

# Software Architecture for Self-organizable Universal Boards

Ryo Ohsawa<sup>1</sup>, Naohiko Kohtake<sup>1</sup>, Kazunori Takashio<sup>1</sup>, and Hideyuki Tokuda<sup>1</sup>

Graduate School of Media and Governance, Keio University  
5322 Endo, Fujisawa, Kanagawa, 252 8520, Japan  
{ryo, nao, kaz, hxt}@ht.sfc.keio.ac.jp  
<http://www.ht.sfc.keio.ac.jp/>

**Abstract.** Recently, the technology of the pervasive computing environment is developing rapidly because of the advancement in computer science. In order to realize such environment, the user must implant multiple computer devices and sensors. However, this is not easy because it is difficult for users who are unfamiliar with computing technology, to use them individually. We have developed a board type smart material called "u-Texture", which has a built-in computer and sensors. u-Texture can be connected with other u-Textures to form various shapes, and can recognize its entire structure. At the same time, the u-Texture can distinguish the possible services it can execute and display to the user. This paper describes the software architecture to achieve the self-organizable universal board.

## 1 Introduction

In pervasive computing environment, computers, sensors, devices, and networks are embedded in or attached to non-smart objects, and connected to create context-aware pervasive services. Recently, many non-smart objects are converted into smart objects by various approaches, and a large number of smart home, smart office, smart furniture, etc., are realized and to support human activities. However, it is difficult for users who are unfamiliar with computing technology, especially pervasive computing technology, to create and maintain pervasive computing environments without experts. Moreover, most of these environments are handmade by users and the cost and time for building such an environment is a barrier to the development of appropriate pervasive services. To solve this problem, previous research for realizing pervasive computing environments, with objects such as tables, shelves, and drawers in homes and offices are not originally smart, but namely most of them are enhanced by attaching computers, sensors, and devices [2] [4].

Our approach aims to enlarge functions by making components, such as boards, legs, and props of tables and shelves, smart in advance. As users assemble objects with these smart components, which are made beforehand to have functions to recognize the surroundings and to realize the pervasive computing environments, these components alter their functions autonomously according to the shapes of how they are assembled, and then, objects work as smart objects. In the process of developing in the prototype of the first smart components, we focused on the "board shape", which is the basic to form most furniture such as tables and shelves. Then, we have developed a board type smart material called "u-Texture" to realize the pervasive computing environment easily by assembling them physically. [1]

## 2 u-Texture : Self-Organizable Universal Boards

This section describes the overview of u-Texture.

### 2.1 Basic Functions

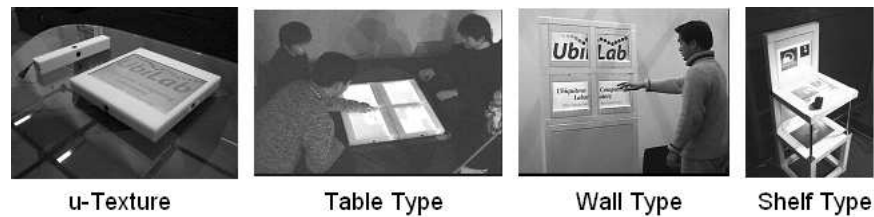
The prototype of u-Texture is 320 mm square, 48 mm thick, and 4300 g, approximately the size and shape of a pizza box. u-Texture consists of the following devices with basic computer devices such as processor and hard disk drive. Additionally, u-Texture has serial port interfaces on its 4 sides to confirm if it is connected or not, and also accelerometers to find its inclination. There are wired LAN interfaces on each four sides to exchange data between adjacent u-Textures. the classification table of a built-in device according to the usage is shown in Table 1.

**Table 1.** Built-in devices

Usage	Device
Adjacent Detection	Infra-red Communication Device x 4
Connected Detection	RS-232C Serial Interface x 4
Data Communication	100BASE-TX/10BASE-T Ethernet LAN x 4 , IEEE 802.11b Wireless LAN
inclination Detection	Acceleration sensor
Input	Touch display , RF Reader for ISO15693
Output	SXGA+14.1 type thin film transistor color LCD , Speaker , RF Writer for ISO15693

## 2.2 Assembly Example

Fig.1 shows the appearance of u-Texture and examples of several smart objects assembled with u-Textures.



**Fig. 1.** u-Texture connects with other ones horizontally and vertically, and collaborates with each other by exchanging location and inclination information, commands, and data.

**Table Type** can exchange data of each u-Texture onto its screen. It can be created by connecting u-Textures and setting it horizontally. When several u-Textures are connected, an arrow will indicate the direction of the other connected u-Textureon screens of each u-Texture. The data will be copied to the u-Texture connected to the direction of the arrow by dragging it to the arrow.

**Wall Type** can display data from a u-Texture widely in cooperation with other connected u-Textures. It can be created by connecting u-Textures and setting it vertically. In the case of connecting u-Textures equally in length and breadth of 2 x 2 or 3 x 3, a magnified picture of one designated u-Texture will be displayed.

**Shelf Type** is a shelf that recognizes what is put on top of itself. It can be created by assembling u-Textures vertically and horizontally. Putting objects with RF-tag on the shelf, it can recognize the object and records it as data. With the action, a user can confirm its detailed information by output from the display of the shelf and also search via network where the object has been put on.

## 2.3 Activate Sequence

The basic three actions of u-Texture as a smart object are given as follows.

**Recognition** . As a user assembles u-Textures, the assembled u-Textures share information such as connectivity with other u-Texture, directions of connection, IDs of adjoining u-Textures, and inclinations of themselves. With this information, each u-Texture recognizes its assembled shape and location on the assembled shape.

**Adaptation** . Available applications corresponding to the recognized smart object will be selected automatically among different applications preinstalled in each u-Texture. A user 's input determines one application when there are several choices.

**Cooperation** . Once which application is to be used is determined, each u-Texture behaves autonomously and works together according to the smart object shape, and each location and inclination.

## 3 Software Architecture for u-Texture

This section describes the software architecture for u-Texture to work as smart objects.

### 3.1 Overview

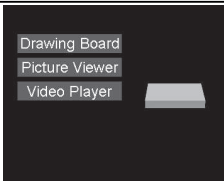
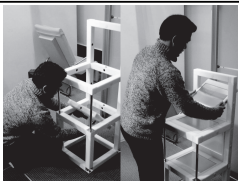
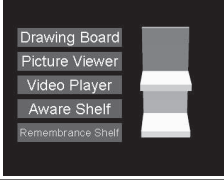

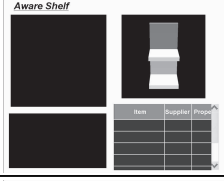
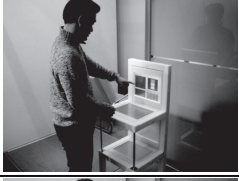
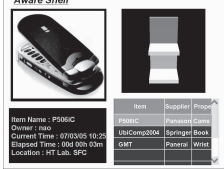

First of all, the overview of software architecture is described. Fig.2 shows the appearance when the shelf type smart object is actually assembled.

**Function Requirement** The function requirement for reassembled u-Textures to offer the service to the user is described as follows.

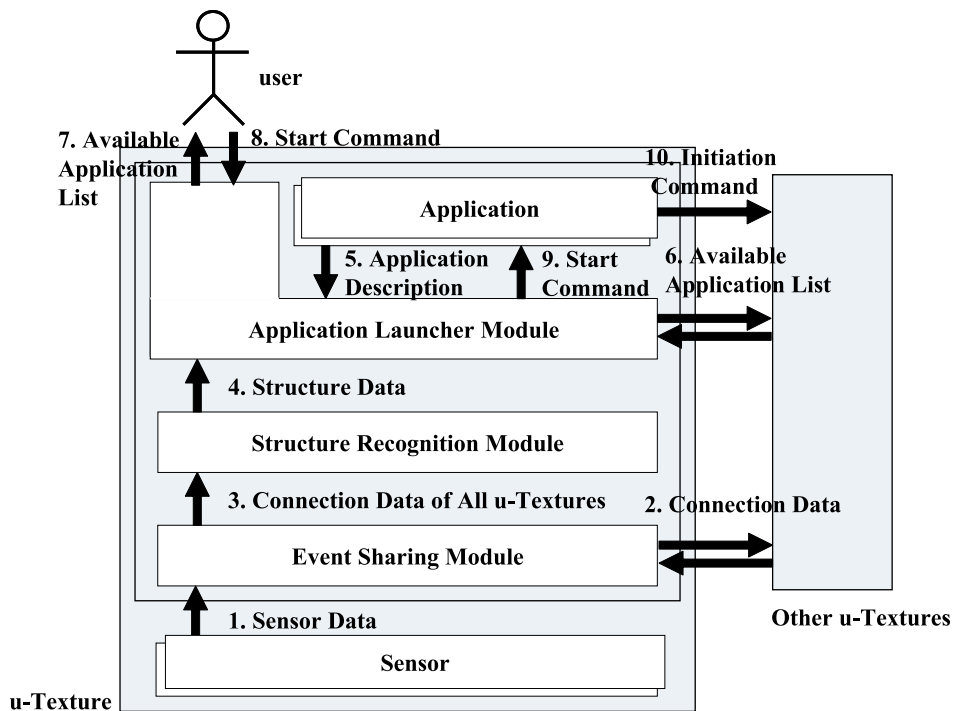
**Recognition** . The algorithm which decides the structure based on shared information is necessary so that individual u-Texture may process it according to the self-position of the entire structure.

**Adaptation** . The executable application is different according to the entire structure. For example, In the case of the shelf type, the multi display application is unusable. Therefore, a u-Texture should recognize whether a applications can be used according to the entire structure.

**Cooperation** . It is necessary to change the model of cooperation according to the kind of the application.

	GUI displayed in the u-Texture that user build in at the end.	Assembly of u-Texture by a user
The user builds in u-Textures. GUI on u-Texture displays the menu list of applications that can be start and entire structure image of u-Textures.		
u-Textures are self-recognized, and change the menu list and the structure image. <b>(Recognition)</b>		
If the user clicks an application menu, the corresponding application starts. In this case, Aware Shelf was started. <b>(Adaptation)</b>		
If the user places objects, the information of the objects can be displayed by sharing information between u-Textures <b>(Cooperation)</b>		

**Fig. 2.** Flow in realization of shelf type smart object by assembling u-Textures from top; A user assembles u-Textures, The u-Textures indicate application candidates by self-organization, A user selects desired application, The u-Textures run the selected application cooperatively



**Fig. 3.** Activation Data Flow

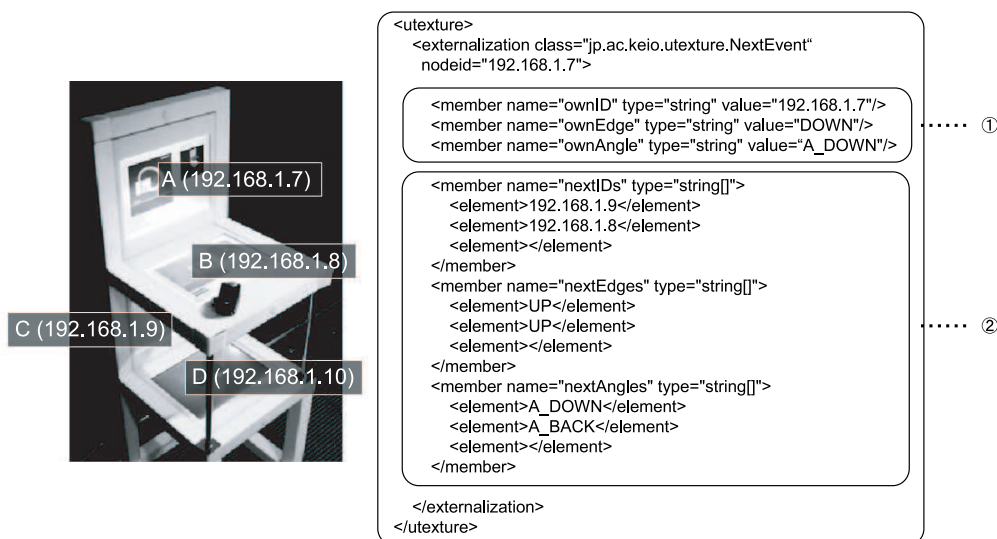
**Activation Data Flow** The activation data flow is showed in Fig.3. Details are described as follows.

1. Sensors (ex. an acceleration sensor, serial interfaces, etc) send sensing data regularly to the event sharing module.
2. u-Textures share the connection data that corresponds data of the acceleration sensor and serial interface.
3. The connection data of all connected u-Textures is sent to the structure recognition module. Then structure recognition module calculate the entire structure.
4. Structure data is sent to the application launcher module.
5. The application launcher module receives the application descriptions from all applications. An application descriptions describes the basic information of the application and the condition by which the application can start. The application launcher distinguish whether the application can start or not in present entire structure.
6. The application launcher module share the list of applications that can start among combined u-Textures. When sets of the application list A, B and C, u-Texture consider the AUBUC as the application that can be started.
7. The application launcher module presents the user with the available application list.
8. The user send the application start command.
9. When the application launcher module receive the start command from the user, it sends the command to the corresponding application.
10. The application sends the initiation command to other connected u-Textures. When a u-Texture has not been installed the application, the u-Texture copy the application from another connected u-Texture and install it.

### 3.2 Recognition

The recognition sequence is divided into the event sharing and the structure calculation.

**Event Sharing** To understand the entire structure, assembled u-Texture should share the change event of the connection and the inclination with all connected u-Textures. First of all, a u-Texture acquires its inclination with an acceleration sensor and information on the u-Texture directly connected by using the serial interface. However, information on the u-Texture connected through other u-Texture is also necessary so that individual u-Texture may understand the entire structure. To solve this problem, all u-Textures broadcast to own information by using Ethernet LAN. Fig.4 shows the example of the shelf type smart object assembled by u-Textures and the data example that *u-Texture A* broadcasts.



**Fig. 4.** Example of the data that broadcasted from *u-Texture A*

The explanation of data example shown in Fig.4 is described as follows.

1. Information on *u-Texture A*.
  - ownID** is uniquely identified by structure of the entire u-Texture and is allocated IP when cable LAN is connected by using Auto-IP function.
  - ownEdge** is the edge connected with another u-Texture. (RIGHT, LEFT, DOWN, UP)
  - ownAngle** is shows the inclination to surface of the earth. This value shows respect under surface of the earth and horizontal u-Texture. For instance, in the case of A.FRONT The display of u-Texture is the horizontal in the lower side with surface of the earth. (A\_ FRONT, A\_ BACK, A\_ LEFT, A\_ RIGHT, A\_ UP, A\_ DOWN)

2. Information on the u-Texture connected directly with *u-Texture A*. Three element exists because three u-Textures can be connected in the maximum compared with one edge.

**nextIDs** are the IDs of the u-Texture connected directly with *u-Texture A*.

**nextEdges** are the edges connected with *u-Texture A*.

**nextAngles** show the inclinations to surface of the earth.

**Structure Calculation** The algorithm that can uniquely decide the structure based on shared information is necessary so that individual u-Texture may process it according to the self-position of the entire structure. u-Texture shares ID, the connection information, and the inclination among combined u-Textures, and respectively calculates the entire structure. An entire of u-Texture structure is expressed by coordinates and the inclinations based on the lattice point shown in fig.5. The inclination has *front*, *side*, and *bottom* of three per one lattice point coordinates. Fig.6 shows example of structure information in the case of the shelf type smart object.

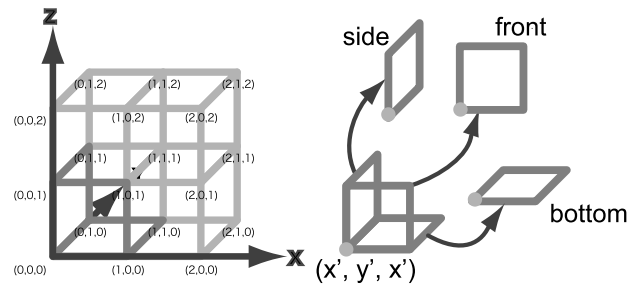


Fig. 5. u-Texture's Coordinate and inclination

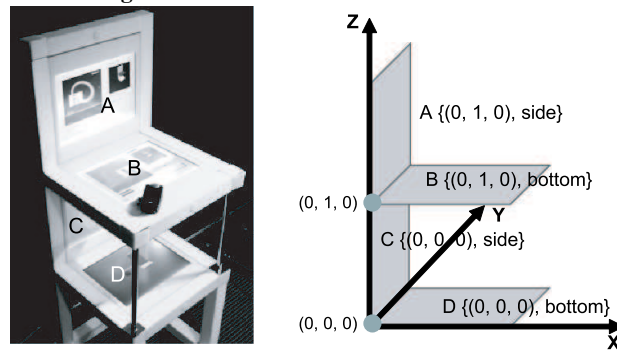


Fig. 6. Example of Structure Information in the case of the shelf type smart object

### 3.3 Adaptation

The executable application is different according to the entire structure. For example, In the case of shelf type, the multi display application is unusable. Therefore, a u-Texture should recognize whether a applications can be used according to the entire structure. Moreover, the application installation function is also necessary as software can be easily built into a u-Texture.

**Start Condition** An individual application has the attribute, and the application developer describe in what structure the application can be started in the attribute. Fig.7 shows the example of describing the application attribute.

**application** are the IDs of the u-Texture connected directly with *u-Texture A*.

**name** is the application name displayed in user interface.

**id** is the ID uniquely identified in the world.

**component-set** describes basic information on the application.

**component** element describes information on a module of a application. *class* attribute describes the path name of the class that initializes the application.

**available-condition** element describes the condition that application can be used. application attribute can set *available-condition* and *uavailable-condition*. Only when both conditions are filled, the application becomes possible the start. Details of the attribute are described in the table2.

**uavailable-condition** element describes the condition that application *cannot* be used.

```

<application name="AwareShelf" id="4631205780235174">
  <component-set>
    <component class="jp.ac.keio.utexture.AwareShelf" />
  </component-set>
  <available-condition>
    <condition object=">1" inclination="bottom" />
  </available-condition>
</application>

```

Fig. 7. Application Attribute

Table 2. Attribute of Available-condition and Uavailable-condition Element

Attribute	Value	Explanation of available-condition (disable-condition)
object	all	All u-Textures exists or is the following state. (It is not.)
	$x >$	U-Textures that are more than $x$ exist or are the following state. (It is not.)
	$x =$	$x$ u-Textures exist or are the following state. (It is not.)
	$> x$	Less than $x$ u-Textures exists or is the following state. (It is not.)
inclination	side	The u-Textures of the number specified by value of <i>object</i> are <i>side</i> (defined in section 3.2).
	front	The u-Textures of the number specified by value of <i>object</i> are <i>front</i> (defined in section 3.2).
	bottom	The u-Textures of the number specified by value of <i>object</i> are <i>bottom</i> (defined in section 3.2).

**Application Installation** The application installation function is also necessary so software can be easily built into a u-Texture. There are roughly separately two techniques for installing the application in a u-Texture.

The First is a technique for download and installing the application from the application offer server by way of the network. (ex. Java Web Start Technology). u-Texture provides UI for the installation of a new application and installs the application according to the user's input. When the user download the application a well-known server to tell the location of the server with the application is necessary. It is like the search engine or the portal site when the user searches the website. After connecting the well-known server, u-Texture sends the request of the application download to the server that offers the application.

The second is a technique for copying the application from another u-Texture connected directly. First of all, Assembled uTextures exchanges application information that can be started in the entire structure mutually each other. When sets of the applications of assembled u-Texture A, B and C are  $A, B, C$ , U-Texture offers the user  $A \cup B \cup C$  as an application that can be started. When the user starts the application, u-Texture in which the application is not installed copies the application from either of connected u-Texture.

### 3.4 Cooperation

The model of cooperation must change according to the kind of the application. There are two models which operate cooperatively among u-Textures. The master-slave model is the model that all u-Textures refer to the data that exists in a u-Texture. The distributed autonomous model is the model that doesn't show the data source in a u-Texture and autonomous operates each u-Texture. In the case of Aware Shelf application that is showed Fig2, The distributed autonomous model is used to share the information of objects.

**Master-Slave Model** is the model that all u-Textures refer the data that a u-Texture has. The u-Texture that has data is assumed to be the master, and u-Texture that refers to data is assumed to be the slave. The application of the slaves ends when u-Texture of the master is removed or ended. Process of slaves are later than their master, because slaves does the processing after receiving data from their master. As the example of the application that uses this model, the application to which one u-Texture applies the Web camera, and the image is displayed by the expansion with two or more u-Texture exists.

**Distributed Autonomous Model** is the model that does not show the data source in a u-Texture and autonomous operates each u-Texture. The application on all u-Texture does not an end event if a u-Texture is removed from the group. It is not like the master-slave model. Moreover, the delay problem of the master-slave model does not occur. However, synchronous processing of the application becomes complex, the possibility that bug is generated is high in distributed autonomous model. As the example of the application that uses this model, the multi display application that connects with video streaming server displays image corresponding to the position of u-Texture exists.

## 4 Discussions

In this section, the consideration concerning the software architecture implemented this time and the view in the future are described.

## 4.1 Recognition

**Event Sharing** . The broadcast that each u-Texture uses to transmit all events to every u-Texture is used to recognize the entire structure. This event sharing mechanism can be used for the sensor event. However, when the amount of the event increase, the overhead grows. It is necessary to implement the look-up mechanism that searches for which offers the event by the broadcast and to transmitting information only to the registered u-Texture.

**Structure Calculation** . Because the structure can be recognized with an acceleration sensor that calculated the inclination from the ground level. We cannot express whether the u-Texture is facing forward or backward with our current implementation. To solve this problem, it is necessary to build the azimuth sensor into the u-Texture or build the computer into the joint that combined between u-Textures.

## 4.2 Adaptation

Available applications corresponding to the recognized smart object will be selected automatically among different applications pre-installed in each u-Texture. A user's input determines which application is to be used when there are several choices. However, it is a hassle for the user to input their choice every time. If a u-Texture can recognize the user, the u-Texture can start the most appropriate application in a present structure based on the history of the user.

## 4.3 Cooperation

In our current implementation of u-Texture cooperation heavily depends on each application. We therefore need a more generic cooperation model. This would be useful when a heterogeneous device were to cooperate with another u-Texture. For example, transmitting image from a digital camera to wall type smart object requires a large amount of computational power and network bandwidth. We could have the camera send its image to a u-Texture and have it duplicated to other u-Textures, instead of sending its image to every u-Texture.

## 5 Related Works

As a relative work of board type smart object, ConnecTables [3], can be used as a table for the collaborative activity like u-Texture. A ConnecTable can recognize other ConnecTables and can cooperatively work. ConnecTable consists of BEACH, which is the software architecture of ConnecTables. It is based on the server-client model, and requires access to a server in order to operate ConnecTables cooperatively. u-Texture have Ethernet LAN Interface in four directions and the framework that supports the serverless cooperation.

## 6 Conclusion

We have developed a board type smart material called "u-Texture". This paper described the software architecture to offer users the pervasive service.

## 7 Acknowledgement

This research has been conducted as part of Ubila Project supported by Ministry of Internal Affairs and Communications, Japan.

## References

1. N. Kohtake, T. Yonezawa, R. Ohsawa, Y. Matsukura, M. Iwai, K. Takashio, and H. Tokuda. Creating pervasive services with self-organizable universal boards. In *Third International Conference on Pervasive Computing (PERVASIVE2005)*, Video To be appeared, May 2005.
2. N. A. Streitz, J. Geibler, T. Holmer, S. Konomi, C. Muller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-LAND: an interactive landscape for creativity and innovation. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 120–127. ACM Press, 1999.
3. P. Tandler, T. Prante, C. Muller-Tomfelde, N. Streitz, and R. Steinmetz. ConnecTables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. UIST'01.
4. H. Tokuda, K. Takashio, J. Nakazawa, K. Matsumiya, M. Ito, and M. Saito. Sf2: Smart furniture for creating ubiquitous applications. *IEEE Proceedings of International Workshop on Cyberspace Technologies and Societies (IWCTS2004)*, pages 423–429, 1 2004.