WILDCAST CONTEXT-AWARE FRAMEWORK: CASE STUDY FOR PLACE SELECTION

Goran Bakraceski
Faculty of Computer Science and Engineering
Skopje, Macedonia

Vladimir Trajkovik
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

Context aware computing is a mobile computing paradigm in which applications can discover and take advantage of contextual information. It offers new opportunities for application developers to develop and adopt system behaviour based on any contextual information. In this paper we will present core functionality and characteristics of one of the many context management frameworks, the WildCAT framework and its usage. We will present a demo application that use contextual information, which is stored in WildCAT and used for decision making.

Keyword: context, context-aware framework, WildCAT, application

I. INTRODUCTION

With the appearance of the new generation of mobile devices, smart phones and notebooks with their huge potential of mobile processing and services, the ubiquitous computing become more popular. Including the context in mobile computing, making them able to adapt their behaviour to the current context state makes their power even greater. But what actually word context really means? The most accurate definition about what the word "context" really means seems to be the following: "*any information that can be used to characterize the situation of the entities (person, place or object) that are considered relevant to the interaction between the user and the application, including the application and the user itself*" [1].

There are many approaches for modelling contextual information but the following seems to be commonly used for creating and storing contextual information:

- Key - value model [2]
- Markup scheme models [3], [4], [5], [6]
- Graphical models [7], [8]
- Object oriented models [9]
- Ontology based models [10], [11]
- Tree - like graph [12]

Of all contexts modelling approaches mentioned above, in our demo application that will be presented in this paper we will use tree-like graph for context modelling and decision making.

In order to simplify the process of developing context aware application an abstraction framework is needed. There are several context frameworks available: SOCAM (Service-Oriented Context-Aware Middleware) [13], CASS (Context-Awareness Sub-Structure) [14], Hydrogen [15], WildCAT [16] etc. The framework on which this demo application is based is the WildCAT framework. We have decided to use this framework because this is the only open source framework that can be found.

In this paper, we first give a quick overview of WildCAT framework (Section II) and its fundamentals. In Section III we will describe which contextual information will be used and mapped with the framework and in Section IV we are about to show a real implementation and usage of WildCAT framework in our *"Place decision maker"* demo application. In section V we will show the decision making process and will cover different use-case and conclude in section VI.

II. WILDCAT FRAMEWORK OVERVIEW

The WildCAT framework is extensible Java framework that eases the creation of context-aware applications. Provides a simple and powerful dynamic model to represent an application's execution context and offer a simple API for the programmers to access this information both synchronously and asynchronously (*pull* and *push* modes).

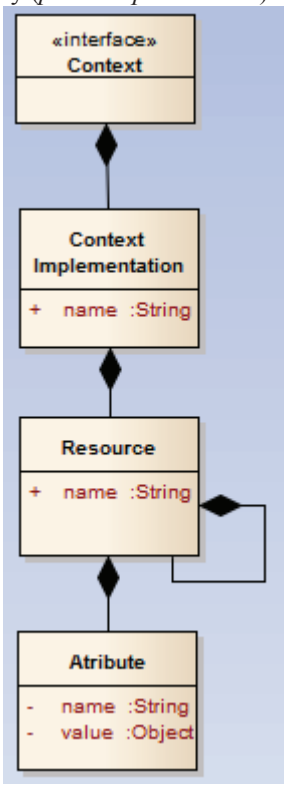

Figure 1: WildCAT logical data model.

Figure 1 represents the logical data model used in WildCAT. The Context class serves as façade for the whole system from the clients' point of view and it is a singleton class [17] which means only one instance exists in each application, representing the whole context.

The Context is organizes as an oriented tree structure with two types of nodes:

- *Attributes nodes* that hold some value. Attributes nodes are always leaf nodes, and every attribute has

a unique parent node. There are three types of attributes: *(i)* Basic attributes (holds static values), *(ii)* Active attributes (sensors) and *(iii)* Synthetic attributes (the result of expressions on other attributes)

- *Resources nodes* that can have zero or more children assuming every child's name is unique. A resource may have more than one parent. There are two types of resources: *(i)* Basic resource and *(ii)* Symbolic link (resources that alter path resolution by pointing to another resource).

The main features of WildCAT are: *(i)* its simplicity from the end-user (i.e. application programmer) point of view, with a familiar hierarchical data model and a small and easy to use API, and *(ii)* its extensibility due to its framework approach which supports different levels of customization.

III. Using WildCAT in "Place Decision Maker"

In order to show the real usage of this WildCAT context management framework, we will develop a demo application (the real application should be mobile application that will acquire contextual information from smart phone sensors and services) that will use different types of context information to make different reasoning operations. For this purpose we have defined four times of different context information:

- *Location* (city)
- *Season* (Winter, Spring, Summer, Autumn)
- *Time of day* (Morning, Noon, Evening)
- *Weather type* (Sunny, Cloudy, Rainy etc.)

The user will be asked to enter all of the above mentioned context information and according to user's input the application will be able to determine the best place for suggestion. This contextual information is stored as tree-like decision graph model (fig. 2) so we have no difficulties to implement decision making module.

WildCAT framework use a syntax inspired by URIs to denote elements in the context while being independent on the actual implementation. A path that is widely used in WildCAT can be used to reach any resource, attribute or all the sub-resources or sub-attributes of a resource. Every path can be syntactically valid but such a place may not exist. The general syntax is root://path/to/resource#attribute. Different examples are show in table 1.

Table 1: Examples how context information is stored with WildCAT

Resource path	Comment
self://Ohrid	A resource
self://Ohrid/*	All resources for Ohrid resource
self://Ohrid/Summer/Noon /Sunny#decision1	An attribute
self://Ohrid/Summer/Noon /Sunny#*	All attributes for given condition

As it is displayed on Figure 2, the contextual information is organized as decision tree-like graph with six levels. The first level is reserved for the root of the tree. Under the root are placed all contextual information that are acquired from the user's input like location, season, time of day and weather type. The last level is reserved for place suggestions that are offered to the user to visit.

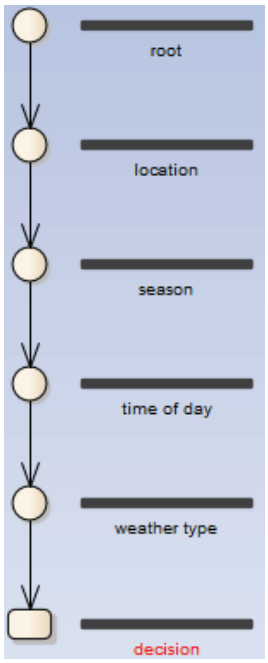

Figure 2: Decision tree context organization.

IV. Place Decision Maker (Demo Application)

The demo application that we have developed called "Place decision maker" is Java Swing application that uses WildCAT as a context information store and processing framework.

When the application is started for the first time, all context information is loaded using "DecisionRepository" class.

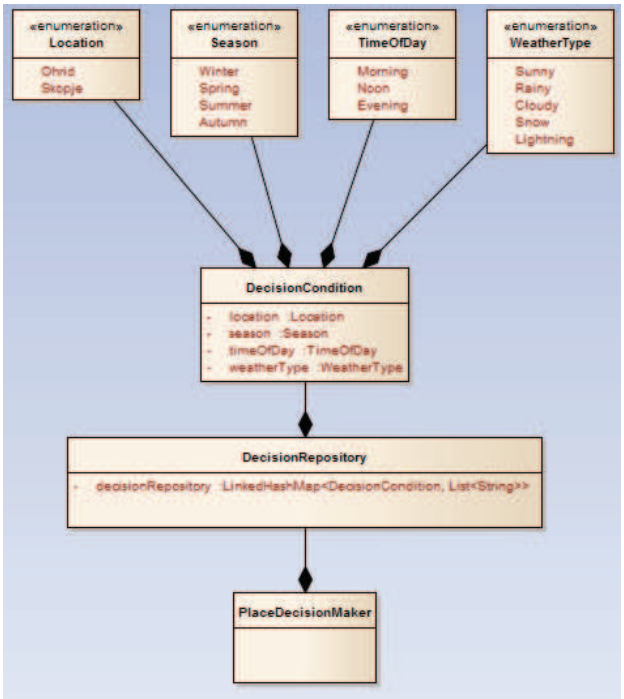

Figure 3: Place decision maker class diagram

On figure 3 is shown class diagram of "Place decision maker" demo application. Each of the contextual information (*location, season, timeOfDay, and weatherType*) is represented as an enumeration in our demo application and is used to create decision conditions which are used in "Decision repository" afterward. Decision repository keeps track of all registered conditions in the application and the decisions that correspond to them.

All decision condition and its decisions are mapped into tree graph using WildCAT framework. Using the WildCAT framework we can easily investigate the graph build by this framework and make suitable decisions (figure 4).

V. Decision making process and use-cases

The decision making process is provided on tree-like graph build by WildCAT framework (figure 4). As we can see on figure 4, the graph has few levels in depth and each represents different context information.

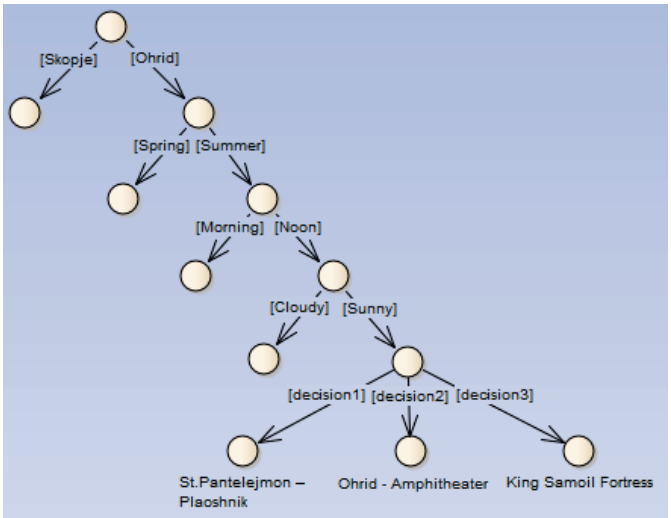

Figure 4: Part of context information build with WildCAT

The first level is reserved for the one and only root node. Every contaxt that is build with WildCAT should start with root node. On the first level we keep information about location context, on the second level we keep information about season context. The third level is reserved for contextual information about which time of the day when the search is indicated. On the forth level we keep information about the weather conditions where user is located. Finaly, on the fifth level we have multiple dicesions that can be seleced if all of the previously mentioned contextual information are satisfied.

The user interface of "Place decision maker" application (figure 5) is consisted of four combo boxes (one for each mandatory context that the user should enter), a button that perform reasoning process and a label where found results are displayed. The user can change any of the contextual information (figure 5) and search for suitable places for the provided conditions.

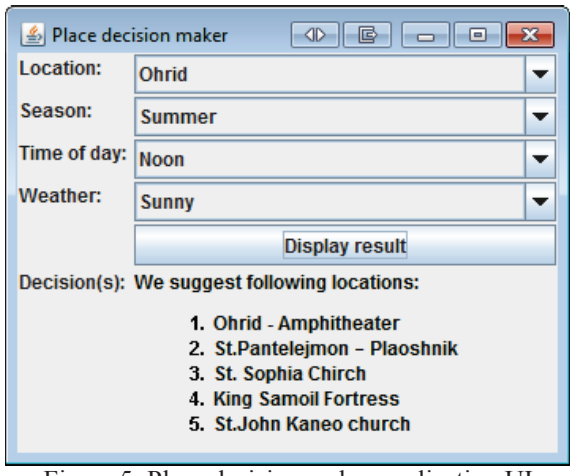

Figure 5: Place decision maker application UI

For example, the user can select following search conditions:

- Location: Ohrid
- Season: Summer
- Time of day: Noon
- Weather condition: Sunny

The program will inspect the contextual graph using WildCAT push mechanism and try to find suitable places for provided condition. As it is shown on figure 5, the result will be following five places: *"St.Pantelejmon – Plaoshnik", "Ohrid - Amphitheater", "King Samoil Fortress", "St. Sophia Chirch" and "St.John Kaneo church"*.

VI. Conclusion and future work

In this paper we presented a demo application that use context-aware framework to easy create and manipulate with context information. We have explained how to use WildCAT as context-aware framework for storing and managing context information and shown a real usage of graph model for keeping context. We will extend this demo application with mobile version that will automatically acquire contextual information from mobile device sensors and provide suggestion to the user.

References

[1] Dey, A.K. and Abowd, G.D. "Towards a better understanding of context and context awareness". In Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, ACM Press, New York (2000)

[2] Henning Maass, "Location-aware mobile applications based on directory services". In Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 23-33, Budapest, Hungary, September (1997)

[3] W3C Composite Capability/ Preference Profiles (CC/PP), http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

[4] Held A. Buchholz S. and Schil A. "Modeling of context information for pervasive computing applications". In Proceedings of SCI 2002/ISAS (2002)

[5] User Agent Profile (UAProf) Specification, http://www.wapforum.org

[6] Chtcherbina E. and Franz M. "Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment". In Proceedings of International Conference on Advances in

Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet Italy, (2003)

[7] Henricksen K., Indulska J., and Rakotonirainy A. "Generating Context Management Infrastructure from High-Level Context Models". In Industrial Track Proceedings of the 4th International Conference on Mobile Data Management, Melbourne/Australia (2003)

[8] Halpin T. A. "Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design". Morgan Kaufman Publishers, San Francisco (2001)

[9] Bouzy B., and Cazenave T. "Using the Object Oriented Paradigm to Model Context in Computer Go". In Proceedings of Context'97 (Rio, Brazil, 1997)

[10] X.H. Wang, D.Q. Zhang, T. Gu, and H.K. Pung. "Ontology-Based Context Modelling and Reasoning using OWL". In Context Modelling and Reasoning Workshop (2004)

[11] D. Preuveneers, J. v.d.Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, and K. De Bosschere. "Towards an Extensible Context Ontology for Ambient Intelligence". In 2nd European Symposium on Ambient Intelligence (2004)

[12] S. R. Safavin and D. Landgrebe, "A survey of decision tree classifier methodology," IEEE Trans. Syst., Man, Cybern., vol. 21, no. 3, pp. 660–674, Jul. 1991.

[13] T. Gu, H.K. Pung, and D.Q. Zhang, "A Service-Oriented Middleware for Building Context-Aware Services". Journal of Network and Computer Applications (2005)

[14] Fahy P. and Clarke S. "CASS – a middleware for mobile context-aware applications", Workshop on Context Awareness, MobiSys (2004)

[15] Hofer T., Schwinger W., Pichler M., Leonhartsberger G. and Altmann J. "Context-awareness on mobile devices – the hydrogen approach", Proceedings of the 36th Annual Hawaii International Conference on System Sciences (2002)

[16] Hofer T., Schwinger W., Pichler M., Leonhartsberger G. and Altmann J. "Context-awareness on mobile devices – the hydrogen approach", Proceedings of the 36th Annual Hawaii International Conference on System Sciences (2002)

[17] Nguyen, D. (1998). Design patterns for data structures. In Proceedings of the 29th SIGCSE Technical Symposium on CS Education. ACM SIGCSE Bulletin, 30, 336–340.