FACE – a Knowledge-Intensive Case-Based Architecture for Context-Aware Services

Monica Vladoiu[1], Jörg Cassens[2], Zoran Constantinescu[3]

[1] PG University of Ploiesti, Bd. Bucuresti 39, 100680 Ploiesti, Romania
monica@unde.ro
[2] University of Lübeck, Ratzeburger Allee 160, 23538 Lübeck, Germany
cassens@imis.uni-luebeck.de
[3] Zealsoft Ltd., Str. Targu Neamt 11, Bucharest, Romania
zoran@zealsoft.ro

Abstract. Technological progress has made it possible to interact with computer systems and applications anywhere and any time. It is crucial that these applications are able to adapt to the user, as a person, and to its current situation, whatever that is. Contextual information and a mechanism to reason about it have demonstrated an important potential to provide solutions in this respect. This paper aims at providing an integrated CBR architecture to be used in context-aware systems. It is the result of our work to develop ePH, a system for building dynamic user communities that share public interest information and knowledge that is accessible through always-on, context-aware services.

Keywords: knowledge-intensive case-based reasoning, context-aware services, user modeling, context modeling, knowledge base

1 Introduction

Within our digitized and integrated world, the way we interact with computers has evolved so dramatically that we quite often have the impression that we live in a Star Trek-like environment. From the ugly and heavy computer on our desks to the slick and slim mobile devices that keep us connected all day around, the journey has been and still is quite challenging. Anyone can interact with computer systems and applications anywhere and any time. Though, there are some significant unknowns in this paradigm: what should be done, when, how and why [1].

Case-based reasoning is a problem-solving paradigm that is able to use the specific knowledge of previously experienced cases to solve new problems. A case refers to a concrete problem situation, which has been either previously experienced (past case) or newly occurred (new case). The new problem is solved by retrieving a similar past case from the case base and by reusing it in this new situation [2]. CBR provides for a mechanism of learning from experience, inspired by the way humans solve problems in real world domains [3, 4]. In this context, the term *problem solving* is utilized in a broader sense that complies with common practice in knowledge-based systems, i.e. problem-solving does not necessarily consist of finding a solution to a given problem

and it can refer to any problem put forward by the user (the justification of a user-proposed solution, the interpretation of a problem situation, the generation of a set of possible solutions etc.) [2].

During the last few years, case-based reasoning has proved itself as being one valuable approach for solving problems that occur in context-aware systems. Lee and Lee have developed a music recommendation system, which utilizes demographics, behavioral patterns and context of the user [5]. Kofod-Petersen illustrates the use of CBR problem solving and learning within tourist and hospital ward domains [6]. Corchado et al. [7] and Kofod-Petersen and Aamodt [8] demonstrate the use of CBR in health care environments. Benard et al. investigate the use of CBR as a mechanism that is able to select the appropriate behavior within collaborative and dynamic situations (virtual training environment) [9]. Kofod-Petersen and Mikalsen [1], and Avila and Cox report on their CBR approach of the travel domain [10]. Ma et al. [11] and Nguyen et al. [12] propose CBR approaches to smart home domains. Kwon and Sadeh [13] report on applying CBR and multi-agent systems to context-aware comparative shopping. Cassens and Kofod-Petersen investigate the importance of explanations for both the reasoning process and user communication in ambient intelligent systems [14]. Dong et al. adopt CBR to provide proactive component selection for mobile context-aware applications [15]. Zimmerman uses CBR to generate recommendations on audio to be listened in a mobile environment (art museum) [16]. Coutand et al. [17], and Sadeh et al. [18] use CBR to personalize location-aware services (message filtering).

This paper aims at providing an integrated CBR-based architecture to be used in context-aware systems. This architecture is the result of our work to develop the ePH system, which is a framework for building dynamic user communities that share public interest information and knowledge that is accessible through always-on, context-aware services [19, 20]. ePH is built around a user-centered digital library (called ePH-DLib) that stores regional information and knowledge. Its content is accessible through always-on context-aware services. Users can get it or enhance it, according to their location: at home or office by using a computer, on road with a specific GPS-based device in the car (called gipix, developed in-house), or off-line/off-road via mobile phone.

The digital library contains public interest information (drugstores, hospitals, general stores, gas stations, entertainment, restaurants, travel and accommodation, weather, routes etc.), historical, touristic, and cultural information and knowledge, users' personal "war stories" (tracks, touristic tours, impressions, photos, short videos and so on), and their additions, comments or updates to the content. This content is available to the ePH's users based on their context. For example, for a tourist being in a given area, the system may suggest several locations to go to (and actions to execute to reach them): a place to see, a restaurant to have lunch at, a museum or memorial house to visit etc. More, if a user is interested in something in particular, like mural art, and s/he is located near a place where such artifact is accessible and s/he can reach it within a reasonable time frame (having time to get back before dark), the system could show the tasks to be executed to guide her to reach that place. In a scenario that takes place in a remote mountain region, in which the fuel is going down rapidly, ePH shows on the car device where the nearest gas station is.

The ePH architecture incorporates the Communications Server, the Location Server, the CBR Engine, the Knowledge Base, the Context Middleware, and the multi-agent action subsystems [20, 21]. The *Communications Server (CS)* provides for the always-on kind of service, regardless of the location where the user is when s/he needs that service. The *Location Server (LS)* makes available the correct service according to the location. The *CBR engine* identifies the current problem situation, retrieves the past case that is the most similar with the one in progress, suggests a solution that uses that similar case, evaluates this solution, updates the system and learns from the current experience. If the new situation cannot be classified above a certain similarity threshold, then a new case is created and stored. The *Knowledge Base* includes general domain-dependent knowledge and specific knowledge (that is embodied by cases) that are used together to find the solution to a specific user's problem (therefore the ePH architecture is *knowledge-intensive*). The *Context Middleware* provides for context management by gathering and maintaining contextual information, and by freeing the agents and the applications of this chore. When the current context changes, the new context triggers a *multi-agent sub-system*, which contains various agents that deal with: the context, the CBR process, the task facilitation and decomposition, and the application-specific activities [21]. As *ePH-DLib* can be used both on- and off-line with ePH, it is not seen as strongly connected within this architecture.

The current stage of the project is as follows: the geospatial engine unde.ro provides the basic ePH functionality [19], the GPS car device, gipix, is in current use, and the critical cores of both the CS and the LS are functional as well. Some experimental results are also available [20, 21, 22]. Currently we are working on the development of the following modules: the CBR engine, the knowledge base and the context middleware. The rest of this paper is structured as follows: the next section gives a brief description of how case-based reasoning works. Section 3 illustrates the knowledge-intensive architecture of ePH's CBR engine. Section 4 presents some typical user scenarios and their related cases. The conclusions' section briefly summarizes the paper, and points out some future work ideas.

2 How CBR works

The CBR approach covers a large range of methods for organization, retrieval, use, and indexing of the knowledge retained from past cases. Cases can be preserved as concrete experiences or as *generalized cases* (sets of similar cases). They may be stored as individual knowledge units, or as smaller parts of them that are distributed within the whole knowledge structure. The cases may be indexed by a prefixed or open vocabulary. With regard to the solution from a past case, this may be directly applied to the current problem, or it may be adapted according to the differences between the two cases. The processes of case matching, solution adaptation, and learning from experience may be performed either by checking syntactic similarity or by using a strong model of general and domain knowledge. More, the CBR methods may be autonomous or they may interact heavily with the user, and past cases may be serially or parallel retrieved [2].

The general *CBR cycle* is usually seen as a dynamic model having four sub-processes: *retrieve* the most similar case(s), *reuse* the information and knowledge from that case(s) to solve the given problem, *revise* the proposed solution, and *retain* what is useful for future problem solving within the case-base [2, 23]. It all starts with a problem, whose initial description defines a *new case*. Then, this new case is used to retrieve a case (or more) from the stored previous cases in the case-base (provided that it can be classified above a given similarity threshold - otherwise the new case is stored as such). The solution of the retrieved case is *adapted* to match the peculiarities of the new case through reuse, and a *solved case* is obtained, namely a proposed solution to the current problem (*suggested solution*). During the revise process, this solution is put into test for success, either by being applied to the real world framework, or by being assessed by an expert. If the testing fails, the solution is altered. Useful experiences are retained (as *confirmed solutions*) for future use either in form of a new *learned case* or as modifications to already stored cases. To prevent degradation of the performance of the CBR system over time or to enhance it, maintenance has been identified as a key issue. Amongst the solutions that have been put forward is the proposal is to add two more processes into the CBR cycle beside retainment: *review* (monitoring the quality of the system knowledge) and *restore* (maintaining the case-base) [24].

3 FACE – a Knowledge-Intensive Reasoning Architecture

The main tasks the ePH's CBR engine has to deal with are as follows: identification of the current problem situation, retrieval of a past case that is similar to the new one, proposal of a solution to this problem, which uses that similar case, assessment of this solution, and update of the system by learning from the current experience. General domain-dependent knowledge and specific knowledge that is embodied by cases are used together in order to find the solution to a specific user problem (that defines the architecture as being *knowledge-intensive*). General domain knowledge may be combined with case-based reasoning in various ways: it can be used as an alternative problem solving method when the case-based method fails and/or it can be exploited within the case-base method itself [25]. The architecture that provides for this reasoning process is presented in Figure 1. We have called this architecture "FACE" to emphasize our aspirations to provide a knowledge-intensive reasoning process inspired by the way in which humans solve problems. For the rest of this section, we present briefly the main components of this architecture along with considerations with respect to their content.

Throughout this work we have considered the context definition from [1]: *context is a set of suitable environmental states and settings that concern a user, which are relevant for a situation-sensitive application during the process of adapting the services and the information that is offered to the user.* The context term is used dually here: first, it denotes what will be perceived from the real world (via *Context Middleware*) and will be stored in cases as *findings*, and, secondly, it refers to the available information when the problem is solved (leaving out what is not relevant to the task to be executed) [6]. More, the context can be seen on two level of abstraction:

a *base level*, where the context that is defined by specific elements (location, objects, persons etc.) resides, and a *conceptual level*, which focuses on the structure and relationships of the contextual information. It is important to notice that some knowledge may be context in one setting and domain knowledge in another [1, 27].

The CBR engine of ePH integrates the classical CBR cycle (Retrieve, Reuse, Revise, Retain) [2] with other reasoning paradigms (rule-based systems, model-based reasoning, deep models – like causal reasoning etc.), as well as other methods of generating knowledge (data-, text- or knowledge-mining). There is still to be evaluated whether ePH can benefit from the two extra-maintenance processes and in what way. The knowledge base incorporates general domain knowledge and case-specific knowledge. The general domain background knowledge can be acquired in a typical way for knowledge-based systems. There is also possible to learn general knowledge from the cases, in a case-based way or by induction [2].

Performance of (not only) context-aware systems could be improved if users were treated as individuals who have distinct personalities, abilities, goals etc. Every interactive computer system has a model of its users, being it implicit or explicit. Making it explicit provides for easier adaptation to different users and change over time. Therefore, before dealing with a person, the application needs to form a model about that person, by collecting a few specific pieces of information and by corroborating that with the knowledge it has about the groups to which the current person belongs. *User stereotypes* provide a useful mechanism to build such individualized *user models*. A stereotype is a cluster of characteristics (*facets*), which are specific to a certain group (of users), along with their specific values. In order to be useful in a computerized environment, stereotypes must be related to a set of triggers, namely "those events whose occurrence signals the appropriateness of particular stereotypes" [26]. Therefore, we need to keep user stereotypes and their specific triggers within the knowledge base, as it can be seen in Figure 1.

The knowledge base includes also the *initial cases*, pre-classified situations that have been acquired prior to first execution, the *point cases*, which are generated to incorporate a new occurent situation, and the *prototypical cases* that are generalized cases (aggregation of knowledge from previous point cases) [4, 6]. Once a new context is identified, the CBR engine tries to retrieve a known case and to classify the new situation relying on this case. After the successful classification of the current situation takes place, the new case will be stored in the case-base as a tuple that includes the contextual information that describes the situation, the problem that corresponds to this situation, and the constructed solution. When the ePH system makes a suggestion to its user, it implicitly predicts the user's behavior in the short term. As time goes by, and the system acquires new cases, it becomes possible to check whether a new case validates or invalidates that prediction. Therefore, the representation of *temporal knowledge* within the case base is necessary.

The contextual model subscribes to a meronomy that articulates various works from the literature [1, 9, 27, 28, 29] and is enriched to fulfill ePH's specific functionality. Thus, the context can be *personal* (user's interests, state of mind, expertise, limitations – time interval, location area etc., preferences, and so on), *social* (user's friends, family, colleagues, acquaintances etc.), *task* (user's activities, goals, operating mode – static or dynamic, and so on), *device* (mobile phone, gipix, PDA, laptop etc.), *environmental* (things, persons, services, weather etc. from user's

surroundings), *spatio-temporal* (time, user's location and movement), *strategic* (something important for a planned effect) and *historical* (for keeping trace of the past experience). These all relate to where the user is, when s/he is using the service, what s/he is using the service for, who s/he is with, what s/he likes etc. However considerations such as how young the user is, or whether it is snowing can be equally important. The *Context Interpreter* is designed to try to predict future intentions and actions of users. It gets one or more contextual entries and provides a single piece of context. The *Context Middleware* provides an easy to use, generic context management infrastructure that gathers and maintains contextual information, freeing the applications of this chore.

The middleware implements a *context space* [1], which is essential to capture both the *transient* (echoes the environment at a given point in time) and *persistent context* (represents a recurrent pattern of transient context) [1, 17]. The context space includes the context history, the current context and the context future. The *context history* helps applications to predict intentions and actions of the user by taking into account their previous contextual information. The results of this deduction process can be stored into the *context future*.

The *current context* consists of the currently relevant elements. When the current context "expires" it will be stored in the history for possible future reference. Each element of a context is represented by an *attribute* (physical or abstract object), its correspondent *features* (particular points of interest of the attribute within the given context) and the most appropriate *action* to be executed in this context [9]. Both attributes and features are described by a name, a value, a weight and a type (fuzzy, string, compound, exact). The user context is encapsulated within the cases to enable comparison between contexts, learning of user behavior and generation of case similarities–based recommendations.

To avoid the potential for infinite definitions of context, aka "a situation where everything is context", the *context representation* is restricted to the *context patterns* that comply with the *context templates*, which define contextual information in a domain dependent way. The *context validation* ensures that a given context instance is valid against a context template. More, the context that is gathered from various sources can be amalgamated via the *Context Merger* provided that the representations have the same structure [1].

The users can be part of some social network or they can be individual users, both types being covered by *context widgets* that are able to acquire particular context information and to make it available to the context-aware applications [1]. The context widgets operate independently from the applications and hide the distribution of the context sensing devices within the architecture from particular applications. Once the current context changes, the new context activates a *multi-agent sub-system*, which contains various agents that deal with: the context, the CBR process, the task facilitation and decomposition, and the application-specific undertakings.

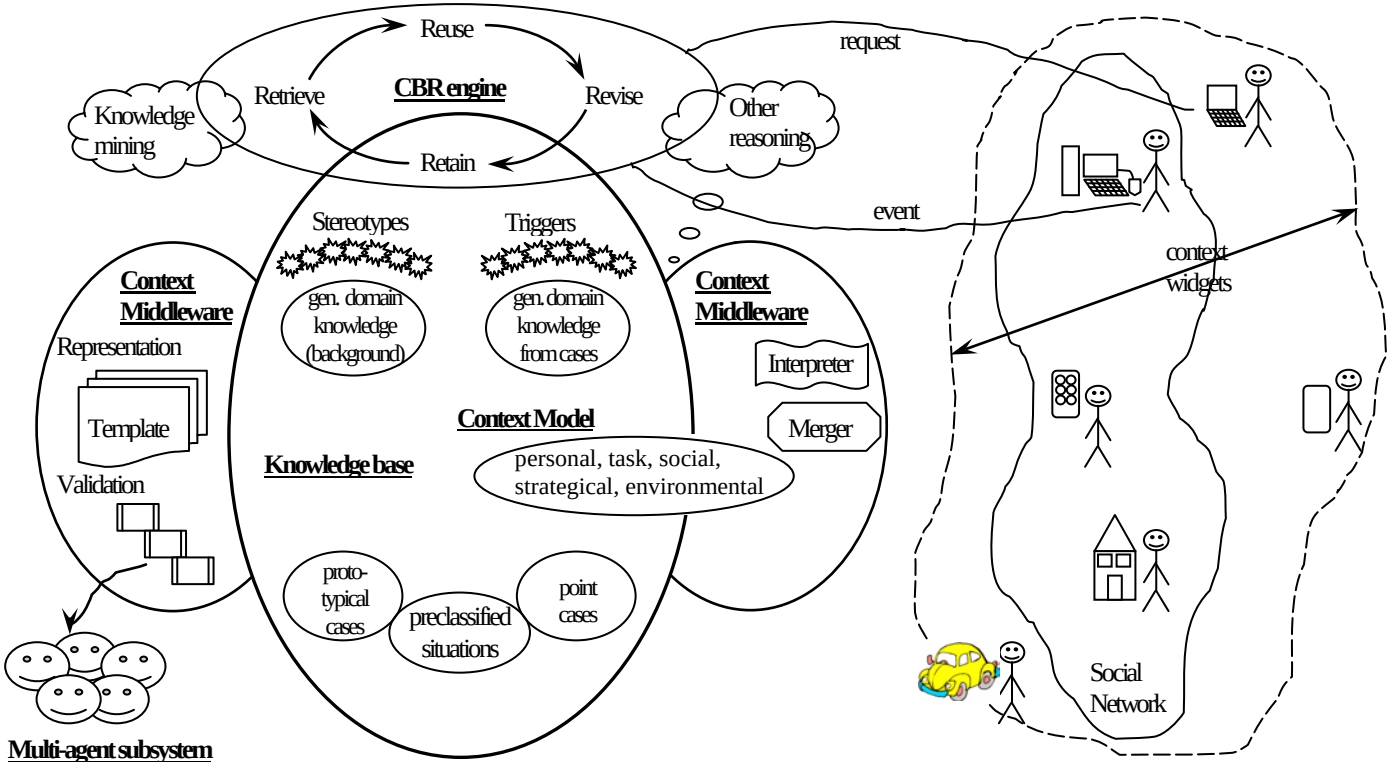


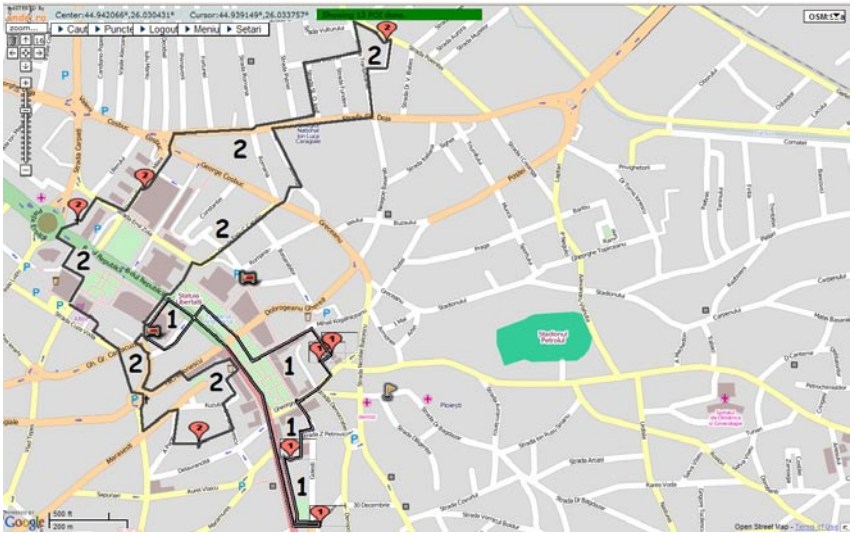
Fig. 1. FACE: a Knowledge-Intensive Reasoning Architecture for Context-Aware Services.

4 ePH User Scenarios and Cases

As shown briefly in the section that describes our system, ePH has a significant potential to support users in various ways: enhancing tourist experiences [20], enabling learning in multi-dimensional learning spaces [21], increasing traffic safety [22] etc. In this section, we present in more details the way in which a user who is interested in touristic attractions can benefit from interaction with ePH. The idea behind this kind of support has been to help a person who is at a given time in a certain location to experience as most as possible as a tourist, in a personalized and effective way, both in the real world and in the virtual one.

There are two significant user scenarios, each of them involving the opportunity to access whatever is relevant to one person's current interest within a given (real or virtual) area. First one is confined inside almost circular area (with a given radius), while the second one takes place along a particular segment of a track (with a given length). The system can support users to fulfill their specific goals in a context-aware fashion, by making recommendations on what is worth to be seen within the specified area, from a touristic point of view, and by showing the tasks to be executed to guide the user to reach that place. Let us consider two scenarios: first one with a person who is interested in visiting our county's capital (called Ploiesti) and would like help to organize and undertake a one-day personalized tour. The tour is supposed to take place in the town and in its surroundings (more or less circular area). In the second scenario, the user is interested in either a round-trip excursion or a trip along a main road, both spanning on a one-day period of time and within our county (Prahova). We assume that a distance that can be easily covered during daylight is around 150 km.

In the first situation, our user, let's call her Sofia, will be provided with the main Points Of Interest (POI) within the town area, along with their specific constraints (appropriate time to visit, ticket availability, and special offers). These points are grouped together in several one-day packages, from which Sofia can choose the most appropriate one according to her personalized option. For example, she can visit The Clock Museum, which is unique in Romania, The Art Museum, The History Museum, and the traditional products market from the city center (Figure 2, tour 1). In the market she can have a traditional snack, with sheep cheese and smoked mutton (by accessing the available glossary service she can find more about these meals). While moving from the History Museum to the market, Sofia will be passing by the Toma Caragiu Theater and she can get notification that there are still tickets for the evening representation. She can be pointed out that other online ePH friends are in the area and she can ask them if they want to join her for one or more of the undertaken activities. More, the POI specific restrictions are both displayed on her device and considered when ePH builds the one-day package. Another possible package includes The Memorial House of Nichita Stanescu (second major Romanian poet), The Central Market Hall (where she can also eat), and the Saint John Cathedral (Figure 2, tour 2). If she has interest in classical music, she can choose to close the day with a concert at The Paul Constantinescu Philharmonic Orchestra. The cases that are related to these scenarios are presented briefly in Fig. 2.

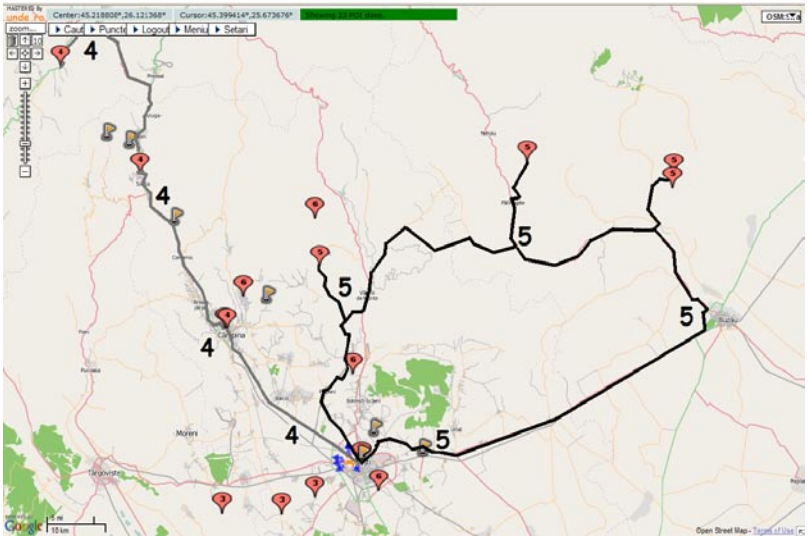


Case: Ploiesti 1	Case: Ploiesti 2
…	…
Context.task: 1 day town tour	Context.task: 1 day town tour
Context.operating_mode: static	Context.operating_mode: dynamic
Context.device: laptop	Context.device: PDA
Context.location: 44.9412,26.0213	Context.location: 44.9412,26.1345
Context.movement: no	Context.movement: yes
Context.time: 2010.03.20	Context.time: 2010.03.21
…	…
Context.interests: museum,tradition	Context.interests: buildings
Context.preferences: lunch,12pm-1pm	Context.preferences: concert,after 8
Context.interval: next day,9am-7pm	Context.interval: today
Context.location-area: around 10km	Context.location-area: around 10km
Context.weather: sunny day	Context.weather: sunny day
Context.friends: yes	Context.state_of_mind: joyful
…	…

Fig. 2. Two possible one-day town tours and the related prototypical cases

In the second scenario, the ePH user, Tudor, is offered more one-day trip packages. Tudor may be planning the trip prior to the journey itself or he might adjust his excursion dynamically, as he gets close to some POIs that are relevant to him. One possible package (3) includes the wisent reservation (European bison) at Bucsani, the Turnu monk monastery (where there are the ruins of five very old churches), and the Vacarescu Calimachi Castle in Manesti. The POIs in this package must be visited in this particular order, in any day but Monday, due to different constraints: the bison eat around 10, therefore is better to be in the reservation before that time, the monastery may be visited after the morning religious service is finished (after 12.30) and the castle is open before 17.00 (except for Monday). Other packages contain: (4) the haunted Iulia Hasdeu Castle, the memorial house of the painter Nicolae Grigorescu,

the Peles Castle in Sinaia, and the Dracula's Castle in Bran (Figure 3, trip 4), (5) the Muddy Vulcanoes in Berca, the Amber Museum in Scortoasa, and The Slanic Salt Mine (Figure 3, trip 5), (6) the Monastery tour: Ghighiu, Zamfira, Suzana and Crasna etc. While on road, the system can let Tudor know that in the vicinity there is a traditional fair taking place and, if he is interested in, ePH can guide him to get to that fair. The related cases are illustrated in Fig. 2.



Case: Prahova 4	Case: Prahova 5
…	…
Context.task: 1 day car trip	Context.task: 1 day car trip
Context.operating_mode: static	Context.operating_mode: dynamic
Context.device: laptop	Context.device: gipix,mobile phone
Context.location: 44.9412,26.0213	Context.location: 44.9331,26.1345
Context.movement: no	Context.movement: yes
Context.time: 2010.03.20	Context.time: 2010.03.20
…	…
Context.interests: castles, haunted	Context.interests: natural phenomena
Context.preferences: take away food	Context.interval: today
Context.interval: next day,9am-7pm	Context.limitations: max 150km
Context.limitations: max 150km	Context.weather: good
Context.weather: good	Context.friends: yes
Context.friends: no	Context.expertise: geological
…	…

Fig. 3. Two possible one-day trips and the related prototypical cases

5 Conclusions

Within this major shift from the desktop computer to the ubiquitous paradigm, the computer systems and applications are expected to adapt the personality of their users and to the current situation as opposed to the previous paradigm where the users were expected to adapt to the systems. CBR provides the means to solve a new problem by retrieving a previous similar situation and by re-using information and knowledge of that situation. CBR is suitable for open and ill understood domains, as it gains its expertise "through remembering the irregularities" [14], and it has proved its potential to development of context-aware applications.

The FACE architecture integrates the basic CBR approach with other reasoning paradigms, and subscribes to the general idea of unifying the problem solving and learning within one integrated knowledge framework. Future research has to be done into the *quality of context information* [29, 30], as an important parameter for modeling context, and how to integrate this within our system. Efforts have to be made towards the inclusion of an *inference mechanism* [31, 32] that enables derivation of context.

From the three features of a context-aware application [1, 12], 1) presentation of information and services to the user, 2) automatic execution of services, and 3) tagging of context – FACE provides only presentation of information and services, partially, for the automatic execution of a service for a user. Tagging of context to information to support later information retrieval is still to be achieved.

Future work needs to be done for better understanding of the relationship between problem solving and learning, and their integration into an autonomic framework, which provides for the system's ability to inspect its own behavior and to learn how to change its structure, in order to improve its future performance.

References

1. Kofod-Petersen, A., Mikalsen, M.: Context: Representation and Reasoning. Representing and Reasoning about Context in a Mobile Environment. Revue d'Intelligence Artificielle, Vol. 19(3), pp. 479-498 (2005)
2. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1): 39-59 (1994)
3. Anderson, J.R.: The Architecture of Cognition, Harvard University Press, Cambridge (1983)
4. Schank, R.: Dynamic memory; a theory of reminding and learning in computers and people. Cambridge University Press (1982)
5. Lee J. S., Lee J. C.: Context Awareness by CBR in a Music Recommendation System. LNCS, Vol. 4836/2010, Springer Berlin/Heidelberg (2010)
6. Kofod-Petersen, A.: Challenges in CBR for Context Awareness in Ambient Intelligent Systems. Int'l workshop on CBR and context awareness, CACOA 2006 (2006)
7. Corchado, J. M., Bajo, J., de Paz, Y.: A CBR System: The Core of an Ambient Intelligence Health Care Application. Soft Computing Applications in Industry, pp. 311-330 (2008)

8. Kofod-Petersen, A., Aamodt, A.: Contextualised Ambient Intelligence Through Case-Based Reasoning. ECCBR 2006, LNAI, vol. 4106, pp. 211-225, Springer Berlin/Heidelberg (2006)
9. Benard, R., Bossard, C., De Loor, P.: Context's Modeling for Participative Simulation. 9th Int'l Florida Artificial Intelligence Research Soc. Conf. FLAIRS 2006, pp. 613-618 (2006)
10. Muñoz-Avila H., Cox, M. T.: Case-Based Plan Adaptation: An Analysis and Review. IEEE Intelligent Systems 23(4), pp. 75-81, IEEE Press, New York (2008)
11. Ma, T., Kim, Tinghuai M., Yong-Deak K., Qiang M., Meili T., Weican Z., Context-aware implementation based on CBR for smart home. IEEE Int'l Conference on Wireless And Mobile Computing, Networking And Communications WiMob'2005 (2005)
12. Nguyen T. V., Woo Y. C., Choi D., CCBR: Chaining CBR in Context-Aware Smart Home, 1st Asian Conf. on Intelligent Information and Database Systems (2009)
13. Kwon, O., Sadeh, N.: Applying case-based reasoning and multi-agent intelligent system to context-aware comparative shopping, Decision Support Systems, Vol. 37(2), pp. 199–213 (2004)
14. Cassens, J., Kofod-Petersen, A.: Explanations and Case-Based Reasoning in Ambient Intelligent Systems. Int'l workshop on CBR and context awareness CaCoA 2007 (2007)
15. Dong F., Zhang Li., Hu D. H., Wang C-L.: A Case-Based Component Selection Framework for Mobile Context-Aware Applications. IEEE Int'l Symposium on Parallel and Distributed Processing with Applications ISPA 2009, pp.366-373, IEEE Press, New York (2009)
16. Zimmerman, A.: Context-awareness in user modeling: Requirements analysis for a case-based reasoning application. In: Ashley, K. D., Bridge, D. G., eds.: ICCBR 2003, LNAI, vol. 2689, pp. 718-732. Springer-Verlag, Heidelberg (2003)
17. Coutand, O. et al. : A CBR Approach for Personalizing Location-aware Services. Int'l workshop on CBR and context awareness, CACOA 2006 (2006)
18. Sadeh, N., Gandon, F., Kwon, O. B.: Ambient Intelligence: The MyCampus Experience. Technical Report CMU-ISRI-05-123, Carnegie Mellon University (2005)
19. Vladoiu, M., Constantinescu, Z.: Framework for Building of a Dynamic User Community - Sharing of Context-Aware, Public Interest Information or Knowledge through Always-on Services. 10th Int'l Conf. of Enterprise Information Systems ICEIS 2008, pp. 73-87 (2008)
20. Vladoiu, M., Constantinescu, Z.: Toward Location-based Services using GPS-based Devices, Proceedings of Int'l Conference on Wireless Network ICWN 2008 - World Congress on Engineering WCE 2008, Vol. I, pp. 799-804 (2008)
21. Vladoiu M., Constantinescu Z., Learning with a Context-Aware Multiagent System, 9th Romanian Educational Network International Conference RoEduNet, submitted (2010)
22. Vladoiu M., Constantinescu Z., Driving style analysis using data mining techniques, in Int'l Journal of Computers, Communications & Control (IJCCC), to be published (2010)
23. Shokouhi S. V., Skalle P., Aamodt A., Sormo F., Integration of Real-time Data and Past Experiences for Reducing Operational Problems, Proceedings of International Petroleum Technology Conference , Doha, Qatar 2009
24. de Mántaras R. L., et al.: Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Engineering Review, 20(3), pp. 215-240, Cambridge University Press (2005)
25. Sørmo F., Cassens J., Aamodt A.: Explanation in Case-Based Reasoning-Perspectives and Goals. Artificial Intelligence Review, 24(2), pp. 109-143, Springer Netherlands (2005)
26. Rich E.: User Modeling via Stereotypes, Readings in intelligent user interfaces, Morgan Kaufmann Publishers, pp. 329 – 342 (1998)
27. Brézillon P., Pomerol J.-C., Contextual knowledge sharing and cooperation in intelligent assistant systems, Le Travail Humain, 62(3), pp. 223–246 (1999)
28. Göker A., Myrhaug H. I., User context and personalisation, In Workshop proceedings for the 6th European Conference on Case Based Reasoning ECCBR 2002 (2002)

29. Chaari T., Dejene E., Laforest F., Scuturici V-M., A comprehensive approach to model and use context for adapting applications in pervasive environments, The Journal of Systems and Software, Vol. 80(12), pp. 1973-1992 (2007)
30. Bringel Filho J., Martin H., Towards Awareness of Privacy and Quality of Context in Context- Based Access Control for Ubiquitous Applications, Journal on Digital Information Management, vol. 7(4), pp. 219-226 (2009)
31. Qin W., Suo Y., Shi Y., CAMPS: A Middleware for Providing Context-Aware Services for Smart Space, LNCS 3947, pp. 644-653 (2006)
32. Jih W-r., Hsu J Y-j., Lee T-C., Chen L-I., A Multi-agent Context-aware Service Platform in a Smart Space, Journal of Computers, vol. 18 (1), pp. 45-59 (2007)