The UbiCARS Model-Driven Framework: Automating Development of Recommender Systems for Commerce

Christos Mettouris[1(✉)], Achilleas Achilleos[2], Georgia Kapitsaki[1], and George A. Papadopoulos[1]

[1] Department of Computer Science, University of Cyprus, 2109 Nicosia, Cyprus
{mettour, gkapi, george}@cs.ucy.ac.cy
[2] Frederick University, 7 Y. Frederickou Str, 1036 Nicosia, Cyprus
com.aa@frederick.ac.cy

Abstract. Recommendations of products to customers are proved to boost sales, increase customer satisfaction and improve user experience, making recommender systems an important tool for retail businesses. With recent technological advancements in AmI and Ubiquitous Computing, the benefits of recommender systems can be enjoyed not only in e-commerce, but in the physical store scenario as well. However, developing effective context-aware recommender systems by non-expert practitioners is not an easy task due to the complexity of building the necessary data models and selecting and configuring recommendation algorithms. In this paper we apply the Model Driven Development paradigm on the physical commerce recommendation domain by defining a UbiCARS Domain Specific Modelling Language, a modelling editor and a system, that aim to reduce complexity, abstract the technical details and expedite the development and application of State-of-the-Art recommender systems in ubiquitous environments (physical retail stores), as well as to enable practitioners to utilize additional data resulting from ubiquitous user-product interaction in the recommendation process to improve recommendation accuracy.

Keywords: Ubiquitous product recommendation
Intelligent recommendations in physical stores · UbiCARS · Modelling

1 Introduction

As customers nowadays have the option to select from a huge variety of high quality products at competitive prices, retailers are in need to provide better service to win the trust of customers and achieve a sustainable customer relationship. Information systems can play a significant role in "sensing" what customers like and the sales trend. When information systems are not used and thus information cannot be acquired immediately, customer demands cannot be met in real time, risking losing customers' interest in shopping [1]. As [1] notes, with the change of market patterns and customer demand, it is particularly necessary for the retail industry to provide a pleasing, safe and convenient shopping environment, with consideration for customers.

© Springer Nature Switzerland AG 2018
A. Kameas and K. Stathis (Eds.): AmI 2018, LNCS 11249, pp. 37–53, 2018.
https://doi.org/10.1007/978-3-030-03062-9_3

Ambient intelligence (AmI) and ubiquitous computing characterize intelligent, pervasive and unobtrusive computer systems embedded into human environments, tailored to the individual's context-aware needs [2]. While AmI facilitates users to smoothly interact with the environment by means of intuitive interfaces embedded in objects, still the information generated by these interactions is not fully utilized. Another important aspect of AmI is *intelligence* by means of machine learning, agent-based software, and robotics [3]. This paper proposes the application of AmI concepts for the exploitation of generated user-product interaction information in physical stores, as well as the usage of State-of-the-Art intelligent recommendation techniques in physical and electronic environments for commerce.

Recommender systems (RSs) have been the answer to the information overload modern life consumers experience for more than two decades. RSs are essentially software tools able to discover the necessary knowledge about users in order to offer personalized recommendations to them. It is proved that, in e-commerce settings, recommendations of products to customers boost sales [4, 5], increase customer satisfaction [4] and improve user experience [6]. Therefore, not only customers benefit from recommender systems, but on-line retail stores as well [5].

While e-commerce sales grow exponentially[1], physical shopping is still the main shopping mode in comparison to e-commerce. Recent technological developments in AmI and ubiquitous computing though suggest that the success of RSs in the virtual world can be replicated in the real world, i.e., in physical retail settings. There already is a trend towards personalisation in the physical retail market and RSs with ubiquitous computing can be the means to make it more sophisticated and effective [5].

While RSs for e-commerce use information about user interaction on items online such as product ratings and purchase history to compute personalized recommendations of products to users, in ubiquitous settings other context-aware methods are applied to track user interest on products, such as tracking customers' in-store shopping path or the customers' staying time in the product area [12, 17]. The aim is to recommend to users similar products or other related things such as brand stores in a shopping mall or provide reviews and ratings on a product. However, to the best of our knowledge, none of the works in the literature uses a combination of user-item interaction data from both the physical scenario and the online scenario with the aim to enhance the recommendation accuracy in both settings. This constitutes the first contribution of this work.

Although open source recommendation frameworks are available (e.g. EasyRec, LensKit, LibRec[2]) for retail businesses to use as recommendation engines in building their own RS applications, it is nevertheless difficult for practitioners (software engineers, e-store developers) that are non RS experts to achieve such a task [7, 8]. These frameworks do not offer an abstraction from the technical details concerning the embedding of recommendations to an e-store, requiring from practitioners to work on code level. Moreover, and more importantly, they do not deal with the ubiquitous recommendation scenario to track user-item interaction at the physical store. Especially, since research has proven that Machine Learning (ML) algorithms are the most

[1] https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/.

[2] easyrec.org/, lenskit.org/, www.librec.net.

efficient in providing recommendations, it is even more difficult for non RS experts to utilize such algorithms in real applications because of their high complexity, especially in physical stores where Ambient Intelligence is required. In addition, due to the substantial number of ML algorithms and their variations proposed in the literature, a clear classification scheme for them does not exist [9], making it even more difficult for practitioners to select the ML algorithm that best fits their needs when developing a RS. Even researchers may find it challenging to track how ML algorithms in RSs are used [9]. Further, proof that expertise in recommendation technologies is needed to effectively implement RSs in retail businesses, is that many companies (vendors) exist that offer commercial proprietary RSs[3]. Such companies develop and deploy RSs in their business clients' websites or e-stores, aiming to increase their sales [10].

In this paper, we apply the Model Driven Development (MDD) paradigm on the physical commerce recommendation domain and propose a novel UbiCARS (Ubiquitous Context-Aware Recommender Systems) MDD Framework that aims: (i) to reduce complexity, abstract the technical details and expedite the development and application of State-of-the-Art context-aware RS in ubiquitous environments (physical retail stores) by practitioners that are not recommender system experts (first contribution of this work), and (ii) to enable these practitioners to track ubiquitous user-product interaction in the physical store and use this information in the recommendation process, together with user-product interaction data from the online scenario, a combination which is expected to further improve recommendation accuracy (second contribution of this work). A new graphical Domain Specific Modelling Language (DSML) for UbiCARS is proposed that drives model-based design and dynamic configuration of such systems for physical and online commerce, as well as system integration with pre-existing e-stores. To the best of our knowledge, a DSML for UbiCARS does not exist.

The paper is organized as follows: Sect. 2 provides the research background and related work. Section 3 discusses the proposed UbiCARS methodology, while the UbiCARS framework architecture is described in Sect. 4. In Sect. 5, we demonstrate the modelling process using the proposed UbiCARS MDD framework and discuss a use-case. The paper completes with conclusions and future work in Sect. 6. In the remainder of the paper we will refer to practitioners as *users* of the system and customers as *end-users*.

2 Background

2.1 Recommendation Methods for E-Commerce

Collaborative filtering (CF) constitutes the most adopted method for RS since is relies on user's *behaviour history*, such as previous transactions or ratings of items [11]. It is also independent of any domain. The most efficient CF approaches are the Model based

[3] Google Cloud Machine Learning (ML), SLI Systems Recommender, Azure ML, Amazon Machine Learning, SuggestGrid, Yusp.

that utilize intelligent Machine Learning techniques such as Matrix Factorization. In e-commerce settings, CF is preferable since it relies only on user behaviour and, therefore, explicit profiles for users and/or products are not needed [6]. Hence, users can receive recommendations without firstly being asked to complete a profile, a task users normally do not like. RSs for e-commerce use *explicit user feedback on products* such as product ratings by users to elicit and model user preferences and offer personalized recommendations of products users would enjoy [6, 12]. In case explicit feedback is not available, RSs use *implicit user feedback on products* by *tracking user behaviour* such as user transaction data (purchase history), clickstream data, click-through rate (CTR) and browser history information [13–16]. Problems with explicit techniques are that they require cognitive effort from users, and they interrupt their task at hand as they have to stop and rate items. A comparison between implicit and explicit feedback [13] found that: (i) the more time users spend on a content indicates that the more they like that content; (ii) more user visits on a content mean the user is interested in it; (iii) multiple user accesses to the same content shows user interest in that content. Yahoo proposed using dwell time on content items on the Yahoo home page as an implicit technique to measure how likely a content item is relevant to a particular user, and reported good results [16].

2.2 Recommender Systems in Physical Retail Stores

Research has been conducted on using the knowledge obtained from e-commerce RSs to offer recommendations to customers while being in physical business locations like a shopping mall or a grocery store. The authors in [12] suggest offering recommendations based (among other) on customers' *staying time* in each shopping mall area to recommend which shops customers should visit next. According to the authors, staying time at each selling area is considered as an important piece of information on user preferences in purchasing of goods. Evaluation of their RS performance against real sales data showed that, not only sales history, but also the customers' shopping path data make a RS highly accurate. In [17], a mobile RS for indoor shopping is proposed that uses indoor mobile positioning by using received signal patterns of mobile phones to recommend brand stores to users in a shopping mall. The proposed RS records users' past activities and context, among other the time spent in each store during every shopping process. In [18] a RS that recommends shops in a mall is described. To detect customers' location, RFID devices and related infrastructure have been deployed in a large-scale shopping mall. The work in [5] describes a scenario where the customer is recommended with products in store via a store-owned device called personal shopping assistant (PSA). The recommendations are based on the items that are currently in the cart, on customers' location, and on their purchase history. In [19], a RS for shopping is proposed that estimates user preferences on items based on their physical distance from the items, whether a user picks up an item or scans an object using a RFID reader device. Pfeiffer et al. [20] present a RS that uses eye-tracking to (implicitly) elicit preference information in a minimally intrusive manner in order to reduce users' effort: glasses are used as a more ubiquitous and personal object than smartphones.

2.3 Related Work on CARS Modelling

Context-Aware Recommender Systems - CARSs utilize the context in the recommendation process [11], making it more accurate, but also turning it into a complex (in comparison to un-contextual RSs) multidimensional problem: user × items × context → ratings. In our prior work [21] we have defined the novel concept of UbiCARSs as recommender applications that facilitate users on site through recommendations by using context-aware intelligent recommendation algorithms that operate on multidimensional datasets. In this paper, the concept of UbiCARS is used for enabling the computation of context-aware recommendations on context-aware datasets from both the physical store and the e-store scenario.

A number of works in the literature propose using modelling or other software engineering techniques to tackle RS complexity. A recommendation framework for assisting developers build CARSs and hybrid RSs is Hybreed [8]. Hybreed incorporates a set of standard recommendation algorithms and provides templates for combining them into hybrids with a significantly reduced amount of effort. In [22], we have proposed a context modelling system and learning tool to guide developers through the process of CARS context modelling. In [9], the authors investigate how ML algorithms in RSs are studied and used, as well as trends in ML algorithm research and development. In [23], the authors deal with the problem of web developers needing assistance, by means of proper methods and tools, for dealing with complexity issues in adopting recommendation techniques in their web applications. They explicitly mention the lack of model-driven methodologies for the specification of RS algorithmic and interface elements. The authors use UML to model an un-contextual RS algorithm. However, by modelling the recommendation algorithm itself, the complexity an algorithm can have to be able to be used in the proposed model-driven process is somewhat limited, in the sense that too complex recommendation algorithms will be difficult to be modelled. In [7] a user modelling framework for CARS is proposed to serve as a tool for developers and researchers to build data models for CARS. A CARS model schema and UML class description is provided.

Although the above works have similarities with our work, our proposal differs in five important aspects: (i) it defines a novel Domain Specific Modelling Language for UbiCARS (UbiCARS DSML), (ii) it focuses on the ubiquitous scenario of UbiCARS aiming to enhance product recommendation accuracy in physical commerce, (iii) it supports complex algorithms and data models from the State-of-the-Art of RS literature, (iv) it is easily extendable and (v) it is directly integrable with existing e-stores.

3 UbiCARS Framework

Model-driven Development aims at the abstract representation of the knowledge and activities of a particular application domain, as well as in automation. Application models are defined at an abstracted level and, using automated transformations or interpretations, they are converted into applications, eliminating or minimizing the need to write code. In this paper, we apply the Model-Driven Development paradigm on the physical commerce recommendation domain and propose the *UbiCARS MDD*

framework. We define the *UbiCARS app* and *CARS system* as follows: UbiCARS app is a mobile application that: (i) enables tracking of end-user interaction with products on-location; (ii) displays product recommendations to end-user on-location. CARS is a server-side system (including the recommendation engine) that: (i) enables tracking of end-user interaction with products on-line; (ii) computes context-aware recommendations; (iii) displays product recommendations to end-users through the e-store.

3.1 UbiCARS Methodology

The proposed UbiCARS MDD framework defines a DSML for UbiCARSs for commerce and a corresponding graphical modelling editor. Via the editor, practitioners can use the DSML to drive model-based design and dynamic configuration of UbiCARS in commerce settings, as well as system integration with pre-existing e-stores. In Fig. 1 the multi-layered software architecture of the framework is presented. The UbiCARS DSML acts on the Modelling layer via an editor for practitioners to design their UbiCARS commerce applications. The DSML is cross-platform, meaning that any platform specific implementation details are abstracted from the designers.

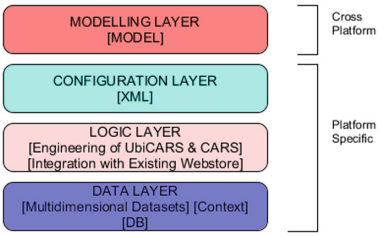

Fig. 1. Multi-layered software architecture.

When UbiCARS design is completed, the framework generates configuration files in eXtensible Markup Language (XML) for the Configuration layer, where a Parser extracts all user defined information and passes it to the logic layer that undertakes the engineering of UbiCARS. The Logic layer builds the necessary data models and database (DB) tables, whereas it also implements the system configurations indicated by the Configuration layer, including system integration with pre-existing e-stores. It also configures the system for the execution of the UbiCARS app. The Data layer prepares the necessary context-aware user-product interaction datasets to be fed to the CARS system. To use the MDD framework to build UbiCARS, a practitioner only needs to use the DSML editor to design the UbiCARS model. Then, automatically, the UbiCARS app and CARS system are configured and deployed.

3.2 Enhancing Recommendation Accuracy

Assuming a retail business with an e-commerce website and physical showrooms with products, our solution proposes that a UbiCARS app is used together with a CARS that operates on the e-store to enhance the overall recommendation efficiency (in [15], it is

shown that users' diverse implicit feedback data can be used to improve recommendation accuracy). User's implicit feedback data acquired within the physical store complement user's explicit and implicit feedback data in the online scenario to enhance end-user preferences modelling and improve the overall recommendations for that user. In addition to e-commerce methods for tracking user behaviour (user purchase history, clickstream data and browser history information), we enable tracking of user's ubiquitous behaviour in real-time while in-store, aiming to acquire the ubiquitous user-item interaction (Fig. 2). Specifically, similarly to the dwell time used in online settings as user implicit feedback [14, 16], we propose utilizing the *"staying time in front of a product"* and *"scanning NFC tag of a product"*.

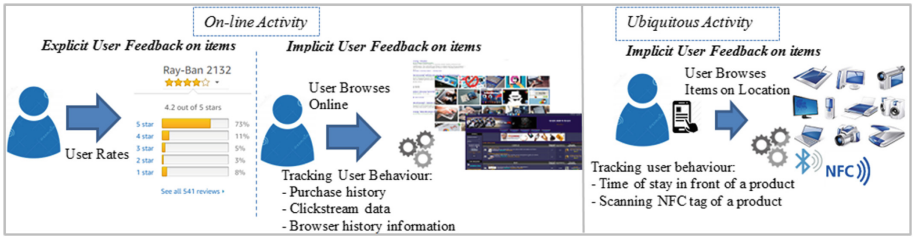

Fig. 2. Modelling user preferences.

A UbiCARS Scenario: RSs are most helpful in large shops with many product categories where the customer base is likely to be very heterogeneous [5]. We consider an electronics store with a large variety of electronic devices and peripherals that each requires a level of knowledge from customers in order to purchase the best suited product for their needs and preferences. The store showroom is equipped with Bluetooth beacons placed on products to track end-user interaction with products via a UbiCARS mobile app. While visiting a showroom, the user is not sure whether the products she is currently looking at are the best for her to purchase in terms of meeting her preferences. The store showroom cannot possibly host all the products that are offered online. End-user needs assistance to narrow down her options to a product or a selection of products that are most suitable for her. At this point the end-user can use recommendations of products.

Nicolas is a regular user of the Public e-store; he has purchased a few products in the past and also rated some of them online. When Nicolas uses the Public e-store, he likes the personalized product recommendations the website offers. Most of the times, the recommended products match his preferences so he checks them out. Nicolas is now at a Public physical store showroom browsing the products. He is quite interested in technology products and especially likes high end laptops. Public includes in its showroom only a small portion of the products that are offered in total by the store. Nicolas approaches a laptop on a shelf that he finds interesting. He reads a few specs from the small label on the laptop; he would like to know more information on its features, as well as receive product recommendations. Nicolas opens the Public UbiCARS app on his smartphone. The app receives signals from the Bluetooth beacons

and sends related info to the server which (i) identifies the product Nicolas is in front of and (ii) computes Nicolas "staying time in front of the product". The Public UbiCARS app recommends to Nicolas three products which he checks out: the two are located in the same showroom, while the third can only be found online, so the app suggests purchasing it online. Nicolas finds it easy and playful to receive recommendations in store so he repeats the process for more products.

For Nicolas the e-store has recorded a number of ratings on products (explicit feedback) and has tracked his online interaction with products: browser history and online purchase history (implicit feedback). From such data, datasets are compiled to be used in the recommendation process (Fig. 3). Browsing history refers to the number of clicks on the product's name or icon in webpages where many products are listed or recommended, as well as the number of accesses in the product's webpage, while purchase history refers to number of purchases of the corresponding product.

The UbiCARS app contributes to the recommendation process by sensing the time (in seconds) Nicolas has stayed in front of a product (e.g. left dataset in Fig. 4). After Nicolas visit to the store, the UbiCARS app dataset will be available to the CARS for use in the next recommendation computation, in addition to those in Fig. 3. Recommendations for Nicolas the next time he will visit one of the Public physical stores or the Public e-store can be more accurate, since more relevant information will be used during their computation.

```
userid,itemid,rating,Day,Time
1,15,4,Weekday,Evening
1,24,5,Weekend,Late_night
1,25,2,Weekday,Morning
```

```
userid,itemid,AccessedOnline,Day,Time
1,24,1,Weekend,Morning
1,26,1,Weekday,Evening
1,25,1,Weekday,Evening
```

```
userid,itemid,purchasedOnline,Day,Time
1,24,1,Weekday,Afternoon
1,28,1,Weekday,Afternoon
1,15,1,Weekend,Afternoon
```

Fig. 3. Datasets: ratings; browsing history; purchased history.

```
userid,itemid,StayedInFrontOf,Day,Time
1,25,231,Weekday,Afternoon
1,26,38,Weekday,Afternoon
1,22,178,Weekday,Afternoon
```

```
userid,itemid,StayedInFrontOf,Day,Time,context
13,25,231,Weekday,Afternoon,ElectronicsSection:Laptops
2,26,38,Weekday,Afternoon,ElectronicsSection:Consumables
1,22,178,Weekday,Afternoon,ElectronicsSection:AppleProducts
```

Fig. 4. User's staying time in front of a product datasets.

3.3 The UbiCARS DSML

We have implemented the UbiCARS DSML to offer the highest possible level of abstraction from technical details for the ubiquitous product recommendation domain, as well as to simplify usage and instantiation by practitioners. The DSML was designed as an Ecore metamodel in Sirius[4], an open-source Eclipse project which allows leveraging the Eclipse Modelling technologies (EMF, GMF) to create custom graphical modelling workbenches. Figure 5 presents part of the Modelling Language (due to space limitations the full metamodel is referenced[5]).

[4] http://www.eclipse.org/sirius/.

[5] https://drive.google.com/file/d/1Dk7XgdCLusH0_AL7Aw-n8_FMupGI7KZY/view?usp=sharing.

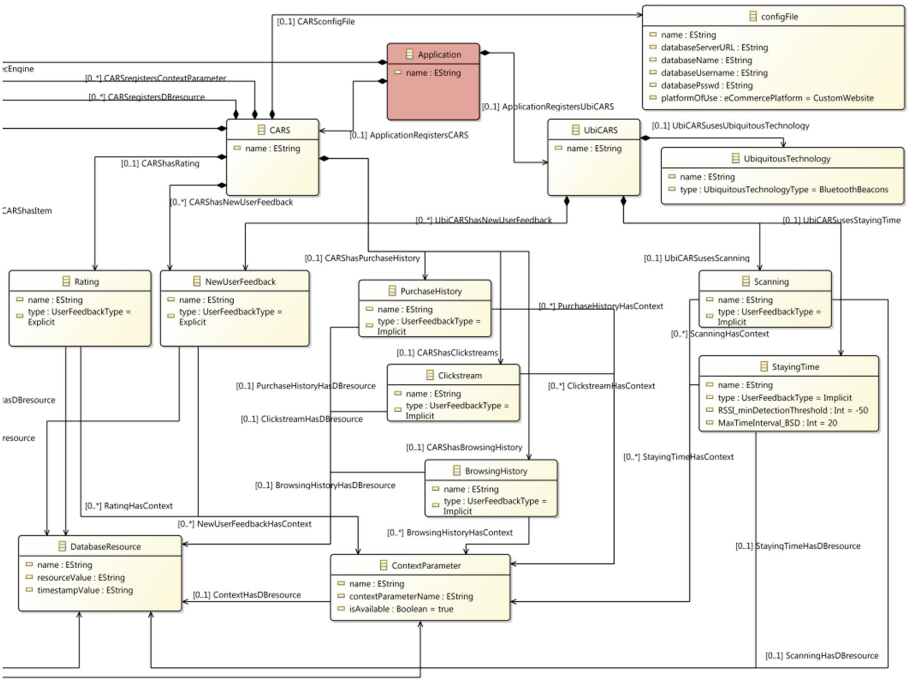

Fig. 5. The proposed UbiCARS DSML.

The `Application` element is the core element that represents a product recommendation system consisting of a `CARS` system and a `UbiCARS` app. `CARS` defines the user `Ratings` on products as an explicit user feedback element, and the `PurchaseHistory`, `ClickStream` and `BrowsingHistory` as those representing the user implicit feedback on products. A `CARS` can instantiate zero or one of these elements, while a number of custom `NewUserFeedback` elements can be defined by the user which can be explicit or implicit (default value is explicit).

In the physical commerce (ubiquitous) scenario, a `UbiCARS` app uses two implicit user feedback elements, the `StayingTime` element representing the staying time in front of products and the `Scanning` element representing the scanning of NFC tags of products. `UbiCARS` has zero or one of these elements, as well as a number of custom explicit or implicit `NewUserFeedback` elements to be defined by the user if needed.

Each of the aforementioned elements uses a `DatabaseResource` and may use a `ContextParameter`. A `DatabaseResource` specifies information about the database resource, where the respective information will be stored and retrieved from (timestamp may be also used as contextual information about time, defining thus the exact time the user interaction on the product took place). A `ContextParameter` captures the context of end-users while interacting with products. For instance, user location in terms of GPS coordinates or any custom defined attributes, such as “store electronics section” or “basement”, can be defined. Context is assigned with a Boolean `isAvailable` to denote whether the corresponding context sensing mechanism is

available or needs to be developed. A `ContextParameter` uses a `DatabaseResource` to denote where the corresponding context will be/is already stored.

`RecommendationEngine` (See footnote 5) computes the recommendations. The CARSKIT recommendation framework (Sect. 4) is the default engine in the metamodel; however, other recommendation frameworks can be used. The engine uses a `RecommendationAlgorithm` which the user can select from the available algorithms offered by the selected engine. Provided that the user has selected CARSKIT to be the recommendation engine, the default algorithm used is context-aware matrix factorization `CAMF_CU`, while two more recommendation algorithms are available in the metamodel: `CAMF_ICS` and `CPTF` (Tensor Factorization) [24]. The metamodel can be easily extended with more algorithms the CARSKIT framework offers. In addition to algorithm selection, related algorithmic configuration parameters can also be defined.

While `RecommendationStorage` defines the place where computed recommendations are stored, `RecommendationPresentation` denotes the `platformOfPresentation` of the recommendations to end-users – whether it is through the e-store (via a Webpage) or through a smartphone screen (via UbiCARS app); the `visualizationFormat` of the recommendations – whether the recommendations will appear as a main or minor object in the screen, or as a widget (e.g. in Wordpress); and whether explanation of recommendations will be enabled. Recommendation explanations contribute to system transparency and trust, e.g. Amazon.com uses *"Customers Who Bought This Item Also Bought"*. Although providing accurate explanations when complex algorithms are used is a difficult task, a practitioner can provide a simplistic, generic version of explanations, such as *"These products are suggested to you based on your previous transactions, product ratings and interaction with products in our showroom"*. *topN* denotes the number of recommendations to be presented (e.g. top-5). It is used by algorithms that solve the top-N recommendation problem, as opposed to those that solve the prediction problem.

UbiCARS Specific Elements. The ubiquitous technology to be used by the UbiCARS app is determined by the `UbiquitousTechnology` element. The two supported technologies are `BluetoothBeacons` and `NFCScanning`. Other technologies could be used as well such as Wi-Fi[6], or more innovative ones used for indoor positioning such as *smart floors* [25] (although some investment will be required).

Bluetooth beacons enable other Bluetooth devices in close proximity to perform actions, indoor positioning among other. Depending on beacon type, their range may vary from 7 m to a few hundreds of meters[7]. Smartphone software can find its relative location to a Bluetooth Beacon in a retail store - retail stores already use beacons for mobile commerce as beacons can create a more engaging in-store experience for customers. Beacons have also been used for providing users with recommendations[8]. An important point is that during indoor positioning, location tracking is done by the smartphone software and not by the Bluetooth beacon, ensuring thus user privacy.

---

[6] lifehacker.com/how-retail-stores-track-you-using-your-smartphone-and-827512308.

[7] estimote.com/products/.

[8] www.huffingtonpost.com/kenny-kline/how-bluetooth-beacons-wil_b_8982720.html.

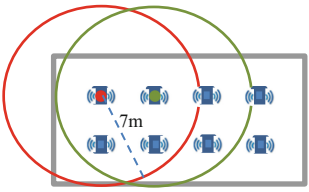

Fig. 6. Signal overlapping example: bluetooth beacons placed on products in a showroom.

Since products in showrooms are usually positioned relatively close to each other, by placing a Bluetooth beacon on each one of them, overlapping of signal coverage occurs (Fig. 6). Hence, customer's staying time in front of a product is calculated by using the Received Signal Strength Indicator (RSSI) of a Bluetooth enabled client (end-user smartphone) when Bluetooth beacons on products in the proximity are being sensed. No user action is needed other than installing the app and running it. The RSSI acts as a distance indicator between the device and the beacon, as devices closer to beacons register higher RSSI values. Devices scan for beacons continuously, taking about 10–15 s between two successive scanning cycles. For each end-user, based on traces of sensed beacons and corresponding received RSSIs in the available space/showroom (see Fig. 4), the system calculates an approximation of the distance from the end-user's device to each Beacon and is thus able to identify the products the user has been staying in front of and for how long. Through the `RSSI_minDetectionThreshold` parameter of the `StayingTime` element, practitioners can specify the minimum RSSI value that needs to be sensed by the end-user's device for the end-user to be considered that she is close enough to the product and therefore, she "has stayed in front of it". After lab experimentation where we have received RSSI values from −30 (corresponding approximately to 2 m) down to −120 (corresponding to approximately 8 m), we have specified the default value for the `RSSI_minDetectionThreshold` to be −50. Note that sensing RSSI values depends on the sensors used and room specifics, and is also sensitive to obstacles: therefore, experimentation and adjustments will be needed in each specific use case scenario. The `MaxTimeInterval_BSD` parameter specifies the maximum time interval between successive detections of an end-user in front of a product in order for these detections to be considered in the same "staying time session" in front of that product. Default value is 20, meaning that successive detections of an end-user device in front of a product beacon when less than 20 s apply between each detection result in the system adding the total time and attributing it to the same session.

Using the above default values, the staying time reasoning algorithm functions as follows (Fig. 7): when a device senses the same beacon for more than once and each successive sensing has a time difference of less than 20 s with the previous one, and furthermore at all times the RSSI value is higher than −50, then the system adds the corresponding time difference between the first and the last sensing and assigns the sum as the staying time of the end-user for that product. The above algorithm ensures that, in case an end-user passes by a product and walks away, no staying time will be assigned to her for the corresponding product.

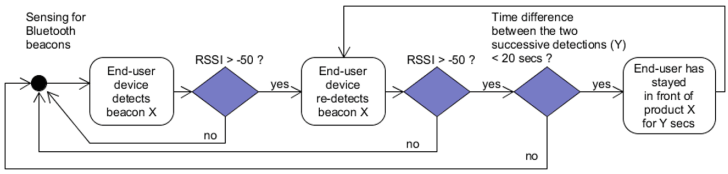

Fig. 7. Staying time reasoning algorithm.

Users can be motivated to scan an NFC tag of a product to find out more information on that product, visit its webpage, be recommended of similar products and read reviews [1, 5, 12, 18–20]. NFC technology requires the user to hold the mobile device in a 5 cm range from the product for an average of 3.3 s for detecting the product [20]. User's staying time in front of a product is a truly implicit user feedback since it happens without requiring user's active participation and without interfering with user's task (user's consent needs to be granted at a prior stage). Having users scan NFC tags of products, however, needs user participation, similar to user ratings, whereas it also interrupts users from their task. When a scan occurs, we can be certain that the user has shown some interest on the product, but not that she actually likes it. The same applies for staying in front of a product: the user may as well be looking at something else other than the product in front of her. Nevertheless, implicit feedback is important and can replace explicit techniques where the latter cannot be used.

4 UbiCARS Framework Architecture

The UbiCARS MDD Framework enriches an existing e-store with context-aware recommendations. The framework architecture is shown in Fig. 8. The online store frontend (client side) tracks user behaviour through the browser software and/or 3rd party software running on the browser. Server side tracking of user behaviour is

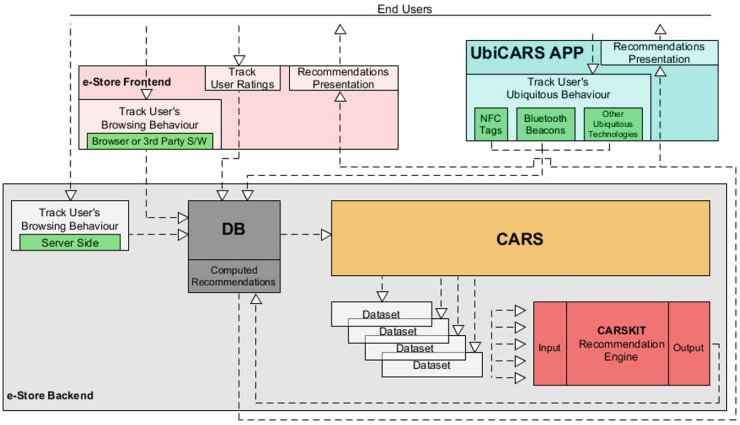

Fig. 8. Framework architecture.

accomplished by analysing server log files. *CARSKIT* [24] is an open-source Context-aware Recommendation Engine that offers State-of-the-Art recommendation algorithms. CARSKIT has been selected among other recommendation frameworks due to the many efficient recommendation algorithms it offers, its ease of use, and the flexibility with which it can be fine-tuned to work with multidimensional datasets.

System implementation considers two of the most well-known, open source e-commerce platforms, WordPress WooCommerce[9] and Drupal Commerce[10]. Practitioners may specify their platform of choice in the model to drive system configuration (`platformOfUse` of `configFile` element). The system provides code snippets and functional modules that can be installed in the corresponding platform and provide the following functionality: retrieve explicit user feedback in terms of end-user ratings on products, retrieve implicit user feedback in terms of end-user purchase history and browsing history (product webpages accessed) on the website, as well as display computed recommendations accordingly, as specified in the model. Integration with the UbiCARS mobile app is provided. Custom e-store configuration is also considered. Note that the cold start problem[11] also applies here. In this sense, after system configuration, the CARS system is not able to produce recommendations until customers have interacted with the electronic and physical stores. In case customer interaction pre-exists in a platform before system configuration, the system integrates and uses it, if relevant. For instance, in case customers have already rated products in WooCommerce before the system takes effect, the latter will consider the data during configuration and use it during the recommendation process.

5 UbiCARS Instance Model and Demonstrator

The UbiCARS MDD framework provides practitioners with a toolset to model, dynamically generate and configure a UbiCARS app and a CARS, as these were described in Sect. 3. The toolset is comprised of a DSML along with its modelling editor. Figure 9 shows the editor, its toolbox and a part of a model.

Through the properties view (right part of Fig. 9), information can be added/edited about the selected element. The model displayed in Fig. 9 defines the `Public UbiCARS` app that uses `BluetoothBeacons` and a `STAYING TIME` element that will track customer-product interaction in the physical store and engineer it into a dataset for the CARS to use in the recommendation process. The `Staying Time Context` adds `Indoor Location` as a contextual element to each customer-product interaction (e.g. the store department this interaction takes place). The right part of Fig. 4 shows a snippet of a resulting dataset for staying time with context. Note that datasets where implicit user feedback data (here the `StayedInFronfOf`) are not scaled as numbers from 1 to 5, need to be normalized before used in the recommendation engine, e.g. the

[9] https://wordpress.org/plugins/woocommerce/installation/.

[10] https://drupalcommerce.org/.

[11] Cold start is an inherent problem of RSs: to be able to produce meaningful recommendations, end-user interaction with products first needs to take place.

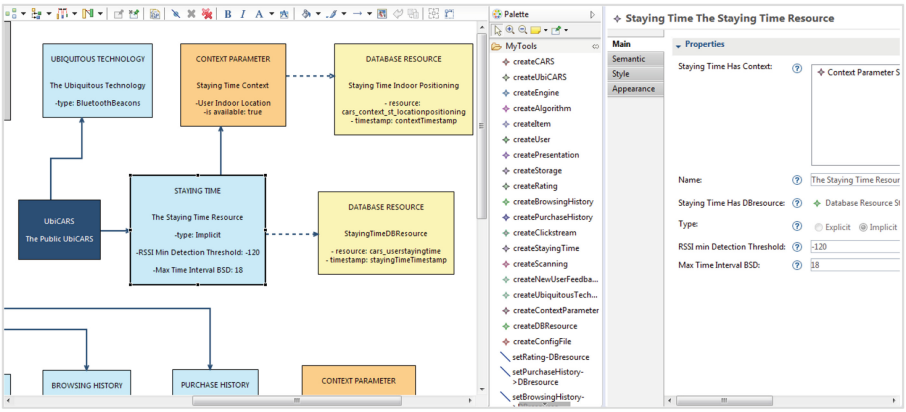

Fig. 9. The UbiCARS instance model & modelling editor.

number of seconds can be scaled from 1 to 5 indicating 5 levels: 1 corresponding to minimum staying time and 5 to maximum (these scales need to be defined by the practitioner depending on the use case scenario). The UbiCARS app provided is extendable and configurable. The system configures it to the level of communicating with Bluetooth beacons to determine the product ID and access the DB to store the data. A similar functionality is defined for scanning NFC tags. Next, context-aware multi-dimensional datasets are compiled by the system and used by the CARS.

For testing purposes we have designed and applied a number of models. Due to space limitations, a model example is referenced[12]. We have set up one example for each e-store platform (WooCommerce and Drupal Commerce), in which we have included electronic products, as these can be found in a real e-store. The datasets used and displayed in this work are produced via interaction of lab personnel with the e-stores. This interaction meant to simulate regular user activity on an e-store, such as browsing products, purchasing products and rating them. To simulate a physical store showroom, we have used a similar layout to the one in Fig. 6 to represent a mobile smartphone showroom. The products were Bluetooth enabled Android and IoS smartphones "for sale", eliminating thus, in this case the need for Bluetooth beacons. Beacons are expected to be more accurate than smartphones, however, when Bluetooth is embedded in the products for sale, budget (for beacons) can be saved.

Utilizing the data from online and physical end-user interaction with products, the system was able to produce 7 datasets: ratings, purchasing history, browsing history, staying time, NFC scanning (when models specify NFC instead of Bluetooth beacons), and two datasets for custom user feedback, one explicit and one implicit. Regarding custom end-user interaction, practitioners may specify the feedback data according to their needs, but the scoring scale still needs to be a number from 1 to 5. For instance, such feedback could be the amount of times a customer has walked passed a product: while the acquired end-user feedback can be a large number, e.g. 112, the dataset needs

[12] https://drive.google.com/file/d/13mXmGcCeImbpJ5G0oxP4Yj1z1bZiviyd/view?usp=sharing.

Fig. 10. Dataset of customers walking passed a product.

to be normalized as shown in Fig. 10. With the generated datasets and the automatically configured e-store, the system was able to compute recommendations and display them to end-users. However, due to the limited data available in comparison with real store data, cold start problems were experienced (Sect. 4).

6 Conclusions and Future Work

In this paper, we have presented our work on an MDD approach toward the development of UbiCARS applications for physical commerce. The approach provides a number of facilitators in the development of recommendations for commerce, aiming for faster development time, enhancement of recommendations accuracy and utilization of more efficient context-aware recommendation algorithms in relevance to existing works. While problems exist when using implicit user feedback on products [6], it is nevertheless important and can replace or complement explicit feedback techniques. In this work we have proposed using additional user-product interaction data from the physical store scenario to improve recommendation accuracy: users are expected to be more satisfied by the recommended products.

As future work, similarly to the evaluation process in [8], we plan to engage practitioners to use the UbiCARS MDD framework. Developers that have worked with e-stores will be asked to use the UbiCARS DSML and editor to integrate State-of-the-Art context-aware recommendations into their e-stores. We plan to use performance metrics, i.e. *task success* (how well was the task completed), *time-on-task* (time needed) [8], Lines-Of-Code (LOC) metric to measure effort, and SUS to measure system usability. As opposed to [8] where a stand-alone evaluation without competitors was conducted, we aim to compare our results with other frameworks by having users conduct the same tasks with other open-source recommendation frameworks. We also intend to study privacy aspects of collecting and utilizing user data, as well as mechanisms to inform users and consider their preferences.

References

1. Chen, C.-C., Huang, T.-C., Park, J.J., Yen, N.Y.: Real-time smartphone sensing and recommendations towards context-awareness shopping. Multimed. Syst. **21**(1), 61–72 (2015)
2. Bick, M., Kummer, T.F.: Ambient intelligence and ubiquitous computing. In: Adelsberger, H.H., Kinshuk, Pawlowski, J.M., Sampson, D.G. (eds.) Handbook on Information Technologies for Education and Training. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-74155-8_5
3. Cook, D.J., Augusto, J.C., Jakkula, V.R.: Ambient intelligence: technologies, applications, and opportunities. Pervasive Mob. Comput. **5**, 277–298 (2009)

4. Ricci, F., Rokach, L., Shapira, B., Kantor, P.B.: Recommender Systems Handbook, 1st edn. Springer-, New York (2010)
5. Walter, F.E., Battiston, S., Yildirim, M., Schweitzer, F.: Moving recommender systems from online commerce to retail stores. Inf. Syst. E-Bus Manag. **10**, 367–393 (2012)
6. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Proceedings of the 8th IEEE International Conference on Data Mining, pp. 263–272 (2008)
7. Inzunza, S., Juárez-Ramírez, R., Jiménez, S.: User modeling framework for context-aware recommender systems. In: Rocha, Á., Correia, A.M., Adeli, H., Reis, L.P., Costanzo, S. (eds.) WorldCIST 2017. AISC, vol. 569, pp. 899–908. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56535-4_88
8. Hussein, T., Linder, T., Gaulke, W., Ziegler, J.: Hybreed: a software framework for developing context-aware hybrid recommender systems. User Model. User-Adap. Inter. **24**(1–2), 121–174 (2014)
9. Portugal, I., Alencar, P., Cowan, D.: The use of machine learning algorithms in recommender systems: a systematic review. Expert Syst. Appl. **97**, 205–227 (2018)
10. Aldrich, S.E.: Recommender systems in commercial use. AI Mag. **32**(3), 28–34 (2011)
11. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. (TOIS) **23**, 103–145 (2005)
12. So, W.T., Yada, K.: A framework of recommendation system based on in-store behavior. In: Proceedings of the 4th Multidisciplinary International Social Networks Conference, New York, NY, USA, pp. 33:1–33:4 (2017)
13. Núñez-Valdéz, E.R., Lovelle, J.M.C., Martínez, O.S., García-Díaz, V., de Pablos, P.O., Marín, C.E.M.: Implicit feedback techniques on recommender systems applied to electronic books. Comput. Hum. Behav. **28**(4), 1186–1193 (2012)
14. Peska, L.: Using the context of user feedback in recommender systems. In: Proceedings of the 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, pp. 1–12 (2016)
15. Yang, B., Lee, S., Park, S., Lee, S.: Exploiting various implicit feedback for collaborative filtering. In: Proceedings of the 21st International Conference Companion on World Wide Web, WWW 2012 Companion, ACM, New York, NY, USA, pp. 639–640 (2012)
16. Yi, X., Hong, L., Zhong, E., Liu, N.N., Rajan, S.: Beyond clicks: dwell time for personalization. In: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys 2014, pp. 113–120. ACM, New York (2014)
17. Fang, B., Liao, S., Xu, K., Cheng, H., Zhu, C., Chen, H.: A novel mobile recommender system for indoor shopping. Expert Syst. Appl. **39**(15), 11992–12000 (2012)
18. Jie, C., Dong, W., Canquan, L.: Recommendation system technologies of intelligent large-scale shopping mall. In: Proceedings of 2nd International Conference on Computer Science and Network Technology, Changchun, pp. 1058–1062 (2012)
19. Kawashima, H., Matsushita, T., Satake, S., Imai, M., Shinagawa, Y., Anzai, Y.: PORSCHE: a physical objects recommender system for cell phone users. In: Proceedings of 2nd International Workshop on Personalized Context Modeling and Management for UbiComp applications, California, USA (2006)
20. Pfeiffer, J., Pfeiffer, T., Meißner, M.: Towards attentive in-store recommender systems. In: Iyer, L.S., Power, D.J. (eds.) Reshaping Society through Analytics, Collaboration, and Decision Support. AIS, vol. 18, pp. 161–173. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-11575-7_11
21. Mettouris, C., Papadopoulos, G.A: Ubiquitous recommender systems. Computing **96**(3), 223–257 (2014)

22. Mettouris, C., Papadopoulos, G.A.: CARS context modelling. In: Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems, KICSS 2014, pp. 60–71 (2014)
23. Rojas, G., Domínguez, F., Salvatori, S.: Recommender systems on the web: a model-driven approach. In: Di Noia, T., Buccafurri, F. (eds.) EC-Web 2009. LNCS, vol. 5692, pp. 252–263. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03964-5_24
24. Zheng, Y., Mobasher, B., Burke, R.: CARSKit: a Java-based context-aware recommendation engine. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 1668–1671 (2015)
25. Fu, B., Kirchbuchner, F., von Wilmsdorff, J., Grosse-Puppendahl, T., Braun, A., Kuijper, A.: Indoor localization based on passive electric field sensing. In: Braun, A., Wichert, R., Maña, A. (eds.) AmI 2017. LNCS, vol. 10217, pp. 64–79. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56997-0_5