A Framework for Mobile, Context-Aware Applications

Suparna De and Klaus Moessner

Abstract— The heterogeneous, dynamic nature of current mobile environments necessitates that all system components that form part of a personalization framework should be context aware. This necessitates the development of a framework of distributed context handlers to derive meaning from context and combine it with application logic. Towards this aim, this paper presents a Service Context Manager (SCM) framework that handles all the stages of context gathering, processing and reasoning to enable personalized service presentation. The validation of the approach is shown through application use cases.

Index Terms—context-aware framework, personalization, semantic technology.

I. INTRODUCTION

THE vision of provisioning services that are relevant to the end-user in a given situation requires that ubiquitous communication environments must be able to collect a wide range of context information and use it. The heterogeneity of the possible context information sources necessitates the development of a framework of distributed context descriptors to manage the varied sources. The framework should support the entire cycle of context definition, acquisition and inference to provide users with relevant and best-suited service presentation.

The scope of context information considered in this paper covers multimodal device capability information as it is an important precursor to realization of customized service delivery. To achieve context-awareness, the framework must produce reliable information in the presence of rapidly changing, often, partially available data from multiple heterogeneous sources. Fusing data from multiple sources to derive meaning from it, can prove additionally challenging. This is particularly true when the context information obtained from different devices (e.g. mobile phone, computer and printer) has different structures and levels of granularity. Moreover, all the relevant steps of context acquisition, monitoring and reasoning, need to be part of a fully automated

framework to minimize user distractions.

To address these challenges, this paper presents a Service Context Manager (SCM) framework that handles all the stages of context gathering, processing, inferring and reasoning to come up with useful recommendations.

The paper is organized as follows: framework types and the current state of the art are analyzed in section II. Identified requirements are outlined in section III. Section IV presents the SCM framework with its various functional modules. This is followed by showcasing of implementation aspects, with a reference application scenario, in section V. Design principles and choices are discussed in section VI, together with a comparison of the developed framework with some examples from literature. The paper concludes with a summary and recommendations for future work in section VII.

II. RELATED WORK

Context-aware systems have received much research attention in the past decade. Initial efforts were typically aimed at building sensor and device-rich smart spaces with accompanying new devices and software [1, 2]. Other approaches have focussed on specific applications of the context-aware system, such as tourist guides [3] or general mobility-aware personal recommenders [4].

In contrast to these tightly-coupled systems, the approach presented in this paper is a framework that makes useful recommendations based on inferred context. The framework coordinates multiple inter-operating components and provides support structures that facilitate context-sensitive service provisioning to a user. Approaches that provide a common architecture for the capture, interpretation and dissemination of contextual information can be categorised as: widget-based, infrastructure and blackboard systems.

A. Widget based

A widget or an interactor is an abstraction that facilitates the separation of application semantics from low-level input handling details. The widget interaction method provides both a querying and a notification, or callback mechanism, to allow applications to obtain input information as it occurs. All widgets have a common external interface within a toolkit, allowing applications to deal with all the widgets in a similar fashion.

The most often cited implementation based on this

Manuscript received December 31, 2008. This research has been funded by the Industrial Companies who are Members of Mobile VCE, with additional financial support from the UK Government's Technology Strategy Board (previously DTI).

Suparna De and Klaus Moessner are with the Centre for Communication Systems Research, University of Surrey, Guildford, United Kingdom. (phone: 44-1483-683429; fax: 44-1483-686011; e-mail: {S.De, K.Moessner}@surrey.ac.uk).

framework is the *Context Toolkit* proposed by Dey et al [5]. However, this approach requires complex configuration. Also, the Context Toolkit requires a special lab-like environment and specific hardware. Another implementation [6] provides context information service to applications. This approach requires that the mappings between equivalent terms be set up for correct query execution. Also, as the authors themselves note, this architecture has issues of usability and scalability.

B. Service Infrastructure

Hong and Landay [7] advocate a service infrastructure approach that shifts most of the context aware computing onto network accessible middleware infrastructures. The middleware is aimed at providing uniform abstractions and reliable services for common operations.

Some of the infrastructure middlewares address heterogeneity of context sources by providing tools for validating context models and separating models from implementation by generating stubs for different programming languages [8, 9]. The Gaia infrastructure [10] consists of context providers, consumers and synthesisers to form new context from existing ones. Logic is the context modeling and inference language. As is also noted in [11], this has the advantage of expressiveness and formality, but suffers from inference inflexibility and uncertainty handling. Also, this model has a smart-space approach as opposed to mobile-device centric.

C. Blackboards

In this approach, messages are posted to a common shared message board. Processes can subscribe to receive messages that have been posted and match a specified pattern. The nature of pattern matching varies between different systems.

Many context-aware frameworks in the field of mobile communications implement different versions of the blackboard approach. These are typically characterised by a context manager that embodies the blackboard and other supporting components, including applications that act as clients to the context manager. The context manager contains the functionalities of context processing, reasoning and providing query interfaces to client applications. For example, Context Broker Architecture (CoBrA) [12] is an architecture for supporting context-awareness in smart spaces. This architecture assumes each smart space to have a designated central context broker for its operation and has a rigid structure. As reported in [13], this framework does not stress on deduction of context through rule engines. Most implementations also do not take into account missing or imprecise context information nor address customization. User oriented customization is the aim of the Device Collaboration System (DCS) architecture of [14]. It probes the ambient environment for usable devices that can be used for service presentation by matching device profiles with service requirements and user history. However, resource description does not take into account the software elements of a device (e.g. file formats supported by the device monitor); only the

hardware elements are considered during resource recommendation.

III. REQUIREMENTS

The proposed Service Context Manager framework participates in both the context management and decision making process to streamline the user's interaction with the devices and services in the communication environment. From an examination of the related state of the art, the requirements placed on such a framework can be identified as follows:

Find and provide access to context sources – this involves determination of available devices in the ambient environment and recognition of services supported on the discovered devices. The services pertain to modality support in the environment and include both the software descriptions and the related hardware capabilities, along with service access mechanisms.

Context monitoring - this requires definition of meaningful update intervals, updating mechanisms as well as flexible distribution to decision making and caching points in the network.

Transparent use of distributed context information - the discovered context information should be made available at a higher abstraction level than that provided by protocol specific description formats. The abstracted context information should be easily exchangeable between reasoning processes and application logic.

Context information interpretation - the context information should be seamlessly processed to derive a usable semantic piece of information.

Facilitate service presentation – the framework should enable reasoning that makes available useful recommendations to facilitate relevant and best suited service presentation to the user.

IV. SCM FRAMEWORK

A. Framework Approach

A comparison of the different framework types is given in [15, 16]. Based on an analysis of the state of the art, the SCM framework has been modelled on the blackboard approach. The arguments for this choice can be summarised thus: both the widget and infrastructure model emphasise direct communication between the context source and consumer, with the consumer having to contact each source separately. This communication is abstracted out in the blackboard model. Also, since data has a uniform structure, flexibility of providing context data from any source is assured, without separate search and connection establishment. Commonality of data structures also enables utilisation of inference engines that can operate on the context data from all sources. The inference engine operation can also be kept transparent to the client applications.

B. Framework Components

This section presents the aims and roles of each component

and how each of these contributes to the overall aim of context management, coordination and semantic operability. Figure 1 shows a schematic representation of the functional blocks in the framework and their subcomponents.

Fig.1 Service Context Manager architecture

1) Device and service discovery function

This component is primarily concerned with discovery of resources in the ambient environment. This relates to discovering the availability of devices, enhanced by the device capability and content support determination. Network connectivity specifications tied to specific media output modalities are also determined. The resource discovery component exposes interfaces for synchronous context information retrieval and an asynchronous event-based interface. Synchronous events such as a device joining the network and exiting are multicast on the network interface and clients monitor this interface to maintain dynamic device-modality availability. Thus, it provides the dual functionality of asynchronous discovery and network monitoring for device announcement and exit. The physical device discovery is followed by retrieval of associated context information (device descriptions). The extended UPnP protocol [17] is used for this discovery step as well as for the retrieval of the XML device descriptions. The descriptions, which consist of a main file and separate XML description files for each hosted service, are retrieved through HTTP GET commands.

2) Transformation Framework

This component is concerned with aggregation of distributed context information and abstracting it into a common, formal structure to provide transparent access to all collated information.

a) Context Processing Module

The Profile Integrator aggregates all the gathered contextual information into a single document. This includes information obtained from the discovery function and also that which may be held in external repositories (e.g. device manufacturer supplied information available on a website). Links to such external information are obtained while parsing the context data obtained from the discovery phase. The external

descriptions are retrieved, parsed and merged with the collated context information.

The Context Formatter module serializes descriptions into a form amenable for direct input into the transformation engine. This may include namespace processing, where context information annotated in different namespaces (e.g. RDF) is extracted into plain XML elements for direct manipulation by the transformation engine. The output of this step is a DOM (Document Object Model) structure.

b) Transformation Module

The serialized context information is mapped into a common, formal structure based on the defined ontology stored in the OWL Facts base. The OWL Facts base constitutes the context model in the framework and represents the domain in terms of classes and properties. Thus, it forms the vocabulary of the system, also known as the 'TBox' or the concept box. The design of the context model is published in [18]. The transformation of the varied context data into a semantic form is accomplished by applying scripts matched to the input description format to transform the XML context to an OWL ontology instance. XSLT (XSL Transformation) is employed for the mapping. The relevant XSL script is matched according to the input description format (e.g. UAProf profile or UPnP description). The selected XSLT template is then applied to the DOM object. The transformation algorithm employs XPath (XML Path Language) expressions to select XML nodes. For each matched XML node selected from an XPath expression, either an instance of the mapped OWL class is created or an object or datatype property is added between corresponding individuals. The resultant OWL instance file embodies knowledge of real-world objects in TBox-compliant statements, thus forming the ABox (assertion component).

3) Reasoning Module

This is a hybrid inference and control system. The first stage of context inference asserts missing context information. The design of this component takes into account the standard device-service context description templates. Due to this heterogeneity of context information, there may be gaps in the generated ABox. Hence, at a first instance, the modeled rules assert links between the physical device and hosted software services. This pre-processing step is transparent to the user and speeds up the next stage of context filtering. The Context Filtering module then applies the rules from the SWRL (Semantic Web Rule Language) [19] Rules Base. The rules take as input the stored user preferences and match incoming content metadata to the processed device modalities' ABox to facilitate content presentation with the best possible combination of modalities. For instance, stored preferences may indicate which device the user would like to 'see videos on'. The content metadata relating to content type and other factors such as resolution, frame rate (for media content) etc. serves to match modalities to content. A query interface gives applications access to the reasoned and filtered context. The

queries have been designed through the SQWRL (Semantic Query-Enhanced Web Rule Language) [20] built-in library mechanism. The choice of the query offers the twin benefits of querying ontology terms directly and also being cognizant of the defined rules during query execution.

V. APPLICATION USE CASES

A demonstrator has been built as a proof of concept of the design principles of the framework. This section aims to show its working through selected use cases that highlight the overall information flow and message exchange between the various modules of the framework.

The demonstrator scenario involves a user who is watching the news on his TV. When he is about to leave home, he switches off the TV. Since he is carrying his PDA with him, he is shown an option to continue watching the news on his PDA. When the user selects this option, the news is streamed to his PDA. This scenario demonstrates a changing view of the ambient environment due to user mobility. The dynamic view of the available devices and their modalities is shown to the user through a GUI. Table I shows a step-by-step breakdown of the functionalities required.

TABLE I. SCENARIO ANALYSIS

Scenario	Scenario step	Functionality required
User mobility	User is watching news (video) on his TV	Service discovery
	User switches off the TV	<ol style="list-style-type: none"> 1. Cognition of device and modality unavailability 2. Update available device, modality list 3. Service query (video supporting devices) 4. Matching of content type (video) to device modalities 5. Ranking of available devices based on user preferences
	User chooses to watch TV on the PDA	Convey device selection to content delivery interface

The sequence diagrams in Figures 2-4 show the various steps involved in this scenario. These diagrams show only the SCM components involved; the interaction with other system entities, viz. the Content Delivery interface, is not shown here.

Fig. 2. Sequence flow – content selection

When the user selects a content type (video corresponding to the news service), the content type (video) and its format information is passed on to the OWLManager. The OWLManager interfaces to the ambient environment context model. The content metadata is then input into a SQWRL query which is presented to the query engine. The result of this query, which is a list of the available devices supporting the content, is handed over to the SWRL rule engine. Here, the device list is ranked according to the stored user preferences and other criteria (e.g. device monitor resolution). This re-ordered list is handed back to the OWLManager, which presents it to the user on the GUI.

The sequence of events that are triggered when the user switches off the TV is shown in Figure 3.

Fig. 3. Sequence flow – device unavailable

When the TV is switched off, the UPnP service running on it sends out a bye message. This is multicast on the network and is received by the UPnP client. The TV and its associated display are, as a result, marked as unavailable for content display. However, since the user should have consistent service access, the framework then automatically tries to re-route the service to another compatible device. The discovery adapter searches for available devices and their supported modalities. The device context obtained from this process is handed to the context formatter, which maps it to the ontology model. The updated OWL file is reloaded by the OWLManager. The application of content matching and user preferences is depicted in the sequence in Figure 4.

Fig. 4. Sequence flow – device recommendation

The sequence of information and control flow after the OWL model is updated is similar to that when the user selects the news service in the first instance. The content type is matched against the updated ambient context and then the recommendation list is formed by applying the SWRL rules. Since the available device (PDA) can show the news stream, it is presented as a recommendation to the user.

URL where the device capability information is stored. This is retrieved by the framework and fed into the transformation module.

Table II provides a taxonomy of the features of a context management framework and compares the developed framework with some examples from current literature that are illustrative of the different types of context frameworks.

TABLE II. CONTEXT-AWARE FRAMEWORKS: FEATURE COMPARISON

Framework	Framework type	Context model	Resource discovery	Intelligence	Applicability
Context toolkit [5]	Widget	Attribute-value tuples	Discovery widgets	Context interpretation and aggregation, query by attributes	Simple design for development and deployment of context-aware services
CoBrA [12]	Java agents, centralized broker	OWL ontology	Context acquisition module	Reasoning over OWL schema, rules and inference engine	Smart spaces
Gaia [10]	Infrastructure	Logic-based	CORBA naming and trading service	Logic-based	Smart spaces
CMF [11]	Blackboard	RDF ontology	Context recognition service	Feature detection and fuzzy logic to build higher-level concepts from sensor data	API for sensor-based context-aware framework
DCS [14]	Blackboard	RDF tuples	Resource discoverer	Rule-based recommender service with device and user profile	Personalized recommender system
Proposed framework	Blackboard	OWL-DL ontology	Discovery adapters	Rule-based reasoning for missing context + recommendations with device, service, content metadata and user preferences	Personalized, context aware service provisioning

VI. DISCUSSION

The use of Semantic Web technologies for context modeling and exchange not only offers a common, formal structure to the heterogeneous information inherent in current communication environments, but also enables application reasoning about service offerings. This follows similar recommendations for use of semantic technologies set out in the literature [8, 21, 22].

The design of the transformation component addresses the efficiency issue outlined in [23]. This concerns the minimum of two hops that are involved in every communication through a blackboard. To overcome this, [23] suggests that the context data should be processed to a sufficient abstraction level before adding it to the blackboard (context store). This pre-processing of context is handled in the context processing sub-module prior to mapping into the context model. It also incorporates methods for abstracting context from multiple context sources. The working of the transformation component is however, based on the assumption that context data is available in a XML-based form.

The framework supports two different models of context acquisition: active advertisements and passive discovery. Active advertisements from devices, for instance in a networked home environment, involves devices hosting their own descriptions. Passive retrieval comes into play when the discovered device holds a pointer to a manufacturer supplied

VII. CONCLUSION

The Service Context Manager presented in this paper offers a semantic, flexible and intelligent context-aware framework to enable an automated, personalized user experience. By enabling automatic and dynamic device selection, the mechanisms developed can plug into content delivery [24] and media adaptation systems [25].

Dissimilarities in the various possible context structures are handled through rule-based inference. However, this framework does not include mechanisms to model uncertainty or confidence levels. The framework would also benefit from learning mechanisms that can provide support for anticipatory responsiveness to changing context.

ACKNOWLEDGMENT

The work reported in this paper has formed part of the Ubiquitous Services Core Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com. Fully detailed technical reports on this research are available to Industrial Members of Mobile VCE.

REFERENCES

- [1] T. Kindberg, "People, Places, Things: Web Presence for the Real World," in *Proc. 3rd IEEE Workshop Mobile Computing Systems and Applications (WMCSA 2000)*: IEEE CS Press, 2000.

- [2] D. Garlan, "Project Aura: Towards Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, vol. 1, pp. 22–31, 2002.
- [3] K. Cheverst, K. Mitchell, and N. Davies, "Design of an Object Model for a Context Sensitive Tourist GUIDE," *Computers and Graphics*, vol. 23, pp. 883–891, 1999.
- [4] O. Riva and S. Toivonen, "A Hybrid Model of Context-aware Service Provisioning Implemented on Smart Phones," presented at ACS/IEEE International Conference on Pervasive Services, 2006.
- [5] A. K. Dey, D. Salber, and G.D.Abowd, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Special Issue of Human Computer Interaction*, vol. 16, pp. 97–166, 2001.
- [6] R. Power, D. Lewis, D. O'Sullivan, O. Conlan, and V. Wade, "A Context Information Service using Ontology-Based Queries," in *Proc. First International Workshop on Advanced Context Modelling, Reasoning, Management held in conjunction with The 6th International Conference on Ubiquitous Computing (UbiComp'2004)*. Nottingham, England, 2004, pp. 7.
- [7] J. I. Hong and J. A. Landay, "An Infrastructure Approach to Context-Aware Computing," *Special Issue of Human Computer Interaction*, vol. 16, pp. 287–303, 2001.
- [8] R. C. A. de Rocha and M. Endler, "Middleware: Context Management in Heterogeneous, Evolving Ubiquitous Environments," *Distributed Systems Online, IEEE*, vol. 7, pp. 1–13, 2006.
- [9] K. Henriksen, "Middleware for Distributed Context-Aware Systems," in *On the Move to Meaningful Internet Systems*, vol. LNCS 3760: Springer, 2005, pp. 846–863.
- [10] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing*, vol. 1, pp. 74–83, 2002.
- [11] P. Korpipää, "Blackboard-based software framework and tool for mobile device context awareness," Doctor of Science in Technology dissertation, Faculty of Technology, University of Oulu, 2005, pp. 229.
- [12] H. Chen, T. Finin, and A. Joshi, "Using OWL in a Pervasive Computing Broker," in *Proceedings of Workshop on Ontologies in Open Agent Systems (AAMAS 2003)*, 2003.
- [13] A. Singh and M. Conway, "Survey of Context aware Frameworks - Analysis and Criticism," University of North Carolina on Chapel Hill, Information Technology Services, 2006.
- [14] J.-H. Park, W.-I. Park, K. Young-Kuk, and K. Ji-Hoon, "Personalized recommender system for resource sharing based on context-aware in ubiquitous environments," in *Proc. International Symposium on Information Technology, ITSIM*, vol. 4, 2008, pp. 1–5.
- [15] T. Winograd, "Architectures for Context," *Special Issue of Human Computer Interaction*, vol. 16, pp. 401–419, 2001.
- [16] M. Baldauf, S. Dustdar, and F. Rosenberg, "A Survey on Context-Aware systems," *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 2, pp. 263–277, 2007.
- [17] S. De and K. Moessner, "Device and Service Descriptions for Personal Distributed Environments," in *Proc. 2nd IEEE International Conference on Digital Information Management (ICDIM'07)*, vol. 2, pp. 695–700, Lyon, France, 2007.
- [18] S. De and K. Moessner, "Semantic Description of Multimodal Devices: Modelling and Evaluation," in *Collaboration and the Knowledge Economy: Issues, Applications and Case Studies*, vol. 5, *Information and Communication Technologies and the Knowledge Economy*, P. Cunningham and M. Cunningham, Eds.: IOS Press, 2008, pp. 1439–1446.
- [19] W3C, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," in *W3C Member Submission*, 2004.
- [20] ProtegeWiki, (2007, November 19). SQWRL [Online]. Available: <http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL>.
- [21] A. Ranganathan and R. H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," in *Middleware 2003*, vol. 2672/2003, *Lecture Notes in Computer Science*. Berlin / Heidelberg: Springer, 2003, pp. 998–1018.
- [22] M. J. van Sinderen, A. T. van Halteren, M. Wegdam, H. B. Meeuwissen, and E. H. Eertink, "Supporting context-aware mobile applications: an infrastructure approach," *IEEE Communications Magazine*, vol. 44, pp. 96–104, 2006.
- [23] P. Korpipää, J. Mantjarvi, and J. Kela, "Managing context information in mobile devices," *IEEE Pervasive computing*, vol. 2, pp. 42–51, 2003.
- [24] C. Hesselman et al., "Delivering Interactive Multimedia Services in Dynamic Pervasive Computing Environments" presented at International Conference on Ambient Media and Systems (AmbiSys), Quebec, Canada, 2008.
- [25] A. Attou and K. Moessner, "Context-Aware Service Adaptation Management," presented at 18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC, Athens, Greece, September 3–7, 2007.