Supporting Smart Space Infrastructures: A Dynamic Context-Model Composition Framework

Sailesh Sathish
Nokia Research Center
Visiokatu 1
Tampere, Finland
+358504835679
sailesh.sathish@nokia.com

Cristiano di Flora
Nokia Research Center
Visiokatu 1
Tampere, Finland
cristiano.di-flora@nokia.com

ABSTRACT

Smart spaces are environments where intelligent devices can provide end-users with personalized and context-aware services. The biggest barriers against effective realization of smart spaces are the lack of interoperability between different ubiquitous computing infrastructures and standardized way of service operations. From this perspective, we present a framework for smart space interaction supporting interoperability and based on ongoing standardization efforts. The framework supports dynamic composition of context models based on multi-device and service interactions taking into account security and privacy issues. We show how we align ongoing W3C standardization efforts with key elements of the framework such as composition models and state-based mart space application development approaches. We conclude by providing our thoughts and plans about future extensions and interoperability needed between specifications for effective smart space realizations.

Keywords

Context models, smart spaces, adaptive applications.

1. INTRODUCTION

A smart space is a multi-user multi-device dynamic interaction environment that is aware of its physical environment and that works on top of heterogeneous networking technologies and software platforms. Smart spaces are defined by certain physical boundaries within which they can provide user applications with a set of ambient services. Ambient services are mostly particular to the domain of interaction and can characterize the smart environment the user is interacting with. Some examples of such services are tangible interactive objects, audio processing and routing systems, ambient video render, multimodal interaction services and environment control systems.

As an example, let us 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 shows an 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, she can exploit intelligent systems within the room as tangible controls providing more natural interaction modes.

Realizing effective smart spaces on top of existing ubiquitous computing standards and solutions is quite a challenging task due to the following main reasons:

- Smart space services are only a sub-set of the overall infrastructure of a smart space that applications should be capable of exploiting. Indeed, the smart space should be capable of enhancing existing services by composing multiple services together so as to add new composite capabilities dynamically to the smart space.
- Applications within smart spaces should be capable of detecting changes in context and to adapt themselves accordingly. Adaptation should regard contents, presentation and provided services. To support such different types of adaptation, the core of a smart space infrastructure should expose an easily accessible, efficient and standardized data access model that can be used by providers and consumers of smart space data for communication[1].
- Effective interaction of end-users with a real-world smart space requires novel multi-disciplinary 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.

We recognize that all the challenges above must be addressed in order to enable an effective framework for smart space application development. However, in this paper we discuss our experiences with addressing the first two ones, and we do not discuss the

MOBIMEDIA 2007, August 27-29, Nafpaktos, Greece

DOI 10.4108/ICST.MOBIMEDIA2007.1708

[1] We use the term "consumers" to indicate services or applications that use available data to provide adaptive features to the end-user- By "providers" we mean those services or applications that provide context data to consumers. Providers can range from intelligent data providers capable of taking raw data from sources such as sensors or modality processors providing higher level abstractions to low-level sensors providing raw data.

issues related to the required multi-disciplinary research. More specifically, in this paper we present a framework for smart space interaction supporting interoperability and based on ongoing standardization efforts. The framework supports dynamic composition of context models based on multi-device and service interactions, and it takes explicitly into account security and privacy issues.

This paper is organized as follows. Section 2 takes a brief look at related work within smart spaces and infrastructure support for context models. Section 3 introduces the delivery context model that forms part of the client framework for adaptive applications. Section 4 describes our framework in detail. In Section 5 we show how we aligned ongoing W3C standardization efforts with key elements of the framework especially with composition models and state-based approaches towards smart space application development. Section 6 concludes our paper by providing our view and plans about future extensions and interoperability needed between specifications for effective smart space realizations.

2. RELATED WORK

State-of-the-art research has proposed multiple approaches and insights into how context models and smart spaces interact, thus resulting in the creation of new architectural patterns. The work by Nixon et al. [7] on context adaptive smart spaces looks at providing context models to application adaptation. More specifically, it looks at the implications of providing context information on tiered architectural patterns. The functional interfaces between the layers change based on context, and the effect is reflected back on the presentation to the user. Integrating a context layer into a tiered architecture introduces an additional vertical layer as opposed to horizontal nature of tiered architecture. This is so because context models can provide input and output services uniformly to each layer and is independent from the general functionality the architecture is supposed to support. Other approaches utilize topographical models through use of overlapping spatial and temporal models [8] for smart space data access.

Related work by Chen et al. [2, 3] describes a context model through use of a brokering engine providing a shared model for context access. Here, standard web ontology languages are used for describing ontology in order to use a common vocabulary. Our framework supports ad-hoc build up of smart spaces through multi-device composition where individual models can be combined. The combined models, done by using standardized composition mechanisms, provide the core infrastructure for our smart space. The work by Christopoulou et al. [4] describes another model using ontology-based context management and reasoning for UbiComp applications.

In the area of inter-connected provision, several existing technologies aim at providing interoperable solutions for connected devices and services. For example, the Universal Plug and Play (UPnP) [16] specifications aim at enabling pervasive peer-to-peer network connectivity for personal mobile/stationary devices. The devices are able to connect to a network, advertise their capabilities as well as service descriptions and learn about other devices at run-time. Jini [5] is a Java-based technology for establishing Java-based networks through which users immediately access network resources and services. Salutation [13] is another technique for service discovery and service management. This offers an open, independent standard abstracting away from operating systems constructs, communication protocol or hardware restraints. Web Services is an example of a Service Oriented Architecture based on HTTP and XML and defines a suite of protocols aimed at data transfer, discovery and service description.

Our framework aims at providing an abstraction layer on top of such technologies mainly viewing them as provisioning systems. We support a central data sharing model for hosting representations of interconnected services enabling easier discovery and notifications of data within a smart space. Section 3 outlines requirements for such a central model and introduces a related ongoing standardization work within W3C called Delivery Context Client Interfaces (DCCI).

3. SMART SPACE CONTEXT MODEL

The smart space framework presented in this paper relies on a specific context-model related to ongoing W3C standardization efforts (the so-called **Delivery Context Client Interfaces** context access model). In this Section we briefly describe the main requirements of a smart space context model, and we show how W3C's delivery context model supports those requirements.

3.1 Context models requirements

Any generalized smart space environment built to support heterogeneous applications should support a common communications infrastructure. The same should also support discovery of data sources and cater for dynamic topology of environment. Towards this end, we take a cue from a design pattern that evolved from Artificial Intelligence (AI) called blackboard [1]. The blackboard is essentially a repository for processed and non-processed data where processors are allowed to build abstractions from lower layers. 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. Each processor can either consume or further process the data and put it back on the blackboard.

Going by the requirements of smart spaces, a data communication model that is needed is not exactly the blackboard in the AI sense. Most smart space applications follow a set of deterministic paths even though unpredictability may characterize the responses of the elements in the system. Also, the model needs to support certain structures capable of capturing relations between the data sources. However, the central concept of the blackboard in provisioning a platform for data repository, access and abstraction creation is valid here. Here, 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 also 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 put 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.

3.2 Delivery Context Client Interfaces

The World Wide Web Consortium (W3C) has been involved in standardizing a context access model specifically designed for client-side access. The specification, called Delivery Context Client Interfaces (DCCI) [7], follows a Document Object Model (DOM)-like interface for context access. The term "Delivery Context" [9] is defined as a *set of attributes that characterizes the capabilities of the access mechanism, the preferences of the user and other aspects of the context into which a web page is to be delivered.* DCCI is meant for web applications where scripts running within a web page can use similar DOM manipulation methods to access delivery context. DCCI represents context data as an ordered hierarchy in accordance with ontology. Ontology describes the terms, concepts and relations between the concepts for a domain. W3C is also working on ontology for delivery context. DCCI interfaces derive from DOM Element interfaces adding additional methods for searching properties, additional attributes and structures as well as DCCI specific events and exceptions. DCCI follow the DOM Event model [14] where calling applications can add event listeners that listen for particular events denoting changes to the DCCI tree. The listeners call event handlers that handle the event fired by a particular property node. The event model follows the same DOM event phases of capture, target and bubble. DCCI supports both static and dynamic properties and hence events are fired when either the value of a node changes or there is a topology change to DCCI tree.

As mentioned, DCCI is essentially a client consumer interface access model. The same can be extended to be a blackboard consumer access interface since it should be 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. The W3C is working on a remote DOM synchronization protocol aimed at synchronizing multiple trees called REX [12] that can be utilized here. Thus, the two obvious advantages would be a standardized consumer access model for smart spaces and a mechanism for multi-device capability composition model. It is also important that DCCI gets accepted as generic access interface for all applications and not just the web domain. Section 4 introduces our proposed framework for smart space infrastructure with the DCCI model forming the central access mechanism for common data communication.

4. SMART SPACE FRAMEWORK

Our proposed framework for smart space infrastructure is shown in Figure 1. The **Data Representation Model and Access** module provides the "view" of data that the framework exposes to consumers. The adopted data representation model is compliant with the DCCI mechanism described in Section 3.2. The data representation model only provides the view and access to the model while the actual data can be locally stored or distributed within the network indicated by the **External Data Repository** in Figure 1. Data within the representation model are managed by the **Data Manager** component. Another function of the data manger is to perform intelligent abstractions and aggregation of data provided by data providers. The abstractions can be constructed based on semantic rules provided by users, developers and other services. The **Data Abstraction** module is responsible for building abstraction data objects utilizing certain rule sets, domain constructs and raw data objects.

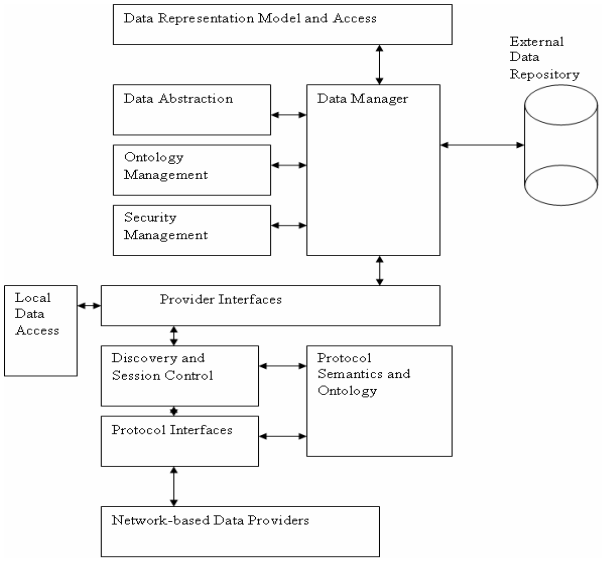

Figure 1: Smart Space Infrastructure Framework

The domain-specific constructs and primary data objects are provided by the **Data Manager**, and they comply with smart space ontology. The adopted ontology describes concepts and relations of data objects within a domain. The ontology can contain information about what abstractions are possible within a domain including where and how the abstractions can be represented within the data representation model. Since representing all possible abstractions within ontology was not possible and would have required complex dynamic ontology management (which is outside the scope of this paper), the ontology provides descriptions of "allowable" abstractions using a known set of base constructs. The information about how the actual abstractions are constructed using the allowed base constructs can be represented through different sets of semantic rules. When dynamic rule sets are created, the **Data Manager** can ensure integrity of the representation model by working simultaneously with the ontology constructs to ensure that the newly added rule sets are valid and allowed by the ontology.

The **Ontology Manager** is responsible for maintaining and managing the ontology while providing the ontology interface to the **Data Manger**. The **Security Management** module is responsible for providing security and authentication services for the framework. This module depends on security and access policies provided through authorized channels. The security model is discussed in more detail in Section 4.1.

Provider Interfaces enable all providers with a unified entry point to access the **Data Representation Model**. Depending on the granted access type, the provider interface can constrain the view to externals. Local data can directly use the provider interface, while network based services would use some protocol languages for communication. For network-based providers, the provider interface can convert the corresponding object pointers to a unique identifier that can identify the provider. This would abstract the model view from providers ensuring blocking system data space from external services. Moreover, since the ontology determines where within the data representation model the provider can provide data to, providers are spared the complexity of knowing where exactly to provide their data to. Providers are granted access based on their service description, metadata and fit to the domain ontology.

The **Protocol Semantics and Ontology** module provides support for translations between different protocols that operate in inter and intra-domain mode. The discovery and session control module is responsible for discovering services and session establishment based on ambient services discovered. The protocol ontologies provide ontological description of semantics employed by each protocol used for communication.

4.1 Security

As part of the framework, we have developed a security framework based on classification of consumers. The security policy deals with mapping each provider and consumer to a particular class where specific rights have been assigned for each type of class. The solution aims at minimizing the risk of supplying invalid or incorrect information to the calling application or of creating bogus properties within the framework by malicious programs.

The policy involves utilizing the metadata interface of DCCI node interface that gives additional information about the property. The security module exposes security classes, which define security rights for components. The term "component" refers to a software program that can be a consumer or provider, an application, or a physical component (such as sensors, audio modules). The components need to register to one of the classes according to a class identifier, which is assigned to the components by an operating system or a middleware component.

For external service providers, the class identifier should be obtained either through a third party assignment service or via direct negotiation with device middleware component through metadata verifications. The exact formats and process for class identifier procurement is out of scope of this paper. The class identifier is generated in such a way that one part of the identifier will determine which class is in question, and the other part of the identifier uniquely identifies the component or application within said class. The class identifiers are used to determine, which class the programs can be registered to.

The components will register to one of these classes based on the specific priority (class identifiers) that has been assigned to them. The security module will look at the particular class to which a component belongs, and then grants the appropriate rights accordingly. Sometimes, it may be required that the class right itself be modified based on certain situational requirements. This means that even the class rights have to be managed. To deal with this, each class is given a set of default rights.

In addition to the default rights, each class is also associated with a modifiable schema that can override the default behavior. The schema will be maintained by the security component and each interaction request will be validated against the schema before execution. The schema can be edited by the user, operating system or underlying implementation. The classification of specific classes, the components that belong to each class and rights associated with each class is outside the scope of this paper.

4.2 Implementation

As part of framework development, we implemented an instance of the proposed context model. The implementation is based on a previous specification of Delivery Context Client Interfaces that was then called Delivery Context: Interfaces (DCI). The main difference between DCI and DCCI lies in that while DCCI interfaces derive from DOM Element interfaces, the DCI interfaces derived from DOM Node interfaces. The change was made due to serialization requirements that were identified for certain use cases by the working group. The implementation was carried out as an extension to Mozilla Firefox browser. On the data provisioning front, we have integrated a SIP-event based provisioning system with the framework utilizing a mock taxonomy of data provider representations for client applications. Due to space constraints of this paper, we refer to [10] and [11] for more detailed information on implementation and the provider framework.

5. ADDRESSING SMART SPACE REQUIREMENTS

The delivery context model and smart space framework described in previous sections can provide consumer applications with an infrastructure for accessing system and environment information. The smart space environment and its existing services can be augmented by exploiting composite capabilities of multi-devices that operate within the environment. The individual capabilities of devices should be combined along with smart space services to provide applications with a single logical view to a composite capability. This would create new possibilities for adaptive applications. For example, an application running on a unimodal platform would be able to support multi-modal interaction when a device supporting speech recognition resource is discovered. We also need an effective way to model smart space applications capable of supporting and reacting to dynamic aspects of composite models. Section 5.1 describes how the data representation model and access module in Figure 1 translates to composition models when migrating to multi-device environments. Section 5.2 looks at state-machine based approach to smart space application development while outlining some requirements for existing standards addressing this domain.

5.1 Composition Models

DCCI supports a tree model for data representation and access. Hence, in theory it is possible to combine and merge multiple DCCI trees together. 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.

5.2 Application Modeling

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. 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 data model. The W3C's State Chart XML (SCXML) [13] is a general-purpose event based state machine language that can be an ideal candidate suited for writing smart space state machines. SCXML allows using custom actions through the common SCXML paradigm. It is the use of such custom actions, that goes beyond the standard SCXML actions, that makes 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 need to be investigated in more detail.

6. CONCLUSION

In this paper we discussed how we extended our existing framework for delivery context model and access in order to support development of applications for smart spaces. We proposed an approach to achieve composition and interaction of services across multiple devices based on building and describing composite smart space capabilities by assembling individual context models together. Our context model uses the W3C DCCI representation. DCCI represents data as tree structure where merging trees would address composition requirements. The W3C REX specification can be modified from simple serialization of DOM nodes for tree synchronization to address issues such as node redundancy, dynamic sessions and logical composition creation issues that can be raised when supporting smart space specific data models. On the application authoring front, we view smart space applications as state machines that can utilize context models. The W3C SCXML specification can be used as a candidate for authoring such applications with extensions for smart spaces. For effective and widespread adoption of smart space concepts, we believe that the approach of combining multiple standards together with appropriate extensions should go a long way in its realization.

7. ACKNOWLEDGMENTS

The authors thank fellow colleagues at Nokia Research Center and W3C for their vision and support towards this work.

8. REFERENCES

[1] Frank Buschmann,, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-Oriented Software Architecture: A System Of Patterns. West Sussex, England: John Wiley & Sons Ltd., 1996

[2] Harry Chen, Tim Finin and Anupam Joshi, An Intelligent Broker for Context-Aware Systems, poster paper for UbiComp 2003, October 12-15, Seattle, Washington.

[3] Harry Chen, Tim Finn and Anupam Joshi, An ontology for context-aware pervasive computing environments. In: The Knowledge Engineering Review, archive volume 18(3), September 2003.

[4] Eleni Christopoulou, Christos Goumopoulos and Achilles Kameas, An ontology-based context management and reasoning process for UbiComp applications. In: Proceedings of the joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies, 2005.

[5] Jini Network Technology, available at www.sun.com/jini

[6] Keith Waters, Sailesh Sathish, Rafah Hosn, and Dave Ragett, Delivery Context: Interfaces (DCI) Accessing Static and Dynamic Properties, World Wide Web Consortium Candidate Recommendation October 2006. Available at: http://www.w3.org/TR/DPF/

[7] Paddy Nixon, Simon Dobson, Sotirios Terzis and Feng Wang, Architectural Implications for Context Adaptive Smart Spaces. In: *IEEE International Workshop on Networked Appliances*. 2002.

[8] Rene Meier, Anthony Harrington, Thomas Termin and Vinny Cahill, A Spatial Programming Model for Real Global Smart Space Applications. In: Proceedings of Distributed Applications and Interoperable Systems, 6th IFIP WG 6.1 International Conference, DAIS 2006, Bologna, Italy, June 14-16, 2006.

[9] Roger Gimson, Sailesh Sathish and Rhys Lewis, Delivery Context Overview (DCO) for Device Independence, W3C Working Group Note, March 2006. Available at: http://www.w3.org/TR/di-dco/

[10] Sailesh Sathish and Olli Pettay, Delivery context access for mobile browsing. In: Proceedings of *International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, August 2006, Bucharest, Romania.

[11] Sailesh Sathish, Dana Pavel and Dirk Trossen, Context service framework for the mobile Internet. In: *Proceedings of International Workshop on System Support for Future Mobile Computing Applications (FUMCA2006)*, in conjunction with Eighth international conference on Ubiquitous Computing (UbiComp), September 2006, Irvine, California.

[12] Remote Events for XML 1.0, available at : http://www.w3.org/TR/rex/

[13] Salutation consortium, specifications available at http://www.salutation.org/whitepaper/bylaws.pdf

[14] State Chart XML, available at: http://www.w3.org/TR/2005/WD-scxml-20050705/

[15] Tom Pixley, Document Object Model (DOM) Level 2 Events Specification, W3C Recommendation November, 2000. Available at: http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/

[16] Universal Plug and Play Specification (UPnP), available at www.upnp.org