An integrated development environment based on ontologies for building context-aware applications

Taciano P. A. Alcântara¹, Mauro Oliveira¹, Luiz O. M. Andrade², Anilton S. Garcia³, César O. Filho¹, Leonardo M. Gardini¹

¹ State University of Ceará (UECE) - Fortaleza, Brazil
² Federal University of Ceará (UFCE) - Fortaleza, Brazil
³ Federal University of Espírito Santo (UFES) - Vitoria, Brazil

taciano@ufc.br, amauroboliveira@gmail.com, odorico@saude.gov.br, anilton@inf.ufes.br, cesarolavo@ifce.edu.br, lgardini@gmail.com

Abstract—Context-aware applications can help information systems users on decision making. However, the development of these applications involves a level of complexity that often complicates the process of creating context-aware systems. This paper proposes an Integrated Development Environment (IDE) to assist in developing context-aware applications and utilizes a promising modeling method based on contextual ontologies. This work presents the architecture, specification and prototyping of the proposed IDE as well as the results obtained in Lariisa, a platform for governance decision-making in health systems.

Keywords—IDE; context-aware; health; system; decision-making

I. INTRODUCTION

Context-aware Systems [8][16] can bring positive results to support management and decision-making. This type of system processes contextual information, in other words, information that characterizes a given situation where a user is and supports the offering of intelligent services.

However, the development of context-aware applications is highly complex because these applications involve the use of knowledge representation concepts, inferences, among others [1]. Moreover, they use data sources (context providers) from various types, such as mobile devices, set-top boxes, sensors, etc., decreasing considerably the developer productivity and increasing development costs.

This paper proposes an Integrated Development Environment (IDE) based on modeling knowledge using ontologies - a powerful contextual representation tool. The proposed environment has several features, such as maintaining knowledge databases, setting rules and actions, information management, integration with context providers, etc. These features are unified into a single environment to support the development of context-aware applications.

As a result, this paper proposes an IDE applied to the Lariisa project [13], a governance decision-making framework for public health care systems. An example is the development of a simple application for monitoring dengue cases [3].

This paper is organized as follows: Section II describes the concepts of context-aware systems and ontologies. Section III presents related work. Section IV shows the architecture, specification and prototyping of the proposed IDE. Finally, Section V concludes the paper and discusses future work.

II. CONTEXT AND ONTOLOGIES

A. Context and Context-aware Computing

With the popularization of mobile devices, such as smartphones and tablets, users may perform tasks on these devices while moving. In this scenario, information could be captured revealing where the user is or what the user is doing, and then this information could be used to offer personalized services and information. This type of information, which characterizes a situation and can be used by decision-making applications, is called a context. Applications that use this type of information are named context-aware applications [2].

Aiming in assisting users in their day-to-day tasks, context-aware applications have been using elements of ubiquitous systems to obtain users context information. A simple example is the use of sensors that detect the presence of people and automatically trigger lighting to an environment, according to the people location and time.

Three important aspects of context are defined as follow [16]: information related to where a user is, with who a user is, and what resources are close to that user. These aspects may change continuously, and a huge quantity of information could be derived from them, such as: luminosity, noise level, network connectivity, communication costs, bandwidth, social status, etc.

A context, to be represented, needs to be modeled by some technique. A context model defines types, names, properties and attributes of the entities involved in context-
aware applications, such as users, and other mobile devices. The model attempts to predict representation, search, exchange and interoperability of context information among applications. A well designed model is the key to any context-aware system [6].

B. Ontologies

The amount of information present on the web is increasing every day. More and more people are connecting on the world wide web, creating and sharing data. However, traditional information recovery does not reflect the data semantic, their relationships and the knowledge that they represent. To have a sustainable growth, it is needed to adequately manage this huge mass of information. Semantic Web helps computing devices to understand the meaning of information stored/transmitted over the Internet [4].

Building a semantic web application needs the creation and implementation of technology standards to establish semantic concepts that make possible sharing information between two or more systems. It is necessary to create mechanisms that describe data and represent the encoding of shared meanings. One of these mechanisms is defined using ontologies.

Information science studies the phenomenas related to the information in its many aspects - trying to understand and follow its outspread. By doing that, in the future, that information can be available on information systems. Understanding a subject point of view. Using ontologies, especially in the Computer Science area, makes possible the communication between different people and computer systems that participate in the same knowledge field - but not necessarily share the same form of conception about the elements of this domain.

An important reason to use ontologies is the guarantee of reliability surrounding vocabulary concepts or languages that are used in certain environments. Thus, using formal representation acquired with this application, it becomes possible automation of consistency verification, generating environments more reliable.

III. RELATED WORKS

A Context-Aware Web Content Generator Based on Personal Tracking [4] is a context-aware system for the creation, annotation and sharing of multimedia content.

There are several applications that use context information to enrich and organize multimedia documents. This information might be proximity of people or objects in the photo, current temperature, date, etc. This type of metadata can be obtained from sensors of mobile devices or from the web. With this information associated, context-aware applications can better organize the multimedia, providing user-friendly visualization of the content, and suggesting annotations for document indexation[9][10][11].

The ubiquitous computing paradigm addresses mobile networks and distributed systems and demand for fast and reliable systems that are accessible anywhere and at anytime [17]. Ubiquitous systems must be able to execute in at least two environments: (i) volatile environments in which mobile devices suddenly appear and disappear; and (ii) dynamic environments in which the network topology, their components and services are constantly changing [18].

These infrastructures provide mechanisms to collect, process and share context information used to infer user’s situation. In our research, we realized that a common solution for ubiquitous systems coordination is based on shared memory space and event-based models [19].

IV. BUILDING CONTEXT-AWARE APPLICATIONS BASED ON ONTOLOGIES

This Section proposes an Integrated Development Environment (IDE) to help developers of context-aware applications modeled with ontologies. The following are the architecture and IDE modules.

A. Architecture

The proposed IDE organizes the modules into an architecture that supports the integration of context-aware applications (Figure 1). The IDE has 5 main modules: Knowledge Management (module to manipulate knowledge database), Context Providers Management (module to manipulate context providers), Rules and Actions Management (module to manipulate rules and actions), Information Management (module to manage the information) and Notifications Management (module to create executable and simulation).

![IDE Architecture](Image 316x202 to 550x394)

Fig. 1. IDE Architecture

Figure 2 shows the activity diagrams of the IDE. It is possible to observe, on this figure, all action flows that allow the developer to create his/her own context-aware application - from the definition of knowledge database,
going through rules, actions, context providers and finally simulation of executable.

**Fig. 2. Activity Diagram of the proposed IDE**

**B. Prototype Description**

Figure 3 shows the prototype’s home screen of the proposed IDE. It contains shortcuts to access the main features (modules) of the IDE.

**Fig. 3. Home Screen Prototype**

1) **Knowledge Database**

Knowledge databases are used to model context representation. A well defined model is important to have accuracy in representing the reality.

As shown in Figure 1, the Knowledge Management module makes use of tools to represent knowledge database. These tools allow ontologies reading and they are able to perform inferences. The knowledge represented on the databases is imported to make possible data storage following domain semantics.

On this module it is possible to find ontologies through search tools integrated to the environment - only informing a key-word. The IDE has integration with a variety of ontologies databases stored on the Internet. The developer has an option to choose one or more ontologies available on the search list, and then import these ontologies to his/her own application.

The search tool specified integrates with ontology databases on the web and then it provides source code for reuse. IDE has some integrated databases and it provides a feature where the developer can add or remove new databases.

The developer may use a variety of ontologies in his/her application and may also edit or create new ontologies. It is provided a simple OWL (Ontology Web Language) editor to make possible perform editions without leaving the tool. If the developer desires an editor more complex, he/she can use third party softwares, e.g.: a graphic editor of ontologies [15], and then export the ontologies to subsequently save them on the IDE editor.

2) **Context Providers**

Context providers are agents able to collect contextual information and send it to a context-aware system or a context server. Context providers make use of raw data that will be analysed to represent contextual information.

A specific context provider is all devices able to provide any contextual information. As an example, smartphones or cell phones with GPS are able to provide geolocation and the user’s phone number.

The IDE provides an interface to the developer manipulate context providers, including features like: add context providers; add directory context providers; select context providers information; select directory information from context providers; remove context providers; and remove context providers directory.

It is also possible to add a group of context providers by just typing the address of the providers directory. This directory has a number “N” of similar devices able to provide the same type of information. For example: temperature sensors installed in all neighborhoods of a city. Thus, it is possible to know the temperature in each area of a city, just using a directory that represents all of these sensors - because all of them provide the same type of information: temperature.

3) **Rules and Actions**

The Rule and Action Management module checks, during each context change, whether a rule was assigned or not. In positive case, it calls the action management. On this module it is defined how will be the rule checking and its relationship with Actions Management.
A Rule represents a specific feature that the system infers when a contextual information configuration is reached. An example of rule related to the health area is the diagnosis of dengue [3]: the system detects a possible infection of a patient if he/she has at least three symptoms of dengue and lives in an area where there is a focus of dengue. Preconditions are assumptions to satisfy any rule, and, when a rule is satisfied, it results in an action.

Action is one of the main features of a context-aware system – it is where a system reacts (though actions) to changes in the context, without having necessarily a human intervention. Action Management is called by the Rules Management (both in the same module) where some rule is reached and step in to notify a context-aware application about changes in the context. This module defines the relationship with Actions Management and how works the notification to the application.

4) Contextual Data (Information)

This module uses specialized techniques to run queries in ontologies and inferences. This module also defines how contextual data will be provided originated from context providers is reflected on the system.

After selecting context providers, these context providers are able to send contextual information. The storage and recovery of contextual information is the responsibility of the Information Management Module.

Features from the user point of view: consult data stored on the contextual database; change data stored on the contextual database; and remove data stored on the contextual database.

Services offered by this module require that a user (e.g. the developer) understand the data query language or that the integrated systems implement a query API.

5) Executable and Simulation

After configuring knowledge databases, rules, actions, context providers and how to access the information, the developer can generate/create an executable of the project. This executable is linked to a context-aware application and it is responsible to collect contextual information, process it, and add notifications to the application that is assigned to receive them (the notifications).

The developer can also use the executable module to simulate an operation within the IDE. The environment still offers mechanisms to emulate a context-aware environment with fictitious context providers and context-aware applications.

The executable created by this module is the product created by the IDE. Using this executable an application can work with context features, interacting directly with the executable to establish communication with the sensor, knowledge database, notifications, etc.

V. PROTOTYPING IMPLEMENTATION ASPECTS

A. Knowledge Database Management Module

The proposed IDE uses ontologies on languages such as OWL, RDF and RDFS to manipulate knowledge databases. Jena [12], a Java framework designed for semantic web, used to represent knowledge and handle ontologies through defined classes, entities, properties, etc, supports the creation of inference rules.

On this module the developer may operate as follows: Search ontologies; Import ontologies; Edit ontologies; Remove ontologies; Create ontologies.

Fig. 4 – Implementation aspects of the Knowledge Database Management Module

The developer is able to import ontologies from the web to the proposed IDE. The developer can also edit ontologies directly on the platform. All manipulation of ontologies via graphic interface will be reflected on the knowledge database through the Jena Ontology API.

B. Context Provider Management Module

A strategy was developed [1] to reflect contextual data on knowledge databases. A consulting adapter (query adapter) was designed to translate information from WebServices to perform queries based on HTTP protocol. It was needed to standardize the way that information on the knowledge databases, deriving from context providers (LISA) or other services that perform queries [7].

On this module the developer can operate as follows: Add a provider; Add directory of providers; Select information from providers; Remove providers; Remove directory of providers.

Fig. 5. Implementation aspects of the Context Providers Management Module

The Joseki (an opensource tool that runs over a HTTP server, supports GET and POST operations and translates queries requests to the SPARQL language – used to perform queries on knowledge databases) was used to perform queries in the knowledge database. An overview of the Context Providers Management Module implementation can be found on Figure 5.
C. Information Management Module

This module supports insertion of data coming from contexto providers and queries that the application perform on the knowledge databases. Applications (or external services provided by applications) may manipulate knowledge database information directly using the HTTP protocol – without the needed for using query adapter.

The Information Management Module also uses the Joseki to perform queries in the knowledge database (as well as the implementation of the Context Providers Management Module).

D. Rules Management Module

For handling rules, the developer uses a graphic interface of the proposed IDE. Internally the rules are represented by one of Jena’s framework module: general purpose rule engine. It makes possible to perform inferences on knowledge database or detect changes of context to set the actions.

On this module the developer can operate as follows: Create rules; Define preconditions that satisfy rules; Edit rules; Remove rules.

Rules are stored in files following the syntax defined by Jena and the rules are handled through an API.

E. Actions Management Module

The proposed IDE uses the pattern Observer to manipulate actions. This pattern allows that interested systems are advised about any change on the condition of any rule defined by the developer.

On this module the developer can operate as follows: Define which rules will trigger actions; Register applications (listeners) to receive actions.

![Diagram](Image)

Fig. 6. Implementation aspects of the Actions Management Module

Figure 6 shows an example of situation where three context-aware systems (listeners) are observing the proposed IDE, and that they are ready to receive notifications. The IDE provides an API, regardless of programming languages, allowing interested systems to implement the Observer pattern following the context-aware notification structure.

VI. CONCLUSION

Integration of the proposed architecture modules (Knowledge Database, Context Providers, Actions & Rules, Information Management and Creating Executable & Simulating), creates a complete environment that provides to the users the main tools for building context-aware applications. Considering all of these features, the proposed IDE has a great potential to help the development of context-aware applications.

The main focus of this paper was to specify and describe the prototype implemented as a Proof of Concept of the proposed IDE. The prototype has been used on the Larissa project [13] in two ways: The first relates to aggregating georeferencing on the concept of context-awareness into the IDE [4]. The second relates to migrating a software platform that will allow the publication and integration of open data related to public health in a cloud environment [14].

REFERENCES


