# Design and Implementation of a Software Infrastructure for Integrating Sentient Artefact

Fahim Kawsar, Kaori Fujinami, Tatsuo Nakajima
*Department of Information and Computer Science, Waseda University, Japan*
*{fahim,fujinami,tatsuo}@dcl.info.waseda.ac.jp*

## Abstract

*This paper presents a framework prototype for sentient environments. The framework provides a generic interface to the applications for interacting with sentient artefacts in a unified way regardless of their type and properties. As a result, application development is fairly simple, rapid and independent from the context-aware environments.*

## 1. Introduction

Ubiquitous computing envisioned a future environment that will be aware of its operating context and will be adaptive to ease our interaction. Our approach towards such environment is the environment itself. That means taking the building blocks of the environment and making them smart and context-aware by capturing people's implicit interaction. We have been developing such building blocks, namely everyday life objects by augmenting various kinds of sensors. We call them sentient artefacts. Our vision is to utilize these objects for value added services in addition to their primary services.

Based on our experiences of developing applications that integrate these artefacts for contextual behavior, we have figured out the necessity of a software abstraction that hides the low level details. At the same time such applications have several others requirements like preference management, reliability etc. To satisfy these requirements we are working on a software infrastructure "Prottoy" that attempts to provide a unified view of the underlying physical spaces to the applications. This paper discusses about the design and implementation of the initial version of "Prottoy".

The rest of the paper is organized as follows: Section 2 and 3 point out our design issues and implementation of Prottoy. In section 4 we have presented two sample applications. In Section 5 we have discussed on several issues of Prottoy. Finally section 6 concludes the paper.

## 2. Design Issues

From our experiences of application development with sentient artefacts we have identified the following requirements that must be satisfied for context-aware applications:

1. Due to the ultra heterogonous nature of such artefacts, the application developers need a generic interface that unifies all access issues.
2. End user preference should be reflected in the applications.
3. A security policy in the physical spaces is necessary to identify malicious applications.
4. Applications need to be robust and reliable.
5. The development cost, time and complexity should be minimal.

Considering these issues we have spawn Prottoy with the following design goals:

1. Providing a generalized interface for the developers to interact with the artefacts removing all access issues.
2. Providing storage and proxy service support with in the architecture. Such proxy service can be utilized when the artefacts are not available for reliability and robustness.
3. Providing an authentication policy to access the physical space.
4. Making context-aware application development fairly simple, rapid and easy.
5. Finally providing a personalization/preference reflection feature.

With these views and design considerations we have deployed the initial version of "Prottoy". In the next section the implementation of Prottoy is discussed.

## 3. Implementation

"Prottoy" is composed of few core components and few pluggable components as shown in the figure 1.

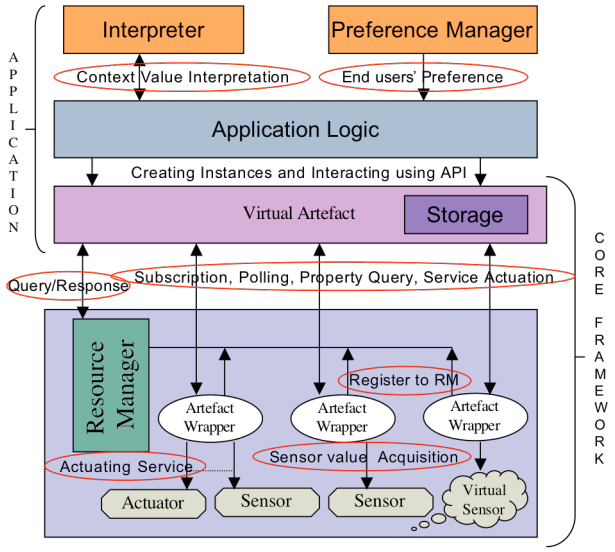


**Figure 1: Framework Architecture**

### 3.1. Core Framework Components

1. **Artefact Wrapper (AW):** It encapsulates the sentient artefacts, sensors, actuators or virtual sensors like weather services, scheduler etc. We have provided a template for the developers to wrap their device drivers or software into this component. AW has its own resource manager that can advertise its service when the global resource manager is absent. In addition it has a simple security measure using IP filtering, that allows an artefact to control access to its service and information from the malicious applications, which approaches to meet our third design goal.
2. **Resource Manager (RM):** As the name implies, it simply registers the properties, services and context information of the artefacts. When application query comes via virtual artefacts it responses accordingly
3. **Virtual Artefact (VA):** It abstracts the smart environments and provides a unified view. Application constructs virtual artefact instances. VA communicates with the resource manager and if an artefact is found VA communicates with that artefact. If everything goes fine VA represents the artefact in the application. From then on, to application this VA instance is the actual artefact. Application can subscribe to this artefact or can poll. Application can also execute services of the physical artefact. Thus this virtual artefact conforms to our first design goal of a generic interface. If storage is enabled, VA creates storage in the application layer. If proxy is enabled then the proxy service of VA activates when the physical artefact is absent. The proxy provides the application a calculated context value with a low accuracy using the storage. These storage and proxy functionalities approach to meet our second design goal.

### 3.2. Components Pluggable to Application

1. **Interpreter:** It maps the context value to the interpreted value. We argue that context interpretation is highly application dependent as the same context can be interpreted in different ways based on the application requirements. So we put this component in the application layer.
2. **Preference Manager:** This component is designed for the end users of the applications developed using Prottoy. It provides the facility to enable or disable the participation of any artefact of the environment on the application based on their preference. We argue that this component meets our final design goal to some extent.

### 3.3. Application Development using Prottoy

The application development using "Prottoy" is fairly simple. In fact developers only need to generate the virtual artefact instance for using the actual artefact. Then developers provide the context to action mapping. A very simple application code snippet with two virtual artefacts looks as follows:

```
/*Specify the artefact properties */
PropertyList props = new PropertyList();
props.add("location","lambdax");
/*Create VA instance, with context,service requirements,properties
and storage and proxy flag*/
VirtualArtefact thermometer = new
        VirtualArtefact("temperature",null,props,false,false);
VirtualArtefact cooler = new
        VirtualArtefact(null,"cooler service",prop,false,false);
/* Poll for value and subscribe */
If(thermometer.status){
        System.out.println(thermometer.poll());
        thermometer.subscribe(this," thermometerListener");
}
If(cooler.status){
Hashtable property=cooler.getProperty() /*Query property*/;
cooler.execute("turn_on"); /*Execute service*/
}
public void thermometerListener(Hashtable data) /*call back */
{
 /*Hash table contains context information */
 }
```

As we see, applications do not need to deal with any network or message management of the architecture; even applications do not have to look for the resource manager

## 4. Sample Applications

We have developed several applications on top of Prottoy; here we are presenting two of them that capture

two scenarios at two distinct places namely dining space and washroom.

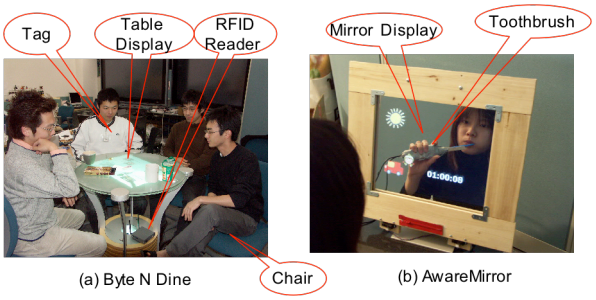


**Figure 2: Sample Applications**

### 4.1. Byte N Dine:

This application shown in figure 2(a), runs in a public/private dining space where the dining table acts as an ambient display. The table displays information/news about topics based on user's preference. We have assumed that the user will carry a RFID tag that reflects his/her preferred topic. This application uses chairs to identify users' presence by chairs' state of use, RFID Tag reader, proximity sensors and the table, which is embedded with a touch screen display. All of these are wrapped in AW.

### 4.2. AwareMirror:

AwareMirror shown in figure 2(b), is a smart mirror installed in the washroom. In addition to its primary task of reflecting some ones image it can also show some information related to the user like schedule, weather forecasting, transportation information etc. The mirror is constructed using acrylic magic mirror board through which only bright color can penetrate. A toothbrush, which is rarely shared with others, is used as an authenticator and also as an indicator of the users' presence. Also proximity sensors embedded in the mirror is used to measure the users' proximity from the mirror. All of these are wrapped in AW.

## 5. Discussion

Prottoy's contributions and distinct features from other works [1,2,3,4] can be summarized as:

1. Generic Interface for all sorts of sensor units and actuators.
2. Complete independence of the application from the underlying architecture.
3. Transparent storage at the application layer and introduction of the proxy service
4. Introduction of the security measure and end user preference management

From our experiences, we have found that application development on top of Prottoy is fairly simple. To be specific, developers only provide the context to action mapping rules. None of the applications that we have developed exceed more than a couple of hundred lines of code

The Virtual Artefact and Artefact Wrapper in conjunction provide the generic interface for everything from a sentient artefact to a single sensor to a web service to an actuator. The artefact wrapper provides the generalization that allows the actual artefact to be replaced anytime with another one. The proxy service is a unique feature of Prottoy. Some of the existing systems provide storage functionality at the artefact layer, our argument is that if the artefact itself is absent in that case the storage is also absent. We think the best use of the context storage or history is the prediction of the context, so it should be somewhere that can be accessible when the artefact is absent. Virtual artefact perfectly solves the problem by hosting the storage and providing proxy service. There is no context interpreter in Prottoy core, as we think context interpretation is completely application dependent. For example consider a chair that provides it's state of use. We can use this information to infer its user is sitting/not sitting (activity) or it's users location (at chair's location) based on the applications requirement. Our argument is we cannot broadly confine the interpretation of context information. So we have separated it form the core and provide it as a pluggable component at the application layer. However there are few issues that we are further investigating like security measure, preference component, proxy service etc. We are working on these issues with great interest and hope to come up with some interesting results soon.

## 6. Conclusion

In this paper we have tried to provide the ins and outs of Prottoy and it's approach in a summarized way. We believe our proposition and ongoing work will be able to resolve all the issues to the utmost level and will provide a seamless development platform for context-aware application developer.

## 7. References

[1] A. K. Dey. et al. *"A Conceptual Framework and a toolkit for supporting the rapid prototyping of context-aware applications"*. Human-Computer Interaction, Vol-16 2001

[2] B. L. Brumittet et al. *"Easy Living: technologies for Intelligent Environments"* In the proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing '2000

[3] Caswell at el. *Creating Web representations for Places* Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing

[4] C. Philip R. et al, *"An Open Agent Architecture"*. In the proceedings of the AAAI Spring Symposium Series on Software Agents,'94