A UBIQUITOUS SYSTEM USING SEMANTIC KNOWLEDGE IN DYNAMIC SERVICE INTEGRATION

Tomasz Dziubich, Paweł Kaczmarek and Sławomir Nasiadka
Department of Computer Systems Architecture, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12 Str., 80-233 Gdańsk, Poland

Keywords: Ubiquitous systems, Service integration, Semantic information, Workflow systems.

Abstract: The paper presents a ubiquitous system that uses semantic information to manage dynamic changes in system elements. We design the system using the Context-Controller-Action architecture that covers the main parts of ubiquitous environments, including: environment devices (such as sensors and actuators), contexts and actions, all of which may be dynamically configured during system operation. A graphical user interface for the system is supplied, which enables non-expert users to perform changes in system configuration. Internally, semantic information is used throughout the system to define a common language between devices, contexts and services. A Web services interface is used in services that handle environment devices to support their integration with the system regardless of internally used communication protocols. The architecture was implemented as a framework for developing and running applications in ubiquitous environments. We integrated a number of devices in the framework, including, RFID sensors, digital displays, cameras, electronic door locks and others. We also defined exemplary contexts and actions in the environment.

1 INTRODUCTION

Ubiquitous systems significantly extend everyday life environments with an ability to dynamically adapt to changing environment conditions and user behavior. The conditions and behavior are detected by dedicated sensors (Poslad, 2009) that supply relevant information for the ubiquitous system. The information is used to determine the current context (Loke, 2006) (Salber et al., 1999) of the environment and to perform various real-world actions using actuators. Examples of performed actions include door unlocking depending on an identified user, modifications of digital display contents, and temperature adjustment.

The emerge of ubiquitous systems opened new challenges and requirements as compared to traditional desktop computing. Systems must be able to dynamically manage changes that occur both in the environment and in a system structure. The nature and the result of environment actions should change depending on user preferences or availability of services. For example, a user might expect a system to change its behavior because of the holiday season, which has not been anticipated earlier. Additionally, the dynamic changes of system behavior should be managed by users who usually do not have advanced knowledge about software development.

Considering the requirements, we propose a system that supports both dynamic management of resources and a user-friendly approach to system programming. Dynamic management of resources is realized by the Context-Controller-Action architecture that integrates different system elements, such as sensors, context definitions, available actuators and actions. Context and action definitions are managed by a user-friendly interface adequate for non-expert users. We shift from traditional coding-based application development to declarative-based development supported by a graphical user interface. Internally, we use semantic information and Web services to improve seamless integration and interoperability of system elements. Semantic information is used throughout the system to define a common language between devices, contexts and actions.

2 SYSTEM ARCHITECTURE

We propose to use the Context Controller Action (CCA) architecture, as shown in Fig. 1. In the architecture, a system is composed from the following main modules:

Dziubich T., Kaczmarek P. and Nasiadka S..
A UBIQUITOUS SYSTEM USING SEMANTIC KNOWLEDGE IN DYNAMIC SERVICE INTEGRATION.
DOI: 10.5220/0003801100450050
In *Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems* (PECCS-2012), pages 45-50
ISBN: 978-989-8565-00-6

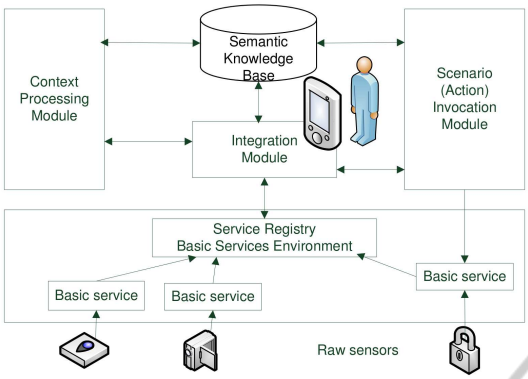

Figure 1: Main components of the Context Controller Action (CCA) architecture.

Basic Services Environment. A basic service represents a single component available in the smart space such as a sensor, an actuator, or a wrapper for an external data source. The basic services environment hosts the services.

Integration Module. The module supplies a user interface enabling the management of system elements such as basic services, contexts and actions. A user can configure which actions are taken in response to a specific context occurrence.

Context Processing Module. In our approach, a context is a set of data that describes the environment and that can be interpreted by the application (Loke, 2006). The module is responsible for runtime processing of contexts. It collects information from basic services and processes them to determine if a specific context occurred.

Scenario (Action) Invocation Module. The module executes scenarios in the environment in response to requests sent from Context Processing Module through Integration Module. We use the concept of scenario to represent a set of basic actions executed.

Semantic Knowledge Module. The module stores semantic concepts concerning basic services. The information is available for other system modules.

2.1 Basic Services Environment

Basic Services Environment hosts basic services that manage sensors, actuators and, potentially, other applications. Basic services supply a Web services interface that is used to integrate with the system. Services typically communicate with sensors and actuators (called transducers) using a device controller that manages low level operations.

Apart from hosting basic services, the Basic services module contains two components: a Service Registry and a Service Wrapper. Service registry stores information about known basic services together with their WSDL definitions, semantic description and internal information. The registry supplies the information for other system modules. Service wrapper manages concrete registrations and invocations of services, which unifies the invocations.

2.2 Integration Module

The integration module is a central part of the system and supplies two main functionalities: communication between other system modules and a convenient user interface. System modules register in Integration Module to enable data interchange and references between the modules.

The module supplies also a user interface that enables definition of contexts, scenarios and associations between them. We used the declarative oriented approach to programming to simplify the development process. Using a graphical tool, a user declares context definition that should be processed by the system. Then the user may register the definition using Integration Module that in turn passes the definition on to Context Processing Module. Additionally, a user may register scenario definitions in BPEL and specify which scenarios should be invoked on occurrence of a context.

2.3 Context and Scenario Processing Modules

Context Processing Module listens for notifications from basic services and processes them to determine if a context has occurred. On notification arrival, it verifies conditions for context occurrence. If a context is identified, the module sends an event to Integration Module that in turn searches for a corresponding scenario and sends a request for scenario execution to Scenario Invocation Module.

We utilize an enhanced Petri net (Jensen et al., 2007) to process conditions required for a context to occur. Initial places in Petri net definition represent input data from basic services or data sources, tokens represent concrete data that occurred, while transitions represent conditions that must be met for a context to occur (AND, OR). Each token has an assigned living time which enables us to identify time dependencies between them.

As an example, consider a context that represents a lecturer starting a lecture in a room. The context can identified by two conditions: the lecturer is in the room, and there is the time for the lecture in the room. We must ensure that the system processes the start of

the lecture only once. The lecturer can be identified either by an image recognition system or RFID. A notification from any of the services places a token in an initial state and fires the first transition. Lecturer's timetable is read from the Timetable service. A context is detected if transitions are fired and they reach the final state of the context definition.

The Scenario (Action) invocation module is responsible for executing scenarios (sequences of actions) in the ubiquitous environment. Scenarios are defined as workflows in the BPEL language and accept data from Context Processing Module as input. Using Integration Module, the scenario is registered in the system by giving a link to its location. Scenario invocation is triggered by a context occurrence identified by Context Processing Module. The module sends a request through the Integration Module.

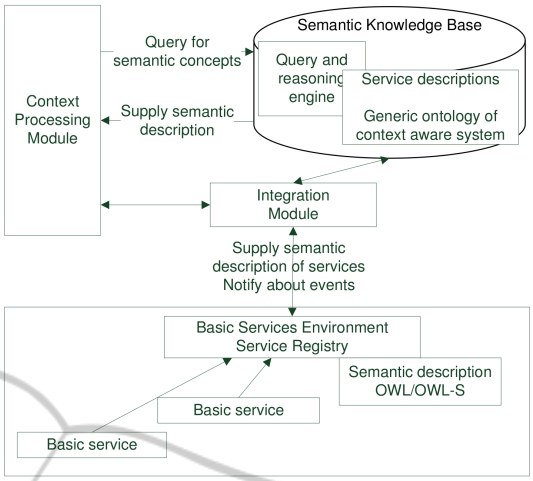

Figure 2: Semantic information structure and processing.

3 SYSTEM OPERATION

The Context-Controller-Action architecture supplies a coherent pattern for development, deployment and operation of context-aware applications. It is focused on both flexibility and a user-friendly interface. Flexibility is achieved by semantic description of concepts and dynamic registration of services, context and scenarios. A user-friendly interface enables non-expert users to define their applications and configure the system.

3.1 Component Integration using Semantic Knowledge

Semantic Knowledge Module is the central point of semantic processing as it stores the ontology that describe concepts occurring in the ubiquitous system. The concepts concern two main aspects: description of environment entities and description of services. Environment entities cover information similar to other context ontologies (such as SOUPA (Chen et al., 2004), CONON (Wang et al., 2004a)). The concepts are used to define the environment and to identify contexts. Selected concepts used in our ontology include: Person, Name, Location, TimeOfDay, GPSPosition, RoadNumber, Weather. Semantic description of services improves service registration and processing of service data. Fig 2 shows main components of the semantic infrastructure.

Basic services are described using concepts from environment and service ontologies. On registration, each service supplies its semantic description to the Semantic Knowledge Module. The description specifies a mapping between raw data of service interface and concepts from the ontology. The mapping is used in Basic Services Environment that receives data from services and translates it into semantic information. The information is then passed to context identification. Context definitions use semantic concepts from the environment ontology without concrete references to services or service data. In this solution, it is possible to exchange a service without modifying context definitions and processing.

The ontology is stored in the OWL format as the most popular format for ontology description. We used the Jena engine to process the ontology and reason about concepts.

3.2 Processing of Information

Information processing covers two main areas: system configuration and system operation. Considering the dynamic nature of a ubiquitous environment and dynamic adaptation of system operation, the areas may interchange. Main steps of making the system operational are as follows:

1. **Registration of Basic Services.** The phase requires installation of adequate devices, deploying sensor services and registering the services.
2. **Configuration of Contexts and Scenarios.** The phase covers definition of contexts using a supplied graphical tool, definition of BPEL scenarios and registering them using Integration Module. Integration Module passes contexts to Context Processing Module and scenarios to Scenario Invocation Module.
3. **Basic Services Information Processing.** The processing is performed in a loop in two alternative modes. Basic Services Environment either

explicitly invokes methods of basic services to get required data or listens for asynchronous notifications from basic services. After receiving information, Basic Services Environment translates raw sensor data to semantic concepts and sends the information to Integration Module that in turn passes it on to Context Processing Module.

4. **Context Processing Module identifies Context.** The module retrieves contextual information and maps it to context definitions using the defined ontology. If a context is identified, the module sends information to Integration Module.
5. **Scenario Invocation.** Integration Module maps an identified context with an assigned scenario and passes a scenario invocation request to Service Invocation Module. The latter invokes the BPEL scenario that may invoke services.

4 PROTOTYPICAL IMPLEMENTATION

As a part of the research, we implemented a framework in the Context Controller Action architecture and used the framework to deploy exemplary services, contexts and scenarios. The implementation covers all modules of the CCA architecture. A Web services interface is supplied for each of the modules to enable inter-module cooperation.

The majority of the system is implemented in the .NET 3.5 Platform. Knowledge Base and Scenario Invocation Modules are implemented in Java considering better support for ontology processing and BPEL processing in this language.

The .NET part contains a dedicated library for processing of the Petri net structure that processes the net considering elements specific for this system; such as token lifetime, different transition conditions and token data. We used the Jena framework for management of ontological structures and reasoning about concepts. Scenario Invocation Module uses the Apache Ode for invocation of BPEL workflows.

The system supplies both a regular user interface and an administration application interface. The regular user interface enables definition of contexts by composing block elements and associations between the elements. After definition of the models, a user may register the context and scenario, and associate contexts with scenarios.

The administration application enables monitoring and verification of system behavior such as: session establishment, registration of a new context or scenario together with its XML contents, connection and disconnection between modules. Fig. 3 shows an exemplary dialog box with monitoring data displayed.

4.1 Exemplary Services, Contexts and Scenarios

Context Processing Environment was used to implement exemplary ubiquitous applications that cover basic services, context definitions and scenario definitions. We used the following hardware devices in prototypical use of the system: BERA Electronic Door Lock, CAENRFID 949EU IP65 UHF RFiD sensor with HUBER+SUHNER AG 806-960MHz antenna, video cameras: Logitech QuickCam Pro5000, DLink SecuriCam DCS-5300G, PDA device based on Intel PXA270, a stepping motor, a multimedia projector and a PDA device. Using the devices we implemented basic services that covered different areas of functionality. Examples of basic services include: context recognition (timetable service, camera service, user authentication based on RFID, projector service), access verification (door lock manipulation, MMS notification), environment actions (PowerPoint presentation management, control of a motor, file transfer to/from PDA). Basic services were used to implement concrete contexts and invoke scenarios:

Management of Lecture Room Access. The functionality uses the RFID identification, Door lock service and Timetable service. Context definition covers place and person identification using the RFID service and lecture time verification using the Timetable service. Door is unlocked if a right person is close to the door, and the Timetable confirms that the person has a lecture in the room.

Illumination Control. The functionality uses the Camera service, Electric stepping motor service and Projector service. Context definition covers illumination calculation using the Camera service and projector state checking using the Projector service. Electric motor is activated (scenario) if illumination is higher that a specified limit and the Projector is turned on. The scenario simulates Roller-blinds closing in case the illumination is too high during a presentation.

4.2 Sensor Service Integration Process

Sensor services were implemented by independent groups of students and then deployed in the system by a single team that was familiar with system operation. The process supplied interesting data for development and runtime interoperability (Ullberg et al., 2008) analysis. All services in the system supply a Web ser-

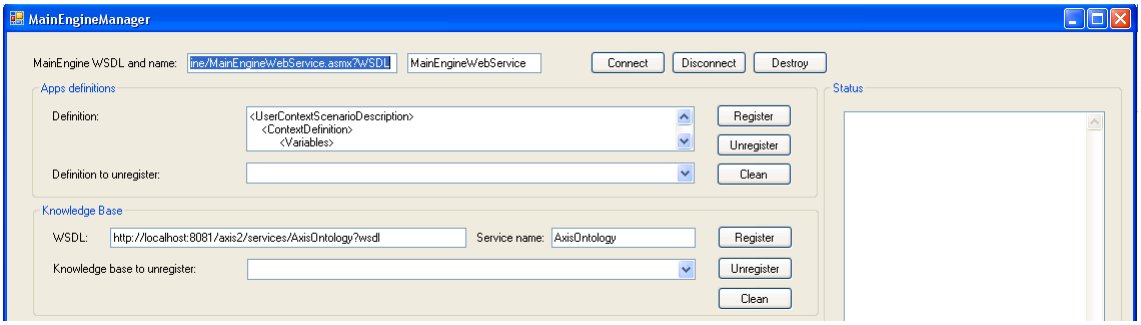

Figure 3: A dialog box that enables monitoring of internal system behavior.

vices interface for system communication, which significantly improved the interoperability. The services, however, have to communicate with the sensors using lower-layer protocols (RS232, 802.11, WEP, FTP, USB, Ethernet), which requires custom implementations (Kaczmarek and Downar, 2010).

Sensor communication was implemented in each service by an individual team, which enabled us to split the work between a large number of developers, but raised difficulties during the deployment phase. The deployment was performed by a single team that had to familiarize with the service. File manipulation, RFID and Gesture recognition services required the highest work to deploy, 24 man hours, 14 man hours and 12 man hours respectively. Other services were analyzed, deployed and integrated in less than 10 hours for each.

As a part of system testing, we performed basic performance analysis (Krawczyk and Nasiadka, 2011). Gathering of context related data has been identified as a potential performance bottleneck. Action invocation was identified as less significant as actions are invoked externally to the environment. We measured the dependence between context processing time and the number of parameters passed for processing (context size). Experiments shown that processing time depends almost linearly on context size and takes approximately 100 ms for contexts with 10 inputs up to 700 ms for contexts with 200 input.

5 RELATED WORK

Context-aware application development is an active research field addressed both by industry and academic teams. Our work differs from existing solution in the following main areas:

- We propose the Context-Controller-Action architecture that extends the standard architecture with Integration and Knowledge Base Modules.
- We enable dynamic management of environment resources and performing environment actions depending on available devices.
- We present a complete solution that uses semantic description in all system elements, which covers basic services management, context processing and action invocation.
- We supply a user-friendly interface that enables non-expert users to dynamically configure a ubiquitous environment.

Context-aware systems are usually composed of an instrumentation layer, a middleware layer and an applications layer (Meyer and Rakotonirainy, 2003) (Hsu et al., 2010). The instrumentation covers smart space sensors and physical devices that supply information to the middleware layer that identifies high level context information. The information is used by the application layer to gather contextual knowledge and deliver the expected service. The work (Rehman et al., 2007) proposes a different architecture for context aware applications that adapts the Model-View-Controller to ubiquitous environments. The work (Sui and Wang, 2006) proposes a solution for automated generation of service flow patterns, which intends to simplify user interaction with pervasive systems.

The use of ontologies and automated service discovery in context-aware applications is an active research direction. Unfortunately, there are “far too many standards” for service description and discovery (Mcgrath et al., 2007) without a consensus about interoperability formats and common semantics. The work (Chen et al., 2004) presents the SOUPA ontology that is an extensive and mature ontology of ubiquitous applications. The core of the system covers information about events, time, people, space, actions and other. Another ontology of context-aware systems is presented in (Wang et al., 2004a). Both ontologies may be extended with concepts specific for application domain. In (Wang et al., 2004b), authors designed and used an upper-level context ontol-

ogy that covers typical concepts occurring in context-aware applications.

Transformation of raw data from basic sensors to high level information is required to identify context effectively. Context awareness may be implemented either on the device level or on the infrastructure level (Loke, 2006). The first requires more intelligent devices, while the second requires more advanced context processing on the application level. A Context Widget (Salber et al., 1999) supplies abstract context information to application hiding the complexity of the actual sensors. The work (Dimakis et al., 2008) presents the SitCom and CHILix systems that enable development of context-aware applications and incorporation of smart space devices.

6 CONCLUSIONS AND FUTURE WORK

The designed and implemented solution proved that the Context-Controller-Action architecture can be used to develop a dynamic ubiquitous system. Sensor services, context and scenarios, can be configured at runtime by a non-expert user who is equipped with a graphical user interface. The internal use of semantic description enables interchange of service information and universal definition of contexts. The Web services interface was used throughout the system to integrate services with the system, and system modules between themselves.

Incorporation of semantic information in scenario definitions is an interesting area of future work. BPEL for Semantic Web Services is an important initiative to include semantics into the workflow technology, but the language has not been implemented in a processing engine or design tool yet. Deployment of further sensors and definition of contexts/scenarios is another important field of future work. The current implementation demonstrated the usability of the system, but it will be advisable to develop more advanced behavior scenarios.

ACKNOWLEDGEMENTS

This work was supported in part by the Polish Ministry of Science and Higher Education under research project N N519 172337. Students from the Faculty of ETI brought significant help in implementation of basic services.

REFERENCES

Chen, H., Perich, F., Finin, T., and Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*.

Dimakis, N., Soldatos, J., Polymenakos, L., Fleury, P., Curin, J., and Kleindienst, J. (2008). Integrated development of context-aware applications in smart spaces. *Pervasive Computing*.

Hsu, H.-J., Wu, S.-Y., and Wang, F.-J. (2010). A methodology to developing situation-aware pervasive applications with service oriented architecture. In *6th IEEE World Congress on Services*.

Jensen, K., Kristensen, L. M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*.

Kaczmarek, P. L. and Downar, M. (2010). Interoperability analysis of sensor interface in ubiquitous environments. In *31th International Conference on Information Systems Architecture and Technology*.

Krawczyk, H. and Nasiadka, S. (2011). A new model for context-aware applications analysis and design. In *UBICOMM: The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*.

Loke, S. W. (2006). Context-aware artifacts: Two development approaches. *IEEE Pervasive Computing*.

Mcgrath, R. E., Mickunas, M. D., and Campbell, R. H. (2007). Semantic discovery for ubiquitous computing. *CiteSeerX - Scientific Literature Digital Library and Search Engine*.

Meyer, S. and Rakotonirainy, A. (2003). A survey of research on context-aware homes. In *Australasian Information Security Workshop Conference on ACSW*.

Poslad, S. (2009). *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley Publishing.

Rehman, K., Stajano, F., and Coulouris, G. (2007). An architecture for interactive context-aware applications. *IEEE Pervasive Computing*.

Salber, D., Dey, A. K., and Abowd, G. D. (1999). The context toolkit: Aiding the development of context-enabled applications. In *CHI, Pittsburgh, PA*.

Sui, Q. and Wang, H.-y. (2006). A service oriented flow pattern in pervasive computing environments. In *1st International Symposium on Pervasive Computing and Applications*.

Ullberg, J., Lagerström, R., and Johnson, P. (2008). A framework for service interoperability analysis using enterprise architecture models. In *IEEE SCC (2)*, pages 99–107.

Wang, C., Zhang, D., Hung, T., and Pung, K. (2004a). Ontology based context modeling and reasoning using owl. In *Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*.

Wang, X., Dong, J. S., Chin, C., Hettiarachchi, S. R., and Zhang, D. (2004b). Semantic space: An infrastructure for smart spaces. *IEEE Pervasive Computing*.