD based on operating system principles for rapid development and deployment of context aware applications. The platform has been inspired by Operating System design, and is based on a layered architecture that is modular, extensible, and follows the design pattern guidelines. The major premise of the platform was to provide a robust environment influenced by the theories of the architectural design of an operating system, and enable application developers to create context aware applications with minimum effort. The platform provides several services to both applications and sensor software. The platform has a number of benefits: it supports a diverse range of context sources, allows applications to access context sources synchronously or asynchronously using filters, decouples application code form context acquisition, and provides a simple rule engine for context processing. The platform supports soft-real time requirements by opening a direct channel between the context source and applications.

A prototype of the platform is currently under implementation along with two very distinct applications: livestock monitoring and smart city parking. These context-aware applications will verify that applications from a wide range of disparate domains can be built rapidly using the platform.

We purposefully omitted the treatment of context modeling, as our focus was to lay out the critical components of the infrastructure. Context modeling work is deferred to our future iterations of the platform. For this reason, our rule service currently provides very rudimentary services.

Acknowledgment

The authors thank to the Scientific and Technological Research Council of Turkey - TUBITAK for their support on the 3001-Starting R&D Projects Funding Program for the project of “Architecture, Design and Implementation of Context Aware Service Platform”, Grant No: 114E938.

References

[1] A. K. Dey and G. D. Abowd, “Towards a Better Understanding of Context and Context-Awareness,” in *Proc. of the Workshop on the What, Who, Where, When and How of Context-Awareness,* ACM Press, New York, 2000.

[2] A. K. Dey, “Understanding and using context,” *Personal Ubiquitous Computing,* 2001, 5(1), pp. 4-7.

[3] J. I. Hong and J. A. Landay, “An Infrastructure Approach to Context-Aware Computing,” *Human-Computer Interaction,* 2001, 16, pp.287–303.

[4] T. Winograd, “Architectures for Context,” *Human-Computer Interaction,* 2001, 16, pp. 401–419.

[5] M. Perttunen, J. Riekki and O. Lassila, “Context representation and reasoning in pervasive computing: a review,” *International Journal of Multimedia and Ubiquitous Engineering,* 2009, 4(4), pp. 1–28.

[6] T. Strang and C. Linnhoff-Popien, “A Context Modeling Survey,” *First Int. Workshop on Advanced Context Modelling, Reasoning and Management at UbiComp,* 2004.

[7] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, and et al., “A middleware infrastructure for active spaces,” *IEEE Pervasive Computing,* 2002, 1(4), pp. 74–83.

[8] H. Chen, “An Intelligent Broker Architecture for Pervasive Context-Aware Systems,” PhD Thesis, University of Maryland, Baltimore County, 2004.

[9] T. Gu, H. K. Pung, D. Q. Zhang, “A service-oriented middleware for building context-aware services,” *J. of Network and Computer Applications,* 2004, 28 (1), pp. 1-18.

[10] P. Fahy and S. Clarke, “CASS - a middleware for mobile context-aware applications,” *Workshop on Context Awareness, MobiSys, 2004,* 2004.

[11] T. Hofer, W. Schwinger, M. Pichler, and et al., “Context-awareness on mobile devices – the hydrogen approach,” in *Proc. of the 36th Annual Hawaii Int. Conf. on System Sciences,* 2002, pp.292–302.

[12] M. Miraoui, C. Tadj and B. C. Amar, “Architectural Survey of Context-Aware Systems in Pervasive Computing Environment,” *Ubiquitous Computing and Communication Journal,* 2008, 3 (3), pp. 68-76.

[13] H. L. Truong and S. Dustdar, “A survey on context-aware web service systems,” *Int. J. of Web Information Systems,* 2009, 5(1), pp 5–31.

[14] J. Y. Hong, E. H. Suh and S. J. Kim, “Context-aware systems: A literature review and classification,” *Expert System with Applications,* 2009, 36(4), pp. 8509-8522.

[15] S. Stallings, *Operating Systems: Internals and Design Principles* (7. Ed.) Prentice Hall, 2012.

[16] E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Elements of Reusable Object-Oriented Software.* Prentice Hall, 1994.

[17] BIRT. http://www.eclipse.org/birt/ [Accessed April 9, 2015]

[18] D. Ferraiolo, J. Cugini, R. Kuhn, “Role-based access control (RBAC): Features and motivations,” *in Proc. of the 11th Annual Conf. On Computer Security Applications* (New Orleans, LA, Dec. 11-15), 1995, pp. 241–248.

[19] G. Kakarontzas, L. Anthopoulos, D. Chatzakou, D. and A. Vakali,“ A Conceptual Enterprise Architecture Framework for Smart Cities A Survey Based Approach,” Int. Conf. on e-Busines, 2014.

[20] M. Baldauf, S. Dustdar and F. Rosenberg, “A survey on context-aware system,” Int. J. of Ad Hoc and Ubiquitous Computing, 2007, 2(4), pp. 263-277.

[21] N. Masuch, M. Lutzenberger, S. Ahrndt, A. Hessler and S. Albayrak, “A context-aware mobile accessible electric vehicle management system,” in *Proc. of the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS),* 18-21 Sept. 2011, pp. 305-312.

[22] S. Gökay, C. Terwelp, C. Samsel, K-H Krempels K-H, S. Rabenhorst and B. Greber, “Parking Lot Management for Charging Stations,” in *Proc. of the 3rd Int. Conf. on Smart Grids and Green IT Systems-SMARTGREENS 2014,* Barcelona, Spain, 2014.

[23] P. Trusiewicz and J. Legierski, “Parking Reservation – application dedicated for car users based on telecommunications APIs,” in *Proc. of the 2013 Federated Conference on Computer Science and Information Systems (FedCSIS),* 8-11 Sept. 2013, pp. 865-869.