A Multi Agent, Service Reassembling Architecture for Context-Aware Systems

A. V. Neelima¹, B. R. Sunitha², and C. G. Aghila²

¹Regency Institute of Technology/Department of Computer Science, ynam, INDIA
Email: neelima_valluru@rediffmail.com
²Pondicherry Central University/Department of Computer science, Puducherry, INDIA
Email: [maanila, aghila]@yahoo.com

Abstract—The system which acts according to user’s current situation for providing the necessary services in that context is known as a context aware system. These systems as such don’t require any explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account. Context is the information which defines an entity. An entity can be a person, a place, or a thing etc. Context is not about an entity but about the surroundings of an entity which provides the base for defining the entity. Such a Context has multiple dimensions like, i.e., time, location, identity, activity, environment, networks, devices, social situations, etc. The study of literature reveals that context aware system architectures has made use of limited context for providing context aware services to the user which are limited to a particular application domain. But it can be viewed that a architecture can be designed to cross cut multiple applications by viewing a combination of contextual elements there by providing application specific user services. Appropriate services pertaining to user from a set of services provided by application is identified and offered by way of assembling the atomic service patterns by taking the user context in to account.

Index Terms—Context Aware Systems, Multi Agent systems, Service orientation, Service Reassembling.

I. INTRODUCTION

The history of context-aware systems started in 1992, whenWant, Hopper et al introduced Active Badge Location System [16], which is considered to be one of the first context-aware applications. This system is based on infrared technology is able to determine a user’s current location which was used to forward phone calls to a telephone close to the user. In the middle of the 1990s a couple of location-aware tour guides [9,15] emerged which provided information according to the user’s current location. While location is the most used attribute of context, an attempt to use the other context information has grown over the last few years. Based on the type of context data used in the applications, the context aware systems can be classified in to four main categories, Location based systems, Identity based systems, time based systems and Activity based systems.

Location based systems considers mainly location information in order to provide the services to the user. Some of the most popular examples of location based systems include Office and Meeting tools [12,18], Guides [9,15], Location tracking systems [2, 23], Conference assisting systems [17, 20] etc.

Activity based systems, gathers activity information of the user and stores it for later use. Generally this activity information will be used to stamp the context data. The activity might be reading, speaking, watching, running, sleeping etc. Some of the example activity based systems are Pepsys [23], Context Aware Activity Notification System CANS [1] etc.

Time Based systems consider time as the context element for providing context aware services. Time based systems mainly act as reminder systems. These systems provide services considering time as main contextual element, in combination with other contextual information like location, profile etc. Some of the examples include Personal Handy Phone System [11], Minimal Attention User Interface (MAUI) [12] etc.

Identity based systems provide the services based on the profile of the user. Profile is a powerful tool for providing personal services to the user. Systems like personal reminder system, tour guides, travel agents etc consider the users personal information like his financial conditions, current location and provides the services. Handheld Event Planner [6], Music recommendation system [8] etc are some of the examples of identity based systems.

A. Approaches for Building Context Aware Systems

The approach for developing the context aware systems must be flexible so as to incorporate dynamic features so that the system can be able to cater to the needs of the context aware application. Some of the approaches for building context aware systems from the literature are

- Server Based Approach
- Agent Based Approach
- Service Oriented Approach

Server based approach will have a centralized component as common element in their design, which is responsible for context acquisition, context management, context reasoning and maintaining privacy. For example a Context Managing Framework (CMF) [10] follows server based approach. It is a stand-alone software framework designed for mobile terminals. Yet another example that follows server based approach is Context
Agent based approach consists of agents encapsulating specific functionality and offering services to exchange information with other agents. Multi agent systems enhance the modularity, flexibility, scalability and ease of use to the system. They make the system reactive and proactive in nature. CoBra [7] (A Context Broker Architecture) provides a sample intelligent meeting scenario represented in OWL schema. OCASS[19](Ontology based Context Aware Multi Agent Service System). OCASS is intended to provide services in a smart home environment.

Most of context-aware systems use service-oriented approach as this approach provides a loose coupling nature to the system. That is this approach provides on-the-fly services to the user which is very much needed in today’s mobile world. Web services are an alternative for providing such dynamic services. Service oriented nature of the system also enhances the scalability, modifiability, modularity and remote accessibility of the system. Also much of the Context information with the exception of information provided by GPS cannot be directly used in context aware applications. For this reason, most approaches of context-aware systems have service-oriented architecture based on Web services with context information such as user’s location, profile, and activities. They evaluate user environment and push information that is relevant to user context. Some of the examples of context aware systems that follow a service oriented approach are Capilla[13],Omni-present[22].

B. Analysis of Survey work

The study of the above systems and approaches reveals the fact that the context aware system architectures are designed keeping a particular application in view. So the present day context aware systems are in such a way that they are not able to span multiple applications, since the underlying architecture doesn’t allow considering other contextual elements which are not addressed in it. As a result the user needs to have separate systems for different context aware applications. Hence there is need for a generic architecture that which can span multiple applications and provide user specific services pertaining to the applications.

That is the architecture should incorporate a mechanism for providing the user specific services relevant to application, by considering all the primary dimensions of the context. Also the architecture should reflect the dynamic change of services from application to application and from user to user. Therefore reassembling of the services is needed to be addressed in the architecture in order to provide filtered out services pertaining to user context from a set of available services of the application i.e., the system should automatically learn from the perceived environment, should be able to reason with the information and should make inference from the information. Then and only then it will be possible to provide appropriate service suited to the user context. Also since the user is aiming at acquiring services any where at any time, it is necessary that the system should incorporate loose coupling nature.

The basic idea is to propose architecture that “considers the primary dimensions of context, selects the appropriate application to be fired, based on the acquired location context data and provides the user specific services relevant to the application by reassembling the service patterns based on the secondary context data.”

The architecture, either by pulling or pushing has to acquire the context data, filter in to appropriate profiles and then uses location information for identifying application and provides user specific services pertaining to the specific application by way of re-assembling the atomic service patterns depending on a mixture of secondary contextual dimensions, there by cross cutting multiple applications. Thus the system can be made a truly context aware system.

II. FEATURES OF THE PROPOSED ARCHITECTURE

From the previous discussion the following features can be identified as the credible facts about the architecture

- The Architecture is generic in nature — The genericity is achieved by way of implementing it so as to handle multiple applications.
- The Architecture is dynamic in nature — The architecture automatically cope up to a new application depending on location context and provides application specific services.
- Provides Application specific and user centric services — This is achieved by reassembling the service patterns depending on secondary context data like profile of user, time and activity of the user. Features of The Proposed Architecture

III. COMPONENTS OF PROPOSED ARCHITECTURE

The proposed architecture as shown in Fig.1 constitutes five major components namely, a Context managing component, a Service identifying component, a Sensor zone, a Repository and Intelligent artifacts.
A. Context Managing Agent

The schematic of context managing agent is as shown in fig 2.

![Context Managing Agent Diagram]

Fig. 2. Work Flow of Context manager

The main task of context managing agent is to acquire the context and perform reasoning on the context data using ontologies. Context managing agent accesses the ontological data and historical context data from the database. Reasoning is performed in order to correctly identify the context so that it might be possible to fire the appropriate application. Once the reasoning process is completed, the context managing agent stores the context data in the database in the form of tables for further use.

Context managing agent acquires the data from the sensors and filters the data into appropriate profiles. Filtering of context data is needed to distinguish between various context dimensions. A profile is like a file corresponding context dimension. Context dimensions are represented by using ontological representation.

The context managing agent uses rule based reasoning approach for reasoning with the context data. Reasoning is performed in order to identify appropriate location. Location is used as the primary context information for identifying the appropriate application. The combination of secondary context information is used to identify the relevant services. The work flow of context managing agent is shown in fig 2.

B. Service Identifying Agent

Service identifying agent is responsible for identifying appropriate application and firing out relevant services. Application patterns are used for identifying appropriate application. A working pattern of service identifying agent is shown in fig 3.

![Service Identifying Agent Diagram]

Fig 3. Work Flow of Service Identifier

C. Sensor Zone

The input information to a context aware system is normally provided by sensors. Sensors will be polled by an intelligent artifact, since they cannot advertise themselves. In this proposed architecture the sensors are simulated in order to obtain the context information. The simulation produces random values of contextual elements. This data internally is represented by using ontologies.

D. Intelligent Artifacts

These intelligent entities, acting as Clients, must be able to poll Sensors and Actuators, in order to use their services. In the proposed architecture currently a desktop system takes the place of the intelligent artifacts.

E. Repositories

The repository consists of data like context history data, ontology data, service patterns, application patterns, service repository. Context manager accesses the repository for getting the historical context data and ontological data which can be used for reasoning and inferencing purposes. The manager also stores the current context data in the repository to be used for future references. Service manager accesses the repository for identifying the application patterns, service patterns therefore providing a right set of services to the user.
F. Ontologies

Ontologies in this architecture are used to specify the concepts of context dimensions. Each context dimension is described by means of ontology. The description of the context dimension is done by means of contextual element attributes.

G. Historical Context Data

This repository is meant for maintaining historical context data of the user. History is maintained by tagging the situation with appropriate context information. This data will be very much useful during reasoning process. A new context data can be inferred by combining the current context data with the historical data.

H. Application Patterns

An application pattern is a schema representing the application. These patterns are needed for identifying appropriate applications. They are static in nature i.e. for each application a schema will be prepared and stored as table structures in the database. An application pattern constitutes Name of the Application, Identity of the application, and Context Preconditions for application to trigger, Resulting Context- post conditions, set of tasks to be performed.

I. Service Patterns

A service pattern is a schema that provides information about the service. These service patterns are static in nature i.e., a schema of the service will be prepared and stored as table structures in the database. The attributes of service pattern constitutes Service identifier, Service Description, Input needed to execute the service, Output of service execution, Set of subtasks. Each atomic service is represented by a service pattern. These atomic services are assembled dynamically in order to provide service to the user.

J. Filtered Context Data

This data base is mainly intended for storing the filtered context data in to appropriate profiles. This gives space for a clear definition of contextual dimensions in separate spaces.

IV. IMPLEMENTATION OF PROTOTYPE

The generic nature of the architecture has been tested by implementing and testing with three different applications. The contextual information is obtained by generating random values periodically for various contextual elements. Agent oriented and service based approach is used for realizing the architecture. Java programming language is used for implementing the proposed system. JADE framework is used for simulating the agents [3, 4]. Depending on the random value generated for the location context, the application pertaining to the location information is automatically popped up with a set of services specific to the user.

For example in the implemented prototype the location id of the library is given in the range of 1 to 30. So when a random value between 1 and 30 is generated the system automatically pops up the user interface pertaining to the library application. Usually the set of services provided in a library constitutes intimation of new arrivals of books in user’s interested area, fine payment details and location based search for books and personnel in the library. But the system while providing the services should highlight only those services which are apt to user’s current context. Hence the set of services to be provided differs from user to user for the same application. The sample screen shots of the one of the implemented application are shown in fig 4 and fig 5.

Fig. 4. Screenshot showing the Service Identifying agent For Library Application

Fig. 5. Screen shot showing the Services pertaining to a particular user in Library application

For example in the library application, Fine payment service, every user may not have fine to be paid at all times when he visits the library. In such case highlighting the fine payment service to every user is ridiculous. Similarly is the case with another service, new books arrival in user’s interested area. Hence the proposed architecture considers each user’s profile, previous time when he visited the library, current time of his visit and provides only the services which are apt to his current profile and time. This is where the reassembling of services is achieved in the architecture.

The Algorithm for implementing the reassembling concept by service identifying agent is explained as follows.
V. EVALUATION OF PROPOSED ARCHITECTURE

The architecture is evaluated against a set of predefined quality attributes that are implemented in the architecture. A set of ten predefined design criteria for an ideal architecture have been identified from [21] and the proposed architecture is evaluated for these ten criteria. The degree to which the proposed architecture meets predefined goodness criteria has been calculated based upon the Design Selection Analysis model as proposed by Asada in [14]. It has been found that the architecture has achieved a goodness value of 78% i.e., the architecture has achieved more than three fourth efficiency of the ideal architecture which implies that the architecture is approaching the ideal architecture.

VI. CONCLUSION

The aim of this work is to develop a generic context aware architecture that suits multiple location aware applications and that also considers other secondary contextual elements so as to provide user specific services. The developed architecture is novel by the way it implements the reassembling of service patterns based on secondary contextual information to provide user specific services pertaining to specific application thereby crosscutting many application domains. The architecture has been simulated to test with three applications. Evaluation of architecture has been performed and the results show a higher goodness value for the same.

REFERENCES


© 2009 ACEEE
DOI: 01.IJRTET.2009.01.01.366

565

ACEEE