Contextual Modelling in Context-Aware Recommender Systems: A Generic Approach

Christos Mettouris and George A. Papadopoulos

University of Cyprus
Department of Computer Science, University of Cyprus, Nicosia, Cyprus
1 University Avenue, P.O.Box 20537, CY-2109
{mettour,george}@cs.ucy.ac.cy

Abstract. Context-aware recommender systems (CARS) use context data to enhance their recommendation outcomes by providing more personalized recommendations. Context modelling is a basic procedure towards this direction since it models the contextual parameters to be used during the recommendation process. Most literature works however build domain specific contextual models that only represent information of a particular domain, excluding the possibility of model sharing and reuse among other CARS. In this paper we focus on this issue and study whether a more generic modelling approach can be applied for CARS. We discuss a possible solution and show through literature review on relevant systems that the proposed solution has not yet been applied. Next, we present a novel generic contextual modelling framework for CARS, discuss its advantages and evaluate it.

Keywords: Context Modelling, Context Modelling Framework, Context-Aware Recommender systems, Context-Awareness, MDA.

1 Introduction

According to Adomavicius and colleagues [1, 2] important research issues related to Context-Aware Recommender Systems (CARS) have to do with contextual modelling, more important of which are how to model the context in order for a CARS to be able to use contextual information directly in (or prior/after) the recommendation process, and develop appropriate methods - or extend existing 2 dimensional ones (2D) to multidimensional (MD) - in order to include more dimensions than the user and item (i.e. the context) in the recommendation process. All aforementioned, as well as more challenges require that the context has been appropriately modeled.

Another critical contextual modelling issue has to do with the most common practice followed in recommender systems that model the context: developing domain specific models that only represent information of the particular application domain (e.g. music, restaurants nearby). Domain specific models cannot be applied in other domains, while most of them are also application specific, meaning that they cannot be applied to other recommenders even of the same application domain. By

A. Haller et al. (Eds.): WISE 2011 and 2012 Combined Workshops, LNCS 7652, pp. 41–52, 2013.

constructing application specific contextual models, many different and very specific models are produced with no reuse and sharing capabilities. Moreover, developers and researchers struggle to design their own models as they think appropriate and according to their own knowledge and skills, with no reference model to use, no guidance and strictly based on the application at hand, often resulting in the production of overspecialized, inefficient and incomplete contextual models.

We argue that the above contextual modelling problems can be addressed to some extent by defining a generic, abstracted contextual modelling framework for CARS: a model template in essence that will be able to uniformly model the most important contextual parameters for these systems and provide a good, solid reference to developers who will be able to use the framework and extend it both at model level and code level in order to build their own application driven models. Developers will be guided by the modelling framework, and through its objects, properties and relations, will be directed towards a more efficient, effective and correct selection and usage of context properties for their own application model. Moreover, the framework will introduce developers to modern concepts derived by CARS research that might not be familiar with, such as the "context dependent rating data" [5], the "supposed context" [3], the "static/dynamic context", the "context weights", etc., as well as the role such concepts can play in a context model and a recommendation process.

Researchers will benefit as well from such a contextual modelling framework for CARS. The framework will provide a spherical view of CARS research and its concepts, assisting new researchers in understanding these concepts, as well as research problems, issues and challenges. The modelling framework will inevitably project any inconsistencies that might occur after an addition of a new concept, and therefore model corrections after additions will be made easier. The most important advantage though of using a generic contextual modelling framework is that following research works can be based on this abstracted framework, enhance and extend it in order to solve important contextual modelling problems in CARS research, while avoiding the risk of being over-focused on a particular domain.

This work, through the bibliographic review of section 2, shows that a generic contextual model, model template or modelling framework for CARS does not exist in the bibliography, and proceeds with our first attempt towards this goal. In particular, section 3 provides our first attempt to design and build a generic contextual modelling framework for CARS. In section 4 we theoretically evaluate the proposed framework by using existing works from the CARS literature, showing how the framework can be used to model application specific CARS and novel research modelling methods. Section 5 completes the paper with conclusions and future work.

2 Related Work

In order to infer whether any attempts towards a generic contextual model for CARS exist in the bibliography, we have reviewed recommender systems that use contextual and conceptual models. Towards this direction, a number of CARS have been reviewed, as well as semantic recommender systems, since many semantic recommenders use semantics to model information, including the context. The review

was focused on whether the contextual or conceptual model used was application specific or generic. To better define the terms "application specific" and "generic", we use the definition of Peis [18]. Peis and colleagues classify semantic recommenders as *generic recommender systems* those that do not focus extensively on a particular domain and as *domain specific recommenders* those that do. Examples of each class of recommenders can be found at [18]. Peis's categorization into generic and domain specific can be applied to all semantic and context aware recommenders: such systems either apply to some generic application area (generic systems), such as recommendation of products, web services, etc. or apply to a particular domain (domain specific systems) such as recommendation of movies, books, etc. In this work, we categorize systems as generic and domain specific based on Peis's categorization, with one additional condition: we also categorize systems that attempt to facilitate *any* application specific domain as generic recommenders.

Our review revealed that most CARS and semantic recommenders in the literature are domain specific, which confirms our initial statement [4, 6, 7, 8, 9, 11, 19, 20, 21, 23]. Unfortunately, the models derived from such works cannot be applied for use in other domains. A number of generic recommenders also exists [10, 13, 16, 22] that either apply to some generic application area, or can be applied to more than one domain by linking domain specific ontologies to their own data and knowledge pool in order to gain domain-aware knowledge and provide domain-aware functionality. One of the most representative examples of a generic recommender system is the one of Loizou and Dasmahapatra [15], who propose a generic, abstracted model in the form of an ontology which could be used by many different types of recommender systems and which ontologically models, not only data and context, but also the recommendation process.

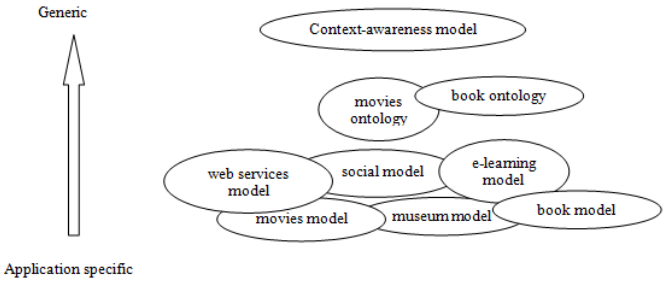

Fig. 1. How conceptual models for recommender systems move from application specific to generic

Although some of the semantic and contextual models try to be more generic, the majority represent information that either concern particular application domains (e.g. tourism, movies, museums), or more abstracted domains (e.g. products in general, web services, e-learning, etc.). Moreover, a common practice is to use general purpose ontologies for facilitating sharing and reuse among semantic recommenders. The aforementioned are depicted in Fig. 1. Lower are shown conceptual models of recommender systems that are application specific. A step higher, but still application specific are conceptual models that can be adopted by a number of recommenders

such as web service recommendations, e-learning recommendations, etc. Examples of generic conceptual models are the general purpose ontologies, which truly facilitate sharing and reuse among many recommenders.

3 A Generic Contextual Modelling Framework for Cars

To the best of our knowledge no attempts have been made towards developing a truly generic contextual model for CARS that will define the basic contextual entities of such systems, their properties and associations so that CARS will be able to *extend* this model to construct application specific models for the needs of the application at hand. A generic contextual model would *simplify* the process of contextual modelling in CARS and enable context *uniformity*, *share* and *reuse*. Moreover, it would introduce developers and new researchers to important concepts of CARS research in order to assist them in building more effective context-aware recommenders, while researchers will be aided by using the model to apply their solutions to research problems relevant to CARS context modelling. Fig. 1 displays the generic nature of the proposed contextual model ("Context-awareness model"). The model has to be generic enough to be able to describe any contextual definition related to CARS. In this work we propose such a model in the form of a modelling framework.

3.1 The Modelling Framework

The modelling framework is essentially a model template, which itself is an abstracted model designed and built as a UML class diagram by using the Eclipse Modeling Framework (EMF) [12]. EMF is a Java framework and code generation facility for building tools and applications based on a model. It provides the means to transform a model into customizable Java code. After designing our framework as an EMF model, we have used the EMF generator to create a corresponding set of Java implementation classes. We have used the EMF tool for three main reasons: (i) so that our framework could be easily transformed into Java code in order to be used by CARS developers in a straightforward way, (ii) in order to be highly extendable and customizable, since the generated code can be easily extended and modified and (iii) EMF provides the opportunity to edit the generated code classes by adding methods and variables and still be able to regenerate code from the modelling framework, as all additions will be preserved during the regeneration. In this way, developers are able not only to extend the code generated from the framework, but also to extend the framework as well, regardless of any code extensions that might occur before the framework extension.

Fig. 2 depicts the contextual modelling framework for CARS. The boxes represent context entities (or classes). There exist two types of associations: the solid arrow from entity "a" to entity "b" (e.g. from "Context" to "itemContext") depicts that "b" is a context property of "a". The other type of arrow from "a" to "b" (e.g. from "itemContext" to "Item") depicts that "a" is a subclass of, and therefore inherits class "b".

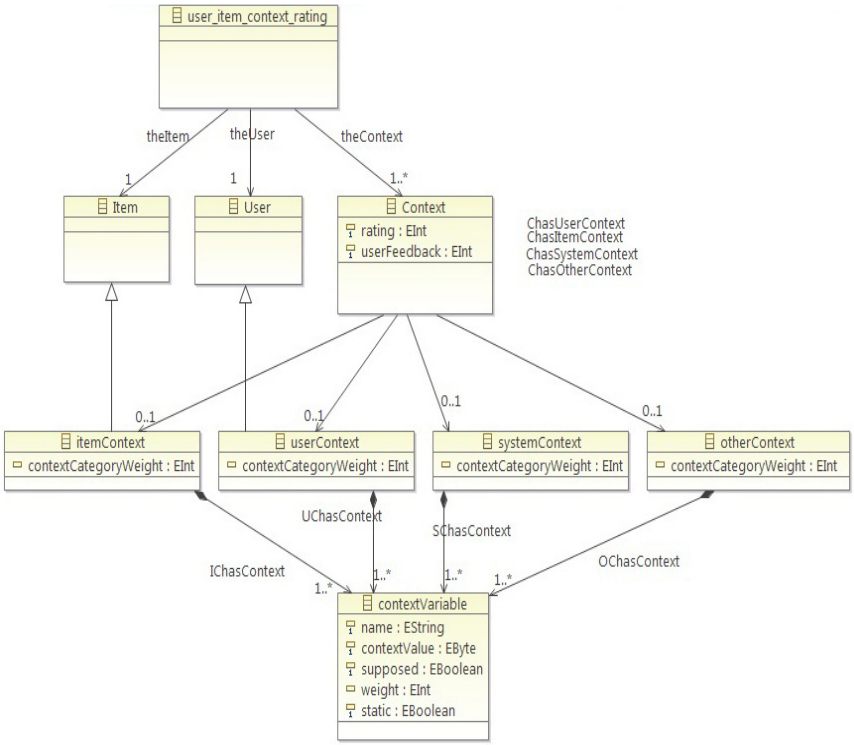

Fig. 2. The proposed contextual modelling framework for CARS

From top-to-bottom, the level of abstraction decreases. The top entity "user_item_context_rating" reflects on the fundamental concept of CARS research: to include the context in the recommendation process in order to result from the 2D un-contextual recommenders: Users × Items → Ratings to the multidimensional context-aware recommenders: Users × Items × Context → Ratings [2]. The "user_item_context_rating" entity represents a single complete recommendation process. For each recommendation attempt, a recommender must examine whether an item is suitable for a user in a certain context. This can be depicted through the question: "what is the rating a particular user would assign to a particular item under a certain context"? This rating score is what a recommender must calculate. Therefore, the "user_item_context_rating" entity has exactly one "Item", exactly one "User" but one or more "Context" entities, each including a property "rating" to ensure that in each context a user is able to rate the same item with a different rating score. In CARS literature this is known as "Context dependent ratings" [5]. Such user ratings are assigned on items in particular contexts: a user may rate multiple times an item, each rating taking place in a different context. Baltrunas and colleagues [5] note that user preferences (and hence ratings) on items depend on the context and therefore

context dependent rating data on items should be available. With the aforementioned configuration, our modelling framework provides the means for including "Context dependent ratings".

The "Context" entity in Fig. 2 represents the *context instance*: the set of context variables with their corresponding values that constitute the context for a single recommendation. The "Context" entity also includes the "userFeedback" property that represents user feedback information for a particular recommendation. As with the "rating" property, "userFeedback" is unique for a particular context but many "userFeedback" can exist for one user and one item: in many contexts.

Regarding the entities "Item" and "User", from a context-awareness point of view, we are only interested in the context information related to the item and the user at hand. This context information is represented in the framework by "itemContext" and "userContext" entities: these entities represent any item and user related context information that participates in a particular context instance. "itemContext" and "userContext" are subclasses of "Item" and "User" classes respectively, inheriting their characteristics to use them as context information, as well as extending and overwriting information and functionality as appropriate. Similarly, "systemContext" represents any system related context information that participates in a particular context instance, while "otherContext" represents any contextual information other than item, user and system context (e.g. weather, time, temperature). "itemContext", "userContext", "systemContext" and "otherContext" constitute the four *main context classes* in our modelling framework and are meant to be perceived as the main context entities for any contextual model of CARS; any context information of any CARS should be able to be represented as a context property of one (or more) of the main context classes, as an entity of the type "contextVariable". The entity "contextVariable" can be a context property of any one of the four main context classes (or more than one in the case where the main context classes share a contextual information).

The "Context" entity may include zero or one of any of the four main context classes (via the corresponding associations "ChasUserContext", "ChasItemContext", "ChasSystemContext" and "ChasOtherContext" appeared to the side of "Context" in Fig. 2 for readability reasons); we have provided the zero possibility in the case where a CARS does not need to use the particular context class for any reason.

Each of the four main context classes mentioned above includes zero or one weight property so that developers are able to denote the importance of each class, and by this provide a hierarchy about which context class(es) is(are) more important. This importance hierarchy is necessary to be included in the recommendation process.

The "contextVariable" entity is positioned at the lowest level of the framework representing the least abstracted entity: the context variable that contains the actual contextual information. Each main context class has to include at least one "contextVariable" entity. Each "contextVariable" has a name and a value, while a weight property is also included in case the CARS developer would like to denote a particular importance for a certain variable. The property "static" refers to whether the context is static or dynamic (static=true/false). Static context cannot change dynamically (e.g. user profile information), as opposed to dynamic that can (e.g.

weather). Hinze and Buchanan [14] propose that context-awareness can help in reducing the amount of data to be accessed real time by pre-retrieving any relevant pre-known data, e.g. the static context. This increases efficiency. By using the "static" attribute, a CARS developer may assign to context variables whether they are a part of static or dynamic context and by that, specify a different functionality for them.

Asoh and colleagues investigate in [3] whether the answers of users during questionnaires about their preferences on items differ when they are in a given context, as opposed to not actually being in that context but only imagine being in it (supposed context). Their results suggest that the ratings of users in supposed contexts may be different than the ratings of the same users in real contexts. Their findings are very important since much information about user preferences often results from user questionnaires on supposed contexts, meaning that these results could be proved misleading, even false. Back to our modelling framework, the purpose of the property "supposed" is to denote whether the particular context variable is a part of the supposed context. Since supposed context usage may negatively influence the rating of a user, it is very important that this type of context can be represented in any context model. In the case where a recommender includes many "supposed" context information, the recommendation results could be misleading. Thus, a context model has to be able to denote whether the context it represents can be fully trusted or not; we use the "supposed" property of each "contextVariable" entity to denote whether the particular context variable can be fully trusted (supposed=false) or needs caution (supposed=true).

3.2 Code Generation

The contextual modelling framework for CARS presented above can be used to automatically generate the Java code by using the EMF. For each of the 9 classes in the diagram of Fig. 2, a Java interface is created, as well as a Java implementation file containing the implementation code. All aforementioned files are generated automatically by the Eclipse tool. For example, for the class "contextVariable" a file `contextVariable.java` is created that contains the "contextVariable" interface, and another one named `contextVariableImpl.java` that contains the interface implementation code. `contextVariable.java` includes the abstracted methods (setter/getter), as well as the variables of the class. The `contextVariableImpl.java` includes the implementations of the abstracted interface methods. The automatically generated Java code can be freely extended and modified by developers in order to become tailored to a specific application domain. Developers and researchers are free to implement the abstracted methods of the interface as they think appropriate, modify them, as well as extend the code by adding new methods and variables. Moreover, the EMF provides the ability to change the modelling framework of Fig. 2 and regenerate the code multiple times. In the case a method was manually changed by the developer prior to a code regeneration, the EMF can be prevented from overwriting the particular method (the developer must remove the tag "@generated" from the particular method).

4 Theoretic Evaluation

Proper evaluation of the modelling framework would require application of the framework in real scenarios, i.e. making the modelling framework publicly available and invite CARS developers to use it for their applications. The aforementioned would provide us with good feedback regarding the framework's strong and weak points, as well as any shortfalls. Due to lack of time, we leave this as future work. In order to theoretically evaluate our framework, we have used three research works from the context-aware recommendations literature: an application specific context-aware recommender system and two research oriented works. The aim of the theoretic evaluation is, on the one hand, to examine whether our generic contextual modelling framework for CARS is able to successfully model the context used by these systems and how this can be done, and on the other hand to observe whether the framework can be used for realizing novel research-oriented context modelling methods.

4.1 Modelling an Application Specific CARS

We have chosen the media recommendation system of Yu and colleagues [23] because the authors consider four different types of context, while most works consider significantly less. The four types are: content context, operating context, user context and terminal context. We will focus on context modelling by examining whether our contextual modelling framework for CARS could be used to successfully model the context in the particular application. According to the authors, content context is the context of an item, operating or situational context is the user's current location, time and activity, user context consists of user preferences and terminal context is relevant to terminal capabilities.

We aim to observe whether our modelling framework for CARS can be used to represent the four different types of context. Starting with content context, our framework provides the context entities "Item" and "itemContext" which can be used for modelling the items as follows: generic item characteristics can be assigned to class "Item" while strictly contextual information can be assigned to the class "itemContext". Note that a context instance includes only one "Item", one "itemContext" but many "contextVariable" entities representing the many pieces of context information related to that item. A contextual information that is assigned as context property to the class "itemContext" is in essence a "contextVariable" assigned to the class "itemContext" via the association "IChasContext" (Item Context hasContext). For example, to assign the contextual information "actor" to "itemContext", the developer will create a "contextVariable" by the name of "actor" and will assign it to "itemContext" via the association "IChasContext". The benefit of our approach is that any context entity represented as context variable can be assigned as property to any of the four main context classes of our framework, even to more than one. This provides developers with the freedom to assign context variables to main context classes as they think appropriate and according to the application at hand. In the case where a context variable is assigned to many context classes, then the developer can specify a different functionality for the particular variable according

to the context class currently used (e.g. context variable "user's activity" can be treated differently when perceived as part of "userContext" and in another way when perceived as part of "systemContext").

The next context type is the operating or situational context that includes the user's current location, time and activity. Such context type does not exist as a single entity in our system, since situational context can vary a great deal among different applications and domains. Instead, we chose to contextually model only entities that are well defined and not controversial among different domains. Hence, we model such context by denoting it as non static context in "contextVariable" entity. By this approach, *any* context information of any type can be denoted as dynamic, which is a developer's decision. Regarding the work under examination, situational context is modeled by our framework as follows: "user's current location" and "user's activity" are context properties under "userContext" and possibly under "systemContext" and "otherContext" (in case user's current location and activity also affect system functionality and other events), while "time" is an "otherContext" property. Finally, "User preferences" are assigned under "userContext" and "terminal context" is assigned under "systemContext".

Regarding "contextVariable" properties "supposed", "weight" and "static", we assign "supposed=false", "weight=0...10" depending on the perceived importance for each one of the context variables by the developer and specify "static=true" for static context: any contextual information regarding the item ("itemContext", e.g. actor, genre, language), a part of "userContext" that is static ("User preferences") and the "systemContext" which is mainly static. "static=false" is specified for dynamic context such as the operational/situational context (see paragraph above).

From the above discussion we state that the proposed generic contextual modelling framework for CARS is able to model the context as specified by the system of Yu and colleagues, and result in an application specific model for media recommendations, which we name as "media model". The advantage of using the contextual modelling framework for CARS instead of the model proposed by Yu and colleagues is that the resulting "media model" allows for sharing and reuse among various applications, and can easily be further extended and modified to suit the developer's demands. Moreover, both the generic contextual modelling framework for CARS and the application specific "media model" can be used as reference and guidance by developers to implement their own application specific models for media recommenders.

4.2 Modelling Research Oriented Works

For each user u and context k, Panniello and Gorgoglione [17] define the user profile in context k, i.e. the contextual profile Prof(u, k). For example, if the contextual variable "Season" has two values (e.g., "Winter" and "Summer"), then the authors assign two contextual profiles for each user, one for the winter and the other for the summer and use the appropriate one according to the context. Similarly,

Baltrunas and Amatriain [4] propose using micro-profiles of the user, which are snapshots of the user profile in certain time periods (e.g. morning, noon, night), instead of using the whole user profile (they use the time as context). By using the reduced time-based micro-profile of the user instead of the whole profile they manage to reduce the input dataset of the recommendation algorithm and thus improve accuracy. Both approaches are based on the same idea: using specific user profiles that are defined based on a particular context instance instead of using the whole user profile to provide recommendations.

After studying the above contextual modelling approaches, we examined whether they could be successfully implemented using our generic contextual modelling framework for CARS. In the framework we provide researchers the ability to define in their models a user profile instance for a particular context instance by using the "userContext" entity which is directly associated with the "Context" entity (see Fig. 2). An instance of the context is composed of all context variables associated with it having a particular value. Schematically, we can say that an instance of the context is a "Context" entity composed of all the "contextVariable" entities associated to it through the four main entities "itemContext", "userContext", "systemContext" and "otherContext" (Fig. 2). On the other hand, the contextVariable: name="Time", value="morning" participates in a number of context instances, each of which is valid when the time is morning. These context instances define the context: Time="morning". The same applies for each "contextVariable" in the modelling framework. In the case where a researcher needs to use time-based user micro-profiles, our framework provides by itself such functionality as follows (suppose time is divided to 3 distinct time slots: morning, noon and night): define three context variables, one for each time slot: contextVariable: name="Time" value="morning", contextVariable: name="Time" value="noon", and contextVariable: name="Time" value="night". Each of the three "Time" contextVariable entities corresponds to a different context in respect to time: morning, noon and night. For each of the three "Time" contextVariable entities, a number of context instances are created which are valid for the particular contextVariable's value. These context instances however, also include a "userContext" entity that contains the user context information that is valid for the particular context instance, and consequently for the particular time. In this way, user context-aware time-based micro-profiles are automatically constructed through the modelling framework. Similarly, by selecting a different context variable than time, e.g. season, we can automatically produce context-aware season-based micro-profiles of the users (or any other context entity).

The advantage of using our modelling framework is that, automatically and by default, the framework's context instances define all valid contextual information around a fact or event (in the example above around a specific time slot). Hence, by using the framework, a researcher is given the opportunity to explore more easily and straightforward the benefits of context-awareness, as in the example above where time-based user micro-profiles are automatically created through the framework.

5 Conclusions and Future Work

After confirming that no existing work attempts to define a generic contextual model for CARS, in this paper we have proposed such a model in the form of a contextual modelling framework and theoretically evaluated it with positive results. As future work we will conduct practical evaluation of the modelling framework by applying it in real scenarios. To test whether our framework is indeed capable of facilitating any CARS, we aim to make the framework publicly available and invite CARS developers to use it for their own applications. This will provide us with valuable feedback about the framework's strong, weak points and shortfalls. Moreover, we will extend our modelling framework by conceptually modelling functionality (i.e. recommendation algorithms) in addition to the context. The goal is to research whether by including conceptual sub-models of the various recommendation algorithms in the framework the implementation of more efficient CARS can become easier for developers and researchers. This can possibly lead to a fully model based CARS development, which is a very important concept to be studied.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) 23, 103–145 (2005)
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 335–336 (2008)
3. Asoh, H., Motomura, Y., Ono, C.: An Analysis of Differences between Preferences in Real and Supposed Contexts. In: Proceedings of the 2nd Workshop on Context-Aware Recommender Systems (2010)
4. Baltrunas, L., Amatriain, X.: Towards Time-Dependant Recommendation based on Implicit Feedback. In: Workshop on Context-Aware Recommender Systems, CARS 2009 ACM Recsys, vol. 2009, pp. 1–5 (2009)
5. Baltrunas, L., Kaminskas, M., Ricci, F., Rokach, L., Shapira, B., Luke, K.: Best Usage Context Prediction for Music Tracks. In: Proceedings of the 2nd Workshop on Context-Aware Recommender Systems (2010)
6. Blanco-Fernández, Y., Pazos-Arias, J.J., Gil-Solla, A., Ramos-Cabrer, M., Barragáns-Martínez, B., López-Nores, M., García-Duque, J., Fernández-Vilas, A., Díaz-Redondo, R.P.: AVATAR: An Advanced Multi-Agent Recommender System of Personalized TV Contents by Semantic Reasoning. In: Zhou, X., Su, S., Papazoglou, M.P., Orlowska, M.E., Jeffery, K. (eds.) WISE 2004. LNCS, vol. 3306, pp. 415–421. Springer, Heidelberg (2004)
7. Bogers, T.: Movie Recommendation using Random Walks over the Contextual Graph. In: Proceedings of the 2nd Workshop on Context-Aware Recommender Systems (2010)
8. Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., He, X.: Music recommendation by unified hypergraph. In: MM 2010 Proceedings of the International Conference on Multimedia, p. 391 (2010)
9. Cantador, I., Castells, P.: Semantic Contextualisation in a News Recommender System. In: Workshop on Context-Aware Recommender Systems CARS 2009 in ACM Recsys, vol. 2009 (2009)

10. Costa, A., Guizzardi, R., Guizzardi, G., Filho, J.: COReS: Context-aware, Ontology-based Recommender system for Service recommendation. In: UMICS 2007, 19th International Conference on Advanced Information Systems Engineering, CAISE 2007 (2007)
11. Drumond, L., Girardi, R., Leite, A.: Architectural design of a multi-agent recommender system for the legal domain. In: Proceedings of the 11th International Conference on Artificial Intelligence and Law, ICAIL 2007, p. 183 (2007)
12. Eclipse Modeling Framework Project (EMF), http://www.eclipse.org/modeling/emf/
13. Emrich, A., Chapko, A., Werth, D.: Context-Aware Recommendations on Mobile Services: The m:Ciudad Approach. In: Barnaghi, P., Moessner, K., Presser, M., Meissner, S. (eds.) EuroSSC 2009. LNCS, vol. 5741, pp. 107–120. Springer, Heidelberg (2009)
14. Hinze, A., Buchanan, G.: Context-awareness in mobile tourist information systems: challenges for user interaction. In: International Workshop on Context in mobile HCI at the Conference for 7th International Conference on Human Computer Interaction with Mobile Devices and Services, Salzburg, Austria (September 2005)
15. Loizou, A., Dasmahapatra, S.: Recommender Systems for the Semantic Web. In: ECAI 2006 Recommender Systems Workshop, Trento, Italy, August 28- September 11 (2006)
16. Moscato, V., Picariello, A., Rinaldi, A.M.: A recommendation strategy based on user behavior in digital ecosystems. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems, MEDES 2010, p. 25 (2010)
17. Panniello, U., Gorgoglione, M.: A Contextual Modeling Approach to Context-Aware Recommender Systems. In: Proceedings of the 3rd Workshop on Context-Aware Recommender Systems (2011)
18. Peis, E., Morales-del-Castillo, J.M., Delgado-López, J.A.: Semantic Recommender Systems. Analysis of the State of the Topic [en linea]. Hipertext.net (6) (2008), http://www.hipertext.net
19. Santos, O.C., Boticario, J.G.: Modeling recommendations for the educational domain. Procedia Computer Science 1(2), 2793–2800 (2010)
20. Sielis, G.A., Mettouris, C., Papadopoulos, G.A., Tzanavari, A., Dols, R.M.G., Siebers, Q.: A Context Aware Recommender System for Creativity Support Tools. Journal of Universal Computer Science 17(12), 1743–1763 (2011)
21. Sielis, G.A., Mettouris, C., Tzanavari, A., Papadopoulos, G.A.: Context-Aware Recommendations using Topic Maps Technology for the Enhancement of the Creativity Process. In: Educational Recommender Systems and Technologies. IGI Global (2011)
22. Uzun, A., Räck, C., Steinert, F.: Targeting more relevant, contextual recommendations by exploiting domain knowledge. In: HetRec 2010 Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, pp. 57–62 (2010)
23. Yu, Z., Zhou, X., Zhang, D., Chin, C., Wang, X., Men, J.: Supporting Context-Aware Media Recommendations for Smart Phones. IEEE Pervasive Computing 5(3), 68–75 (2006)