Position Paper for Workshop on Declarative Models of Distributed Web Applications (http://www.w3.org/2007/02/dmdwa-ws/)

Sailesh Sathish, Nokia Research Center, Nokia Corporation (sailesh.sathish@nokia.com)

Using declarative models for multi-device smart space environments

1. Introduction

A smart space is a multi-user multi-device dynamic interaction environment that is aware of its physical environment, working on top of heterogeneous radio technologies and software distribution platforms. Providing smart space capabilities in today's world would require supporting a heterogeneous environment within which devices and services communicate. This support would mean hiding the complexities of underlying network and communication technologies thereby abstracting the concepts to applications, end users and developers. Current research in Ubiquitous computing and smart spaces has concentrated on the following areas:

- User experience
  - The development of new models of interaction that incorporate the relationship of ubiquitous computing with the physical world
  - emergence of methods that focus on gaining richer understandings of settings
  - The development of approaches to assess the utility of ubiquitous computing
- System-support:
  - The development of system support technologies and methods for ubiquitous computing environments

While on one hand, each of these research activities enabled evaluation and realization of several ubiquitous-computing systems and prototypes, on the other hand technology-oriented system-support research and user-experience research have traditionally progressed separately and only a few works have recently tried to evaluate the impact of system-support solutions on usability and other HCI factors. From this viewpoint, smart spaces are quite a challenging example of ubiquitous computing systems to realize by using existing state-of-the-art solutions. Indeed, effective interaction of end-users with a real-world smart space requires novel solutions to design and realize technologically complex and easy-to-use smart spaces, i.e., requires user-experience and system-support research activities to be aligned and to cross-pollinate each other. By smart spaces, we mean environments defined within certain physical boundaries, providing a set of ambient data sources that can be used intelligently to provide certain services to the user. The ambient data sources that can exist within a smart environment can be smart devices that can form some sort of association with the environment and the user, sensors or objects that expose state values, and ambient networked services.

As an example, consider a user with a mobile device in a smart room. She uses the mobile browser to browse her favorite music site. She selects a list of songs and clicks on play. The media player loads on her browser. The browser window now show icon for a stereo player that is present in the room. She now notices the player and uses the volume control of the stereo to change the volume of the song running in the media player within the browser. She can also use the stereo stop, pause, forward controls to control the playback of the songs. Thus, intelligent systems within the room aid as tangible controls providing more natural interaction modes

We use the terminology "consumers" to indicate services or applications that use available data to perform some adaptive services to the user and the term "providers" that perform data provision to consumers. In a smart space environment, there can be different provisioning services that can range from collective data representations exposed by intelligent devices to unit data exposed by dumb devices or objects. We believe that there can be several consumers of the same data where the end objective logic resides within the consumers themselves. We also assume that providers and consumers work within heterogeneous environments with each talking different protocols and semantics bringing in the need for interoperability between the disparate systems. As such, it is imperative to provide an abstracted platform that is easy, efficient, intuitive and standardized in every sense.

In smart space environments, there can be at least two types of applications. The first type is the standard applications that run on devices that are capable of utilizing smart space components thereby providing enhanced or adaptive performance. Adaptation for such application can happen on three fronts such as: 1. Presentation adaptation: where the presentation of content is adapted according to context (such as multimodal) 2. Content adaptation: where the content itself is adapted according to context and 3. Service adaptation: where services provide dynamic behavior based on dynamic context. The second type of application associated with smart spaces is provided by the environment itself. These are generally tied to a particular smart space and not considered to be mobile. Such applications adapt based on current smart space context, user preferences and environment settings. These are characterized more by their flexibility to accommodate wide range of context, dynamic configurations, preferences, and privacy and security settings along with a uniform, modularized and extensible model for application development. Examples are a smart kitchen suggesting a menu based on available ingredients in the fridge or a system controlling lights within a work space based on day-time information and light detectors. Any standardized smart space platform thus, has to cater for the two approaches.

2. The blackboard approach and delivery context access

The essential part of any smart space environment is providing a mechanism for dynamic discovery and communication. The predominant approach for providing a smart space infrastructure is to provide a centralized communication hub where providers and consumers of data can communicate. We borrow from a similar design pattern originated from Artificial Intelligence (AI) called the blackboard concept where the blackboard is seen as a repository for processed and unprocessed data. The blackboard approach is meant for non-deterministic environments where a direct solution path to a problem cannot be pre-determined. The data providers put data into the blackboard while the consumers or processors can take this data; either consumes it or processes the data further and put it back on the blackboard. Here, by data, we mean context data and even though the environment does not reflect a true blackboard in the exact AI sense, we would need provision of a centralized data service with organized structures where abstractions can be constructed from lower levels.
Context data is mostly dynamic. Coupled with mobility, we can assume the environment within which information is solicited to be highly dynamic. Adaptive applications relying on contextual data need access to an efficient data access model. The data access model is analogous to the blackboard concept where the data comes from a non-deterministic environment. The role of the representation model is to organize provider data and provide support for consumer application access to those data. The representation model should provide uniform access methods for both consumers and providers to the maximum extend possible. The representation model needs to reflect the underlying dynamic nature of the environment it is operating in. This means that the model should cater for addition and removal of data sources and address dynamic value changes of these sources. In addition to an efficient access mechanism for both providers and consumers, the model should be capable of managing dynamic notifications to consumers. Since the dynamic nature of context data varies between sources, it is imperative that the model puts in adequate mechanisms through which controlled notifications can be sent to interested consumers. For highly dynamic sources, the model should provide a way to limit notifications as required by the calling consumer. This is difficult as in most cases the onus can be on the data provider. The provider can expose interfaces that support parameters for data rate control or provide solicited information only. On the other hand, consumers can control rate at which they are notified by embedding notification code within notification handlers they provide to the model. Another way would be to introduce a new filtering language that would be supported by the model but this needs standardization and may not span when multiple consumers demand different notification rates.
Context data and data sources can exhibit certain relations to one another. Any representation model needs to capture these relations well so as to facilitate semantic understanding and easier discovery.
The requirements for context representation model within smart spaces are:

- Support dynamic nature of environment
- Support discovery and subscription of services
- Reflect relations of objects within environment
- Provide support for consumers to easily search and find data
- Support dynamic values and value types
- Support for security and privacy issues
- Adequate support for provider services
- Enable single run-time instance facilitating data share

The W3C’s Delivery Context Client Interfaces (DCCI) provides a DOM-based interface for accessing context data from a client device. DCCI supports a hierarchical model for data access (as an interface thereby abstracting the actual data model underneath) and the DOM event model. DCCI is essentially a client consumer interface access model. The same can be extended to be the blackboard consumer access since it is possible to model smart space data as hierarchical. The level of standardization that DCCI has undergone and the provisions within DCCI interfaces to provide extensions counts in its favor. However, the biggest advantage is that if DCCI gets widely adopted within client devices as the de-facto standard for context access, building a composite capability by combining multiple DCCI trees (DOM trees) to form a blackboard central should be straightforward. Thus, the two obvious advantages are a standardized consumer access model for smart spaces and a mechanism for multi-device capability composition model.

The DCCI model does not specify a provider interface. A provider interface is needed for data providers to access and provide data to their respective representations within the DCCI tree. A standardized provider access interface is needed for both locally resident providers and remote data providers. Remote data providers would use client stacks corresponding to provider interface functionality for data provision services. Since DCCI follows the DOM model, using XPath expressions for access to data objects is possible. The providers can specify to the model where they want to provide data to. This would mean providing a valid XPath expression that can be resolvable within the DCCI context. A more efficient model can be provided if providers do not need to mandate a specific XPath expression to get an object reference for data provision. The blackboard (or client DCCI framework) can look at the provider description, refer to DCCI ontology and create an object representation within the DCCI model subject to permissions and validity. The framework can then provide just the object reference to the respective protocol stack or direct to local provider who can then use the provider interfaces for data provision. This mandates the need for a standardized DCCI ontology. Ongoing work within W3C and OMA to define a dynamic ontology for delivery context can be one suitable candidate. A particular concern related to having a standardized ontology is ontology management where updates and new properties including their metadata need to be accommodated. Security is another major hurdle that is yet to be addressed. Security policies can work together with ontologies where it may be possible to combine validation, security and access grant during session establishment between a provider and the context framework. Privacy policy and access management for consumer applications is another area that needs to be addressed.

Supporting multi-device environment and services would mean combining different DCCI trees (similar to DOM tree merging) into a composite tree structure. The underlying data models can be locally resident within the devices while the new tree provides a composite view and thereby a logical blackboard. There is thus the need for a DOM synchronization protocol. The W3C's Remote Events for XML (REX) activity is addressing the issue of remote DOM synchronization and an extension of this protocol can be used for DCCI tree synchronization. Synchronizing and combining different DCCI trees raises several issues such as validation of each node against a composite ontology, dynamic changes to nodes and topology, response times and network reliability. When considering merging of different trees, redundancy of different nodes within the trees need to be accounted. Finding redundant nodes may not be easy as the system needs to ascertain that the nodes are dissimilar. This means comparing not just the main node information but also the metadata as well as authenticate that the providers of those nodes are different. It is not clear at this point whether these issues need to be addressed at the protocol or framework level. The activity should also define a standard URI for the composite tree as well as mechanisms for discovery and access of the composite tree root node within a particular smart space.

Based on the above, the minimal requirements that should be addressed (within scope of UWA working group) are

- Address security models for delivery context
- Develop provider interfaces for DCCI with support for local and remote data providers
- Multi-device DCCI combination and synchronization mechanisms
- Recommendation of standard ontologies and ontology management

3. State machine approach for smart space applications

Ambient services that characterize smart space applications perform dynamic adaptations based on context changes. There can be several ambient services present within a smart environment satisfying specific use cases as well as variations depending on the type of smart spaces they operate in. The different types of smart space domains can be personal smart space, work smart space, home smart space, mobile smart space and public smart space with each smart space having their own domain policies and applications. Smart space applications, since they adapt based on environmental conditions can be seen as state machines where they occupy a certain state in time-space dimensions. Adaptation of a smart space application can be seen as state transitions triggered by a change in context where applications or services jump from one state to another. Each state would be characterized by state specific actions that can be rendered by ambient devices or services and the application code can be specific to the device in which it is run. Thus, the ideal smart space development environment should encompass a standardized framework for developing state machines that can utilize a centralized blackboard context mechanism. The framework should support running service specific code by delegating control to ambient platforms when a respective state is reached. The state transitions would be triggered by events that are fired due to context changes within the blackboard framework.

The W3C's State Chart XML (SCXML) is a general-purpose event based state machine language that can be an ideal candidate suited for writing smart space state machines. SCXML supports a standard set of actions that can be executed when a specific state is entered into. States are entered into through transitions that are triggered by events and valid guard conditions if one is available. It supports powerful state flows including composite states and parallel states. SCXML allows the use of custom actions through the common SCXML paradigm. It is the use of such custom actions that goes beyond the standard SCXML actions that make SCXML an attractive candidate for running ambient controls within a smart space. The specifics of SCXML mechanisms are beyond the scope of this paper.

To support a centralized context access model, the smart space development framework should support a uniform discovery mechanism for the context model. Any declarative markup that contains state machines should be able to add event handlers to specific objects of interest within the context model so that notifications can be handled. The event handlers can trigger the state transitions. Information about the target object that triggered the transition can be used to upload specific processing code either through a direct interface or done through the blackboard (similar to distributed processing). The specific type information of objects can be accessed through its metadata representation within the blackboard conforming to specific ontologies. The mechanisms and protocols for how interaction occurs between state machines, a DCCI-type context model and services that have representations within the DCCI model are not clear at this point and needs to be investigated.

To summarize, organized representations of context data within local devices can be combined in a multi-device smart space scenario to form a composite model that represents the collective capability of the environment. DCCI follows the DOM model for representing data objects in a hierarchical manner. Remote DOM synchronization protocol such as REX can be extended to support multi-tree combination and synchronization to build a composite tree. On the application development front, specifically for developing ambient services that belong to particular smart spaces, a state transition based approach could allow for creation of efficient, adaptive and dynamically extensible applications. The State Chart XML that is an event based state machine language can be combined with a centralized DCCI model as a candidate towards developing a standardized approach for smart space applications. There should be support for universal mode of discovery coupled with support for event handling and service invocations. Some of these can already be solved by combining multiple approaches such as XML events. However, several challenges that are known and unknown exists and needs to be overcome if an efficient framework for smart space application development is to be realized.

References

1. Delivery Context Client Interfaces (DCCI): http://www.w3.org/TR/DPF/
2. Remote Events for XML (REX): http://www.w3.org/TR/rex/
3. Blackboard system (Wikipedia source): http://en.wikipedia.org/wiki/Blackboard_system
4. Document Object Model (DOM): http://www.w3.org/DOM/