RC22480 (W0206-029) June 10, 2002
Computer Science

IBM Research Report

Personalizing Behavior in Context-Aware Workspaces

Sachiko Yoshihama, Paul B. Chou, Danny Wong
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Personalizing Behavior in Context-Aware Workspaces

Sachiko Yoshihama, Paul Chou, Danny Wong

IBM T. J. Watson Research Center, Yorktown Heights, NY 10598
{sachiko, pchou, dcwong}@us.ibm.com

Abstract. It has become increasingly important to support agile organizations with easily re-configurable work environments. This paper describes the on-going work on a framework for personalizing the dynamic behavior of context-aware workspaces. The framework promotes the development of futuristic workspaces that support people's work practices in an unobtrusive, context-aware, and personalized manner. The framework accommodates individual preferences and organizational policies in a way that allows rapid and impromptu customizations. There are several advantages to this, in particular, the ability to represent users' preferences about their work environments separately from the actual configurations of the physical spaces they occupy at a given time, thus supporting emerging workplace needs such as hoteling and impromptu group settings.

1 Introduction

The advent of pervasive computing technologies that allows employees to work from anywhere at anytime and the continuous pressure for businesses to reduce cost have prompted corporations to re-evaluate their workplace objectives and strategies [1]. In addition to providing a place for individuals and teams to function, the ability to dynamically tailor workspace's functions based on its usage has become more desirable for the support of better space utilization and enhanced individual satisfaction with the work.

Already digital technologies are integrated with the furniture and architectural elements of the physical space, particularly in the areas of communication and audio/video support [2][3]. We envision that workspaces will become increasingly intelligent with the integration of additional technology elements such as smart sensors, actuators, and displays. From a user's perspective, not only the workspace supports her work in an unobtrusive [4][5], context-aware manner [6][7][8], but also supports it in the way consistent to her preferences. No longer must the user be bounded to a particular office in order to enjoy the familiar workspace experience, she can be working in any available space yet still feels like at home.

This paper describes the work aimed at supporting dynamic behaviors of context-aware workspaces as a part of an on-going office-of-the-future research project. Our goal is to provide a framework that makes such workspaces easily programmable and customizable. Towards this goal, the framework makes extensive use of common programming metaphors that have made other programming environments, such as

graphical user interfaces, simple to program and to customize. The overall framework is composed of two parts. The core is a *context aggregation framework* for expressing the structure and condition of the workspace – essentially providing a model of the context. The second is a *behavior management framework*, which facilitates the interactive nature of the workspace. The behavior management framework builds upon the data aggregation and dissemination capabilities of the context aggregation framework.

2 Behavior in Context-Aware Workspaces

Behavior is an important characteristic of context-aware environments. A behavior can be viewed as an action, response, or interaction that is carried out within a particular context. Behaviors often involve the participation of human users, but may also occur in a natural, unobtrusive background mode. Behaviors may occur for a variety of reasons, for example, in relation to some physical stimulus, the satisfaction of some condition, the arrival of a user, or for non-physical reasons such as the expiration of a timer. The macroscopic behavior of a workspace is defined through the aggregate effect of a collection of individual behaviors, as well as through the interplay among individual behaviors. For example, a key aspect of the overall configuration of a workspace is deciding which behavior should occur when the user enters the workspace. The framework provides a systematic way of expressing and managing preferences about the overall behavior of the context-aware workspace. The framework also provides a way of associating particular behaviors with elements of the workspace, in configuring which of them should provide a particular response.

There are several key design goals in managing behavior. First, behaviors must be easily configurable and extendable. Secondly, the overall behavior of the space should be treated as an aggregate of many individual behaviors; each of them is adaptable and reusable. Finally, behaviors are not hard-wired to a particular space. The behaviors must be dynamically reconfigured as the situation changes; e.g., as the owner of the space changes.

3 The Context Aggregation Framework

The context aggregation framework provides a mechanism for modeling, describing, and managing the state of the various entities in the physical environment, such as people, spaces, services, and devices. The framework also allows the representation of conceptual entities such as conditions, tasks, and intermediate concepts such as the presence or absence of a user.

The context aggregation framework is an extension of several well-known programming constructs. In particular, it extends a “blackboard” model [9] to support a hierarchical representation of entities. In this model, the context of the workspace is represented, essentially, as a tree. The nodes of the tree represent either physical or conceptual elements of the workspace. Parent and ancestor nodes may represent groupings or higher-levels of abstraction of the physical and conceptual elements. The

blackboard mechanism provides a publish-subscription capability. Each node of the tree may be addressed by a simple path expression. Data values may be published to the nodes of the hierarchy. Interested application components may register themselves as subscribers to a node. Subscribers subsequently received event notifications when the condition of the data associated with the node changes, when the node is created or removed, or when application-specific events are explicitly posted to the node. Additionally, an event propagation mechanism allows an event to "bubble-up" from the node it occurred toward the root of the tree. It allows subscribers to intercept and handle events occurring in subordinate nodes in the hierarchy, thus allowing subscription at a proper level of abstraction of the context.

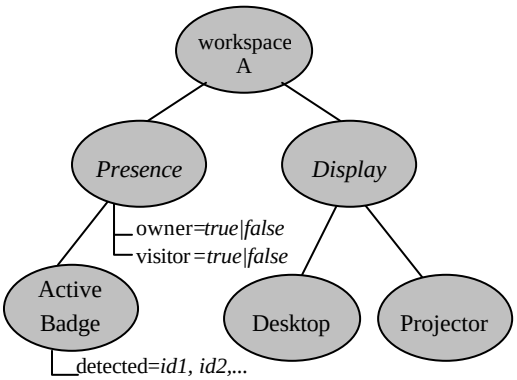

Fig. 1. Example Workspace Hierarchy

Fig. 1 shows an example hierarchy that represents a workspace and contextual information. In this case the workspace is modeled as a collection of two conceptual entities, with the Presence node representing the status of occupancy, and the Display mode representing display capabilities within the workspace. Note that the Presence node has two properties associated with it, indicating whether the presence of the owner and/or a visitor have been detected. Additionally, the hierarchy includes nodes representing three specific devices, ActiveBadge representing a radio frequency badge reader and Desktop and Projector representing a desktop display device and a projection device respectively. Publishing values to these nodes results in the display of the published value. The dynamic behavior management capabilities of the system, described below, provide an automatic mechanism for sequencing responses and interactions among these elements of the workspace in a way that useful end-to-end behaviors are formed.

4 The Behavior Management Framework

Programmability of behavior is achieved through the use of application-defined components known as *agents*. At its simplest level, an agent is an application component that carries out its action by responding to events that occur at one or more context nodes. An agent is activated upon an event being posted to that node, or upon a datum being published to that node. The agent can then analyze information from the posted event or published datum, may collect further information from data associated with other context nodes, and execute commands that cause physical actions within the environment. It may generate and post further events, or publish data to other context nodes. This may, in turn, cause other agents to be activated.

4.1 Managing Behavior by Managing Agents

Proper behaviors are achieved by choosing, assigning, and configuring agents and their relationship to particular nodes within the context aggregation framework. In the example above, a simple behavior would be achieved by associating an HTML renderer agent with the Desktop node in the context hierarchy. Whenever an HTML datum is published to that node, the renderer agent is activated, causing the datum to be interpreted and displayed in an HTML browser. Associating a different renderer agent with the Desktop context element, for example, a voice browser, would result in a different behavior, and a completely different experience, at the desktop. More complex behaviors are achieved by arranging sequences or groupings of agents and associating them with nodes of the workspace model.

Individual behaviors are defined through agent definitions, and are stored in a preference repository. Agent definitions consist of several parts that describe the agent's basic nature, capabilities, and how it should interact with the context hierarchy as well as with other agents. The basic elements of an agent definition include: a unique *agent identifier*, a set of *targets* that contains the context nodes with which the agent should be associated, an optional *filter* that may be specified to discriminate among the posted events and published data arriving at a targeted context node, and most importantly, the *implementation* of the action associated with the agent.

4.2 Managing Aggregate Behaviors

A core component of the behavior management framework is the *agent manager* that manages the aggregate behavior of the workspace. The agent manager provides a clearly defined life cycle for agents as shown in Fig. 2. In its life cycle, an agent can be in one of the 3 states: idle, ready or active. An instance of an agent is created using the information in the agent implementation field of the agent definition. The agent is initially placed in the *idle* state. Then the target field is evaluated to identify the relevant context nodes. If all of the target nodes exist, the agent is attached to them as an event listener and is placed in the *ready* state. An agent that is not attached to its target nodes and awaiting subsequent attachment, remains in the idle state. When an agent in the ready state is notified of the arrival of an event, it is placed into the *active* state, in which it can proceed to take appropriate actions. Upon completion of its action, it then returns to the ready state to wait for the next activation. When the agent

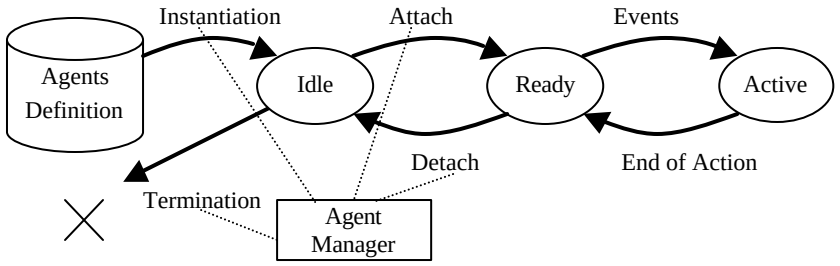

Fig. 2. Agent Life Cycle

is no longer needed, it is detached from what it is listening to, then terminated.

The agent manager observes particular types of events that may trigger life cycle changes in agents. There are several key situations that are monitored. First, the agent manager monitors the inventory of agent definitions. When an agent definition is updated, the agent manager awaits the existing agent to exit from the active state, terminates it and creates a new instance of the agent based on the updated agent definition to adopt the new behavior. Second, the agent manager monitors the structure of the context hierarchy and attaches or detaches the agents as the certain context nodes are created or removed. The subscription mechanism provided by the context aggregation framework makes this job easier, as the agent manager may simply subscribe to the root node. When any change occurs anywhere in the hierarchy, the agent manager is notified as the result of event propagation. Third, the agent manager monitors ownership of the physical space. It instantiates appropriate agents when a user is associated with (or gains the ownership of) the space. The agent manager may terminate certain agents when the user is unassociated from (or releases the ownership of) the space.

4.3 Managing Behavioral Preferences through Agent Selection

Behaviors are grouped according to whether they are general (or *universal*), whether they belong to a particular user or a space. For example, universal behaviors are generally expected to apply to all workspaces (e.g., to turn-on the light when somebody is there), other *user* behaviors may signify individual overriding preferences (e.g., to turn-on only the task light and not the ceiling lights). Similarly, *space* behaviors may be expected to apply to a particular space. For example, pull-down the window shade at 3:00 p.m. only if the space has windows facing the west.

The agent definitions are stored in a preference repository, which is logically divided into 3 categories: universal, user, and space. Each user or space has an entry in the repository, which contains agent definitions. As a situation requiring the change of the behavior arises, such as a new owner of the space has been identified, the agent manager composes the new behavior by selecting and initializing an appropriate set of agents from the repository according to the identities of the user and space.

For example, in a simple approach, the agents defined in the different categories with the same name may be regarded as the competing agents, thus, only one of them will be selected for instantiation. First, the agent manager attempts to initialize each of the universal agents, unless a space agent overrides the universal agent with the same name. Similarly, a user agent may override the space agent with the same name. It is possible to explicitly specify a flag in the universal or space agent definition, so that a space or user agent cannot override it.

4.4 An Example of Programmable Behavior: Personalized Wallpaper

Let's use a simple wallpaper application to demonstrate how the context aggregation and behavior management frameworks work together to support dynamic behaviors. Again, let's assume the workspace is equipped with an active badge reader that

provides the presence information and two display devices: a desktop display and an Everywhere Display projector (ED-projector) [10]. The wallpaper application displays a graphical image on a wall using the projector or on the desktop display. The choice of the image depends on the identity of the current owner of the space and the status of occupancy. The owner may choose different pictures for different occasions: a family portrait when she is alone, the view of Niagara Falls when a visitor is present, and a smiley when she is away. The owner may also specify a preferred location for displaying the image in different circumstances.

Figure 3 illustrates how personalized wallpaper application behavior can be accomplished with the use of three agents. The Presence Aggregator Agent analyzes the badge data reported by the ActiveBadge node and updates the properties of the Presence node. The Wallpaper Agent listens to the events on the Presence node. When the office owner arrives, the Wallpaper Agent sends a command to the display service represented by the Display node to display his family portrait. Similarly when a visitor drops in, the Wallpaper Agent issues another command to replace the picture with the view of Niagara Falls. The Display Selector Agent, upon receiving a command event from the Display node, forwards the incoming request to a suitable display device currently available. Note that the Wallpaper Agent is defined in the "user" category and the Display Selector Agent is in the "space" category.

When the workspace is assigned to a new owner, the agent manager terminates the existing instance of the Wallpaper Agent and creates a new instance of the Wallpaper Agent based on the new owner's preference stored in the preference repository. When the previous owner moves into a different office, the wallpaper application employs a different Display Selection Agent specific to new office, tailored to the display devices that are available in there.

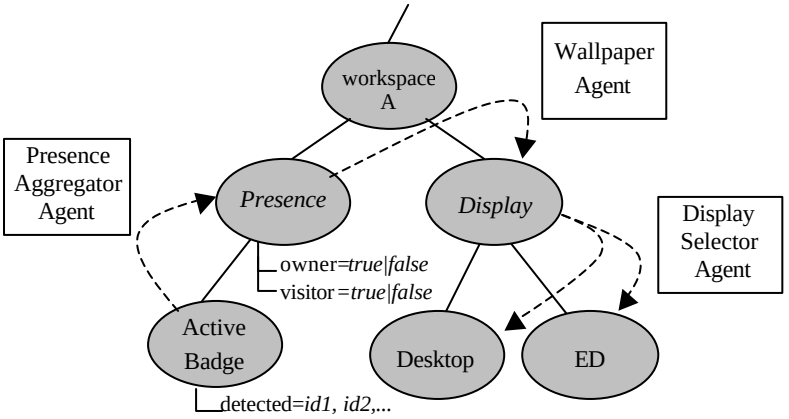

Fig. 3. Wallpaper Application

5 Current Implementation

The context aggregation and the behavior management frameworks are being tested to support the BlueSpace prototype [11] in a hoteling [12] environment. The workspace incorporates a set of sensors for measuring environment conditions such as ambient lighting, temperature, humidity, and noise level. It employs an active badge reader and a seating sensor for detecting occupant presence and position (i.e. sitting vs. standing). Lighting (both ceiling and task lights), local temperature, and airflow of the workspace can be digitally controlled. In addition, the workspace is equipped with an office-front display for showing the office owner's information and status to people walking by. Two flat-panel displays are installed on the desktop with one used as a peripheral display designed to provide easy control of the devices and quick access to frequently accessed information. The space also provides an ED-projector capable of projecting content onto various surfaces in the space. Currently a set of application behaviors can be personalized in BlueSpace, including the situation triggered environment adjustment, alert delivery, wallpaper display, and status sharing. The reader is referred to [11] for a more detailed description of these behaviors.

The overall framework is written in Java and packaged as an OSGi [13] bundle. Device drivers, services, and applications are installed as separate bundles. The OSGi framework allows us to dynamically change the system configuration through dynamic installation and update of bundles.

Remote access to the context aggregation framework is provided by a set of API's exposed through SOAP/HTTP. Remote processes can subscribe, query, and update property values or post data into the context hierarchy. They can also initiate structure changes to the hierarchy such as adding a node that represents a new device service available to the environment.

Presently, three types of agent implementation are supported. The developer may choose to use a simple XML-based scripting language or the familiar JavaScript for implementing simple agents. Or, she may choose to write in Java to implement more complicate actions. The Java-based implementations can be packaged as additional OSGi bundles. The agent instantiation mechanism for each type of implementation is pluggable, allowing easy incorporation of additional types.

6 Conclusions and Future Work

The context aggregation and behavior management framework described in this paper represents an attempt to provide a consistent and scaleable treatment for developing, deploying, and personalizing context-aware workspaces. The framework provides a simply yet powerful mechanism for representing, aggregating and disseminating contextual information with a hierarchical model. It also provides the support of customizable and re-configurable behaviors by employing a programming model that promotes composition and adaptability. The support of the BlueSpace applications suggests that the framework is capable of supporting rich, composite behaviors.

Much work remains, however. The subject of identifying and resolving conflicts among agents competing for the control of the same resources deserves a thorough

treatment. It is also interesting to extend the framework to treat behaviors as first-class entities that can be composed, selected, instantiated, and terminated. Another interesting topic for further exploration is the support of fine-grain access control of the entities in the hierarchical model to accommodate privacy and confidentiality considerations. Lastly, it is important to address the need of supporting mobile devices in light of the emerging web services architecture.

Acknowledgments

We would like to thank other members of the BlueSpace team: Scott McFaddin, Anthony Levas, Marco Gruteser, Jennifer Lai, Mark Podlaseck, Claudio Pinhanez, and Marisa Viveros. Their contributions and creative minds made this work possible. We would also like to acknowledge our partners from Steelcase: Joe Branc, Joel Stanfield, Charlie Forslund, Mark Baloga, and Jason Heredia for sharing their workspace knowledge and contributions to the BlueSpace prototype design.

References

1. Budd, C.. The Office: 1950 to the Present. In *Workspheres: Design and Contemporary Work Styles*, P. Antonelli (ed.). The Museum of Modern Art, New York, New York, NY. pp. 26-35, 2001.
2. Raskar, R., G. Welch, M. Cutts, A. Lake, and L. Stesin, and H. Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. SIGGRAPH 1998.
3. Fox, A., B. Johanson, P. Hanrahan, and T. Winograd. Integrating Information Appliances into an Interactive Workspace. IEEE Computer Graphics and Applications, pp 54-65, May/June 2000.
4. Roman, M., C. Hess, A. Ranganathan, P. Madhavarapu, B. Borthakur, P. Viswanathan, R. Cerqueira, R. Campbell, and M. D. Mickunas. "GaiaOS: An Infrastructure for Active Spaces," UIUCDCS-R-2001-2224, University of Illinois at Urbana-Champaign, 2001.
5. Weiser, M. . The computer for the 21st Century. Scientific American 265(3): 66-75, 1991.
6. Schilit, B. N., Adams, N., Want, R. Context-Aware Computing Applications. IEEE Workshop on Mobile Computing Systems and Applications, 1994.
7. Salber, D., A. K. Dey, Abowd, G.D.. The Context Toolkit: Aiding the Development of Context-Enabled Applications. Proceedings of CHI'99, pp 434 - 441, 1999.
8. Dey, A.K.. Understanding and Using Context, to appear in Personal and Ubiquitous Computing, Vol. 5, 2001.
9. Lehman, T. J., Stephen W. McLaughry, and Peter Wyckoff. T Spaces: The Next Wave. Hawaii International Conference on System Sciences (HICSS-32), 1999.
10. Pinhanez, C.. The Everywhere Displays Projector. Ubicomp 2001, pp. 315-331, 2001.
11. Chou, P., M. Gruteser, J. Lai, A. Levas, S. McFaddin, C. Pinhanez, and M. Viveros. BlueSpace: Creating a Personalized and Context-Aware Workspace. IBM Research Technical Report RC 22281, 2001.
12. Steelcase Inc.. Alternative Officing Strategies. Grand Rapids, MI. 2000.
13. Open Services Gateway Initiative. OSGi Service Platform. http://www.osgi.org, 2001.