

DELIVERABLE REPORT

D5.4.2

“Recommendation Services”

collaborative project

MASELTOV

Mobile Assistance for Social Inclusion and Empowerment of Immigrants with Persuasive Learning
Technologies and Social Network Services

Grant Agreement No. 288587 / ICT for Inclusion

project co-funded by the
European Commission

Information Society and Media Directorate-General
Information and Communication Technologies
Seventh Framework Programme (2007-2013)

Due date of deliverable:	30 September, 2014 (month 33)
Actual submission date:	19 October 2014
Start date of project:	Jan 1, 2012
Duration:	39 months

















Work package	WP5 – PERSONALIZATION AND RECOMMENDATION
Task	T5.4 – Recommendation Services
Lead contractor for this deliverable	AIT
Editor	Sofoklis Efremidis
Authors	Sofoklis Efremidis, Iakovos Georgiou, Ioannis Christou
Quality reviewer	Charlie Pearson (PP), Lucas Paletta (JR)

Project co-funded by the European Commission within the Seventh Framework Programme (2007–2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

VERSION HISTORY

version	date	author	reason for modification	status
001	25.07.2014	Sofoklis Efremidis	First draft version	Internal
002	01.09.2014	Sofoklis Efremidis, Iakovos Georgiou	Second draft version incorporating list of recommendations	Internal
003	30.09.2014	Sofoklis Efremidis, Ioannis Chrisotu	Third draft version incorporating section on statistical recommenders to be submitted for internal review	Internal
004	2.10.2014	Sofoklis Efremidis	Final draft version after internal review	Internal
005	19.10.2014	Lucas Paletta	Final version after review	For EC

© MASELTOV - for details see MASELTOV Consortium Agreement

MASELTOV partner			organisation name	country code
01	JR		JOANNEUM RESEARCH FORSCHUNGSGESELLSCHAFT MBH	AT
02	CUR		CURE CENTRUM FUR DIE UNTERSUCHUNG UND REALISIERUNG ENDBENUTZER- ORIENTIERTER INTERAKTIVER SYSTEME	AT
03	AIT		RESEARCH AND EDUCATION LABORATORY IN INFORMATION TECHNOLOGIES	EL
04	UOC		FUNDACIO PER A LA UNIVERSITAT OBERTA DE CATALUNYA	ES
05	OU		THE OPEN UNIVERSITY	UK
06	COV		COVENTRY UNIVERSITY	UK
07	CTU		CESKE VYSOKÉ UCENÍ TECHNICKÉ V PRAZE	CZ
08	FHJ		FH JOANNEUM GESELLSCHAFT M.B.H.	AT
09	TI		TELECOM ITALIA S.p.A	IT
10	FLU		FLUIDTIME DATA SERVICES GMBH	AT
11	BUS		BUSUU ONLINE S.L	ES
12	BUS_UK		BUSUU ONLINE Ltd.	UK
13	FUN		FUNDACION DESARROLLO SOSTENIDO	ES
14	DAN		VEREIN DANAIDA	AT
15	MRC		THE MIGRANTS' RESOURCE CENTRE	UK
16	PP		PEARSON PUBLISHING	UK
17	ATE		AUSTRIAN INSTITUTE OF TECHNOLOGY	AT

CONTENT

Version History	2
1. Executive Summary	5
2. Overview	6
3. Recommender Subsystem and Personalization.....	9
3.1 Formalism for rules.....	10
3.2 Correlated concepts	12
3.3 Actions (recommendations)	12
3.4 Types of recommendation	12
3.5 Design alternatives.....	13
3.6 Statistical recommenders.....	15
3.7 Rule Management and Maintenance.....	18
3.8 Avoiding Repeated Rules for the Same Users.....	19
3.9 Reading and Managing Recommendations By The Users	20
4. MApp Events and Notifications to the User Profile	20
5. Recommendation Rules	26
6. Recommender Prototype Implementation	29
6.1 Workflow.....	29
6.2 Rules.....	30
6.2.1 User.Location	31
6.2.1.1 Expiration	33
6.2.2 LanguageLearning	33
6.2.2.1 Expiration	34
6.2.2.2 Knowledge Base Structure	34
6.2.3 .InfoSearch	35
6.2.3.1 Expiration	36
6.2.3.2 Knowledge Base Structure	36
6.2.4 POI Entry	37
6.2.4.1 Expiration	38
6.2.5 TextLens	38
6.2.5.1 Expiration	39
6.2.6 Coins.....	39
6.2.6.1 Expiration	39
7. References	40

1. EXECUTIVE SUMMARY

This document is a follow up of Deliverable D5.4.1, which reported on the architecture and design of the recommender subsystem, and the services it offers to the MASELTOV user. The recommender is a background component of the overall MASELTOV platform, which generates personalized recommendations based on the contextual information that is collected from the user actions and the data and preferences the user has declared. The design of the recommender is presented along with its functionality specifications and its interactions with the other components of the MASELTOV platform. This document presents the updates to the recommender since its initial design and prototype implementation that were presented in D5.4.1. The document gives an update of the list of the recommendations that are produced and the corresponding rules that are used by the underlying rule engine.

2. OVERVIEW

This document is a report on the Recommendation System of the overall MASELTOV platform. It provides details on its architecture, its interrelation with the User Profile, as well as its relations and interactions with the rest of the MASELTOV applications and services and corresponding components. It also presents design alternatives and design decisions pursued for its structure and functionalities. Finally, it shows sample recommendation rules that were fed to the recommender and the results produced by it.

In this document we refer to MApp applications and services. Applications provide functionalities to users and are initiated by them. Examples of MApp applications are the TextLens, the Info, the Help Radar, the Serious Game, Navigation etc. Services on the other hand always exist and provide functionalities mainly to MApp applications. For example, the location service provides the current user position and speed to MApp applications that request it. Services typically “run in the background” supporting MApp applications. For conceptual purposes we distinguish between events that are sent by the MApp applications and notifications that are sent by the services, even though their representation is similar. Both capture the user context, which is communicated to the User Profile.

Figure 1 shows the architecture of the Recommender subsystem. The recommender comprises subcomponents which are deployed on the user’s smartphone and the back-end server. The bulk of the work of generating recommendations is performed by the back-end component, whose functionality is based on a rule engine and is driven by a set of rules. The back-end component is fed with the events and notifications that are logged in the back end database by the User Profile and applies the rules for generating recommendations. Rules are external to the recommender and are programmed in their own language. They are defined so as to result in useful recommendations based on the detected user context. Each rule has a precondition and a corresponding action; when the precondition is satisfied then the action is taken. Rule preconditions can be formulated based on user contextual information, which is captured by the logged events and notifications that are communicated to the User Profile and user data and preferences. As a result the generated recommendations (rule actions) can be highly personalized and targeted.

The functionality of the recommender depends on the User Profile, which is a central component of the MASELTOV platform; its details are presented in Deliverable D5.2 “User Profiling and Personalization”. The User Profile either contains user related data and preferences, and logs of user contextual information, which is used by the recommender to produce targeted and personalized recommendations.

Fehler! Verweisquelle konnte nicht gefunden werden. shows a more detailed architecture for the recommender and places it in the context of its interactions with the User Profile. The architecture that is presented in **Fehler! Verweisquelle konnte nicht gefunden werden.** shows the two platforms on which the components of the User Profile and the Recommender system are deployed. The User Profile makes use of a back-end database to store user preferences data and information that is communicated to it, like events coming from MApp applications and notifications coming from MApp services. The stored information can also be used by more advanced recommender systems either rule-based ones or statistical ones.

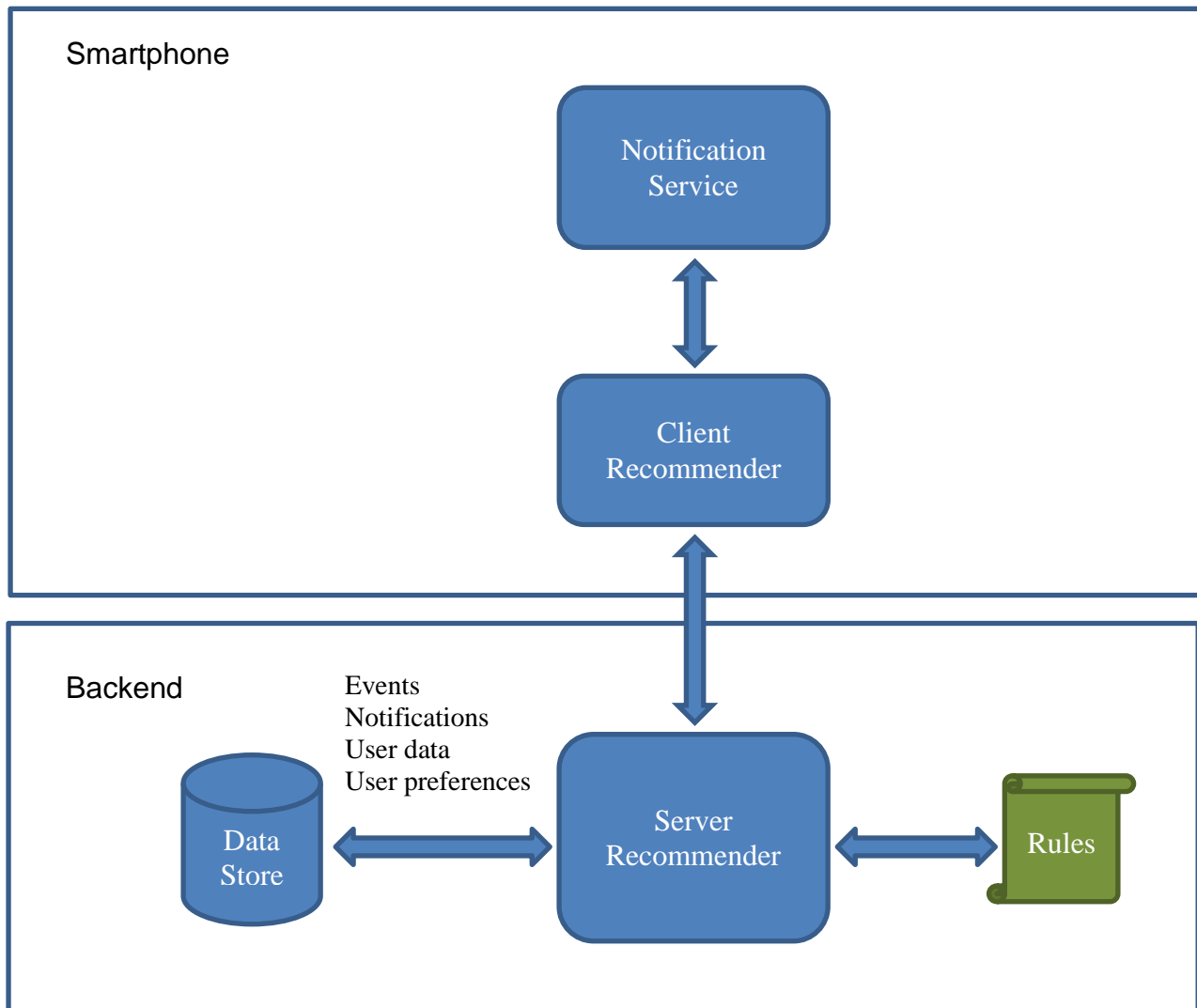


Figure 1: Architecture of the Recommender.

The architecture shows a lightweight rule-based recommender system that runs on the smartphone and a fully-fledged rule-based recommender system that runs on the back-end server. The reasoning behind this approach is to allow the lightweight recommender to provide its services even in the absence of any connectivity, allowing it to issue quick and lightweight recommendations. On the other hand the backend rule based recommender will be able to issue more powerful recommendations by making use of a richer set of rules. The rule scripts that are used by the recommender engine are also maintained in the User Profile even though they are shown separately in order to emphasize their use by the recommender engine. Statistical recommender systems are not part of the objectives of MASELTOV but are shown in the overall architecture as they can provide for powerful future extensions of the MASELTOV platform.

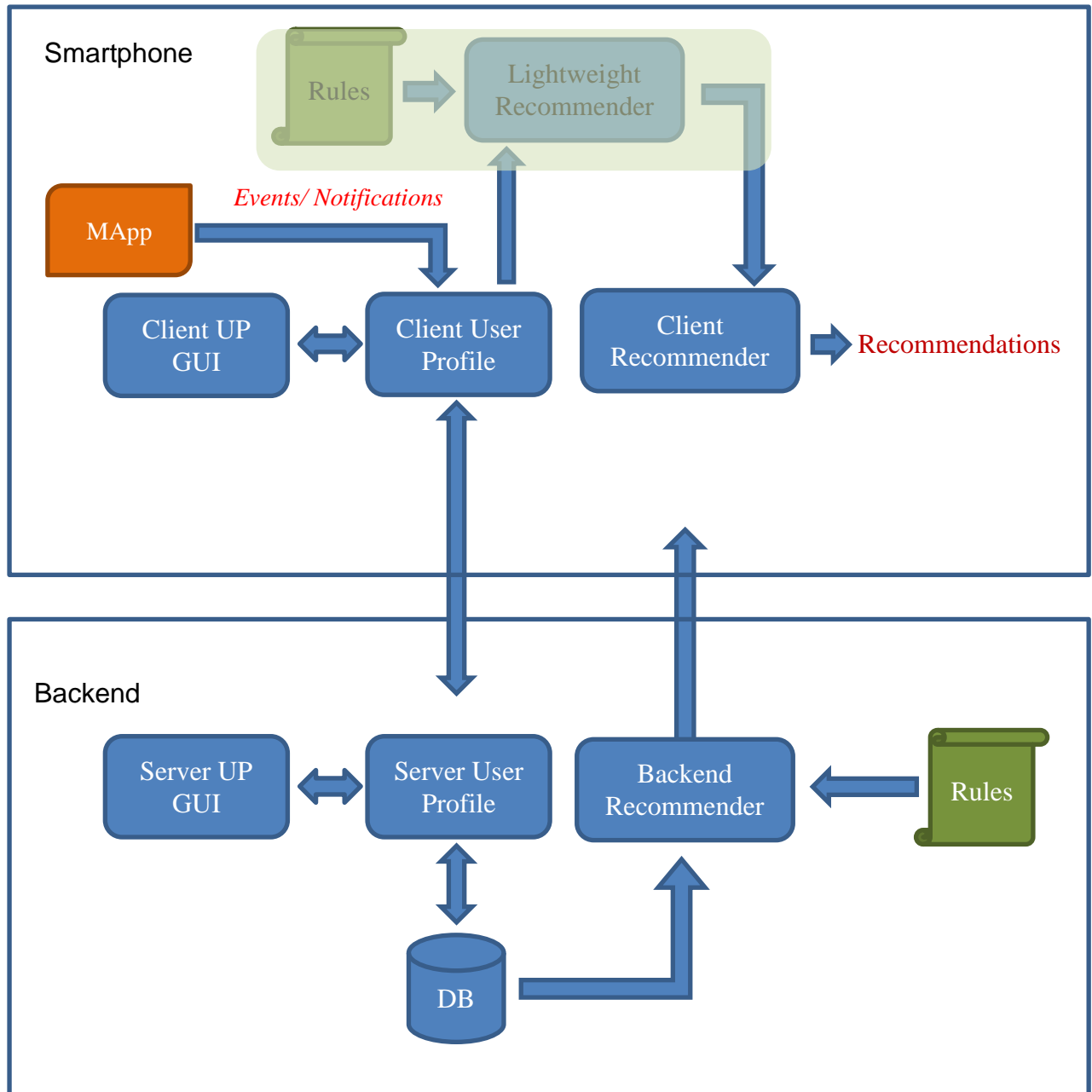


Figure 2: Reference Architecture of the User Profile and the Recommender System.

It should be noted that the architecture of **Fehler! Verweisquelle konnte nicht gefunden werden.** provides hooks for future extensions regarding the lightweight recommender that runs on the client side. As noted before, such technologies are being developed and current implementations are rather unstable or rely on non-standard ways of integrating packages to Android. For example DROOLS on Android relies on AWT classes and needs to be packaged in a non-standard way, breaking compatibility with non-rooted Android devices. Therefore in this document we keep the architecture of **Fehler! Verweisquelle konnte nicht gefunden werden.** as a reference for future implementations and we actually ignore the shaded component of it.

Additional extensions of the back end server of the reference architecture that is shown in **Fehler! Verweisquelle konnte nicht gefunden werden.** are also possible. The architecture shows a back end recommender without hinting towards its nature. For example the recommender can be based on a rule-based system or be a statistical one. The present prototype implementation makes use of a rule based recommender that is built on top of the DROOLS rule engine. Alternatively, statistical recommenders may be used instead. The following section gives more details on the recommender that is used in the prototype implementation and also the different types of statistical recommenders and the algorithms they use.

The following sections present design details of the recommender system, their interfaces with the rest of the MASELTOV components, and the structure of events and notifications that are communicated to them. A list of all events that are collected by other MApp applications, as well as the set of rules that have been implemented is presented.

3. RECOMMENDER SUBSYSTEM AND PERSONALIZATION

The objective of the recommender system is to generate personalized and targeted recommendations based on the events and notifications it receives from the user profile. The function of the recommender is based on the context features that it collects from other MApp applications. The current context comprises features that are either user-induced, or automatic.

The user-induced features are generated as a result of explicit use of MApp applications. They are the interpretation of the selected events that are transmitted to the recommender by other MApp applications. Such events may be relevant to:

- The state of a MApp application
- Input received or sensed by a MApp application
- Keywords encountered by a MApp application.

Automatic features are the interpretation of notifications that are generated continuously as sensed by the various smartphone sensors. They may be relevant to:

- The current user location
- Current environmental conditions (e.g. temperature, and humidity)
- Specialized sensory data (e.g. one produced by the accelerometer).

3.1 FORMALISM FOR RULES

The recommender makes use of a set of rules, which specify actions (recommendations in this case) to be taken when certain conditions are satisfied. Abstractly, the general form of a rule is as follows

$$\text{Predicate} \rightarrow \text{Action} [\text{expiration specification}]?$$

The interpretation of this rule is that when the predicate is satisfied the rule fires and the action is taken. The predicate is a well formed formula that evaluates to true or false and may contain references to events, components of an events, as well as user data and preferences. A specification of the syntax of a predicate is given in the sequel. An example rule is the following:

Predicate:

dist(current location, 37o58'51 12"N, 23o45'15 81"E) < 1Km
&&
User Profile contains keyword "Music"

Action:

Recommend attending tonight's concert at Athens Music Hall

The predicate is a conjunction of two terms, the first one being a relational predicate (less than relation) and the second a primitive function (or method). The predicate is satisfied (and the rule will fire) if the current user location (as communicated to the recommender by the geo-location service) is within 1 km away from the Athens Music Hall and also the user has indicated in their profile that "Music" falls in their interests.

The language of rules that have been implemented as part of the back-end recommender supports four predefined types as follows:

- boolean: its values are *true* and *false*
- integer: its values are all integers that can be represented in the underlying machine
- float: its values are all floating points values that can be represented in the underlying machine
- string: its values are sequences of characters
- coordinates: its values are geographical coordinates expressed as latitude and longitude

The following grammar is a reference specification of the syntax of the rule predicates:

```
pred : | aexpr  
  
aexpr : bterm { '|' bterm }*  
  
bterm : bfactor { '&&' bfactor }*  
  
bfactor : cfactor  
        [[ '=' | '!=' | '>' | '>=' | '<' | '<=' ] cfactor ]?  
  
cfactor : dfactor { [ '+' | '-' ] dfactor }*
```

```
dfactor : efactor { [ '*' | '/' | '%' ] efactor }*

efactor : [ '!' | '-' ] efactor
        | UserProfEntry
        | EventEntry
        | predef '(' expr-list ')'
        | number
        | 'true'
        | 'false'
        | string
        | '[' lat ',' long ']'
        | '(' aexpr ')'

expr-list :
        | expr { ',' expr }*
```

As is evident from the grammar above the standard arithmetic, boolean, and string manipulation operators are supported. Moreover, the grammar allows a number of predefined functions for manipulating User Profile entries, event entries and parts of them, and geographical coordinates.

A User Profile entry may be referenced as `UP.<field>`. For example `UP.gender == "F"` is true iff the user is female. Similarly, `UP.yearsInCountry > 2` is true iff the user has stayed more than two years in the country. To query if a User Profile field has been defined the predefined predicate *UPdefined* can be used. For example `UPdefined("PreferredLanguage")` or `UPdefined(UP.PreferredLanguage)` is true iff a field with the given name has been defined in the User Profile and has an associated value. Similarly, `UPget("PreferredLanguage")` or `UP.PreferredLanguage` returns the value that is associated with the given field name. If additional structure has been defined in the User Profile it can be navigated by listing the fields of the structure, for example `UP.Preferenecs.Games.Adventure`.

For manipulating events, a number of predefined predicates are supported. For an event *e*, `e.getSource()` returns the source of the event, `e.getTimestamp()` returns its timestamp and `e.isDefined(k)` returns true if event *e* contains a definition for key *k*. `e.get(k)` returns the value that is associated with key *k* for event *e*. For example `e.get("learningLevel") > 2` may be used to check the learning level of the user according to the received event.

Geographical coordinates are expressed as `<lat, long>` according to the grammar above. Such values are generated by MApp services, which monitor the current user location, and are transmitted to the User Profile through events.

Recommendations may also carry an expiration specification to avoid overloading the user with recommendations that become stale after some time. Expiration indications, when present, are attached to the recommendation rule itself. For example rules may carry additional annotations as follows:

- `expire <duration>`: recommendation expires after `<duration>` time. For example `expire 1 day` sets the expiration time one day after the current time when the rule fires. Similar specification may be `expire 5 hours`, `expire 1 week`, etc.
- `expire <time>`: recommendation expires at specific `<time>`: For example `expire June 23 18:30` sets the expiration time to the one specified.

- `expire <event-source> (<key>,p)*`: recommendation expires when an event coming from `<event-source>` is received, which contains a key `<key>` and for the corresponding `<value>`, `p(<value>)` is satisfied. As explained below, events carry information about the source component that generates them and a set of `(<key>,<value>)` pairs that encode the necessary contextual information of the event. In this formalism, `p` is a predicate, which evaluates to true or false. For example, `expire learning level (fun x: x >= 3)` specifies that the recommendation to which the specification is attached will expire when an event that has as its source “learning” (probably coming from the MApp learning application) carries a key “level” for which the corresponding value is greater or equal to 3.

Rule actions are taken when a rule fires. An action typically has effect in the environment of the recommender. For the purposes of the MASELTOV recommender, an action is the production of a string that carries the recommendation to the user. Other actions may be the sending of an SMS message, posting a tweet, turning off the mobile device, removing a rule from the ruleset, and so on. The recommendations that are generated are active entities. This means that they may contain references to URIs or other MApp components; clicking on them transfers the user to the corresponding resource (it can be another MApp application).

The recommender system provides flexible and user friendly ways for the handling of recommendations by the user. For example, if a recommendation is not picked by the user, it is flagged as such, so as to be offered again at a later time.

3.2 CORRELATED CONCEPTS

The quality of the recommendations that are issued by the recommender can greatly be enhanced by maintaining lists of correlated concepts, which can be used to trigger more sophisticated recommendations.

Assume that in the context of a healthcare application the recommender maintains links for (hospital, treatment), (hospital, doctor), (healthcare, doctor), ..., etc. When user Mary is looking for a doctor, the recommender links this to healthcare and makes a recommendation to Mary to check the wiki content for the content on healthcare.

3.3 ACTIONS (RECOMMENDATIONS)

A rule specifies the actions to be taken when the predicate is satisfied. As noted above, for the purposes of the MASELTOV recommender, an action is a string that is sent to the user. The recommendation that is specified in the rule can be parameterized, i.e. contain placeholders that refer to elements of the rule predicate. An example of a recommendation text referring to the Learning MApp application may be: “You should now advance to level $\{ \$1 + 1 \}$ of the Italian language course”. $\$1$ is a placeholder which refers to `UP.LearningLevel`, for example, of the rule predicate. So, if `UP.LearningLevel` is currently 2, the outcome will make a recommendation for pursuing level 3. Similarly, references to events and elements of events can be referenced in the recommendation action.

3.4 TYPES OF RECOMMENDATION

A number of recommendations have been implemented in the context of the MASELTOV recommender. The following list contains a sample of the types of recommendation that have

been implemented or even deployed in the actual prototype. The complete list of recommendations, the preconditions and corresponding actions are shown in the chapter 5.

- Recommendation on “relevant nearby places”
- Recommendation to “participate in an event”
- Recommendation to “play the serious Game now!”
- Recommendation to “do some language learning on XX”
- Recommendation to “find relevant information on XX in the INFO”
- Recommendation to “get assistance on XX by using the Help Radar”
- Recommendation to “ask for help in the forum to get advice on XX”
- Recommendation to “go and talk to someone on topic XX”

3.5 DESIGN ALTERNATIVES

As noted above, the recommender makes use of a set of rules for generating targeted recommendations. There are two design alternatives for the recommender, as presented in the sequel. One option is to hard-code the rules in the recommender itself. In this case the resulting recommender is an ad-hoc component that is tied to the specific ruleset it implements. The disadvantage is that a new personalized system like this must be implemented for each user or each group of users with similar profiles. A second option is for the rules to be external to the recommender itself. In this case the recommender can be based on a rule engine that applies the rules to the inputs it receives (i.e. events and notifications from other services) and generates recommendations according to the ruleset specifications.

The overall architecture that is shown in the first section of the document hints for the second option. The advantage of the second option is that rules can be updated and enhanced, different sets of rules can be defined for different groups of users and even specialized to individual users, by employing the same rule engine. Only the rulesets need to be updated, which is a rather straightforward task. The rulesets that are pertinent to a user or user group are maintained along the user profile structure. As hinted above, a ruleset may comprise two parts: a group ruleset, which contains all rules for the group to which the user belongs (based on nationality, or ethnic group) and rules that are specific to the individual user. Both parts can be used to specialize the recommendations to the extent desired.

The implication of having a rule engine to implement parts of the recommender’s functionalities is the existence of a repository that contains a number of rulesets that have been defined by service provider experts for different classes of user. The experts are assumed to have knowledge of the specificities of user groups and based on it, define rules specific to each group. For example, when the recommender receives an event from the Serious Game concerning a restaurant appearing in a background scene, it will recommend a halal restaurant to a Bangladeshi user and a fish restaurant to a Filipino user. It should be noted that these rulesets form the basis for the deterministic (rule based) part of the recommender and have to be specified by experts as mentioned above. On the other hand the architecture of **Fehler! Verweisquelle konnte nicht gefunden werden.** allow for a background statistical recommender as well, which will be able to learn from the behavioural patterns of users, classify the information, and offer recommendations based on observed behaviours.

As explained above, the recommender subsystem has been implemented on top of a rule engine. A number of rule engines are freely available; some of the most advanced implement

the Rete algorithm and are widely used in expert systems. Two prominent implementations of the algorithm are DROOLS a business rule management system, which runs under the JBOSS application server and JESS. DROOLS is a rather heavy-weight implementation as far as the infrastructure requirements are concerned and as such it is not well suited for applications running on mobile devices. Another implementation of the algorithm is the Jess Rule Engine, which has been developed in Sandia Laboratories and has been designed to run on Java. Jess is a lightweight implementation, with a small footprint and fits well into the Java environment which is used in Android smartphones. The rules themselves are expressed in a Lisp-like language but this should not be of concern to the user as it is supposed that rather experienced engineers will develop them and store them in a repository. Currently Jess has not been adapted to run on Android as it depends on JavaBeans components, which are not available on the Android platform. Nevertheless chances are that the port to Android will be accomplished sometime in the near future. For this reason this document maintains the architecture of **Fehler! Verweisquelle konnte nicht gefunden werden.** even though the client lightweight recommender is kept for future extensions when the technology will advance.

The introduction of the ruleset repository implies that the recommendation system should be able to search and download the appropriate ruleset or updates of already downloaded rulesets.

DROOLS implements the Rete algorithm for deciding which rules fire so as to take the corresponding actions. The algorithm works on a number of rules and a sequence of facts that are fed to it. In the case of MASELTOV facts are the events that are generated by the MApp applications. Rules are represented as tree like structures on which traces of the parts of the rule predicates are maintained. When a new fact is received, the algorithm updates its knowledge for the parts of rule predicates. When it detects that a predicate is satisfied based on the facts that have been received, it fires the corresponding rule and takes its action. The algorithm has been documented extensively in the literature ([1], [2], [4], [5]).

A complementary approach to recommender systems makes use of statistical recommenders. These types of recommender do not make use of fixed sets of rules that prescribe what the recommendation to be produced is. Instead they update their knowledge based on detecting behavioral patterns of users, effectively identifying users by their actions and not by their personal data.

The present implementation of the MASELTOV recommender is based on DROOLS. The architecture of **Fehler! Verweisquelle konnte nicht gefunden werden.** allows for future extensions that may also employ statistical recommenders, which may work in parallel with the rule based on of the present implementation. A statistical recommender is based on general patterns that are detected from the behaviour of users. They typically build a model from the past activities of a user or a group of users. They then use the constructed model to issue recommendations. For example based on the user behavioural data it may detect that most female immigrant users from Colombia in the age group 20–25 are inclined towards taking French classes. Therefore, when a user is detected to fall within this group, the recommender will issue a recommendation for taking French classes. The available dataset is compiled from the activities of a group of users and is analysed by considering vectors in multidimensional spaces. In the previous example such a vector is defined across the dimensions (sex, country of origin, age group). By collecting and analysing a large number of

such vectors a statistical recommender may produce recommendations that reflect the tendencies, or what is highly probable to be case (taking French courses in the previous example). A statistical recommender produces recommendations that have an associated level of probability. The accuracy of the statistical recommender depends on the available dataset as well as the algorithms used for analysing it. It is apparent that the functionality of a statistical recommender depends on the availability of a data set (or training set) from which a model is created to be used in subsequent recommendations. It also turns out that the size of the data set affects the quality and accuracy of the produced recommendations; larger data sets will typically result into more accurate recommendations.

The two major approaches for statistical recommenders are based on collaborative filtering and content based filtering. Collaborative filtering systems recommend items to the user based on past ratings of other users. Content-based recommending systems recommended items to the user based on similar items the user has liked in the past. Statistical recommenders are not in the scope of the design and prototype implementation of WP5 but they can be a useful complement to the rule based one for future extensions of the prototype.

3.6 STATISTICAL RECOMMENDERS

This section gives a brief sketch of the statistical recommender engine research landscape (mostly from the view-point of algorithm design), and shows how the best algorithms of their kind can be applied in the context of MASELTOV that uses events to determine what recommendations to show to its users.

Marketing departments have long ago discovered and perfected the use of population segmentation, usually according to demographic criteria, so as to better target their marketing and sales campaigns. For example, after certain market research, product managers would determine that a particular product (or whole product category) appeals better to Asian-American teenagers between 14 and 16 years old whose family income was above a certain threshold. They would then proceed to create advertisements aimed at exactly such a young audience, with the purpose of maximizing next quarter sales.

In the past two decades, however, the advent of computational power, data accumulation about individual consumers' purchases in large relational databases, and algorithmic advances in the fields of machine learning and data mining have allowed far more sophisticated and personalized approaches to materialize. Personalized recommendations are now commonplace in many if not most large retail websites (amazon.com etc.), or video-on-demand service providers (NetFlix, Hellas OnLine, etc.). The common characteristic of the recommender engines behind all these providers is that they compute completely personalized recommendations to their customers, instead of recommendations based on demographic elements such as age group, ethnicity etc.

Today's recommender engines are able to provide far more accurate recommendations to customers, essentially by adhering to the motto "It's not the color of your skin or your age, but your actions that define you"! Current recommender engines therefore essentially all but abandon demographic-based segmentation (which is too crude to be effective), in favor of using individual customers' transaction histories to understand what each one is most likely to appreciate. There are currently three major recommendation approaches within this domain, which are presented in the following paragraphs. The examples show a number of users and

the past history of purchases they have made. Based on this knowledge recommendations can be issued for additional items to be purchased.

User-based Collaborative Filtering (Shardanand & Maes, 1995). Figure 3 illustrates the basic idea behind user-based Collaborative Filtering (CF). To determine what Anna would be likely to purchase, compare Anna's history of past transactions with that of other customers. Figure out an appropriate set of customers (maybe 10 or 100) whose history of purchases matches the products Anna has purchased to the highest degree, and then, recommend for Anna the products that Anna has not yet purchased, yet appear the most times in those other customers' purchasing histories.

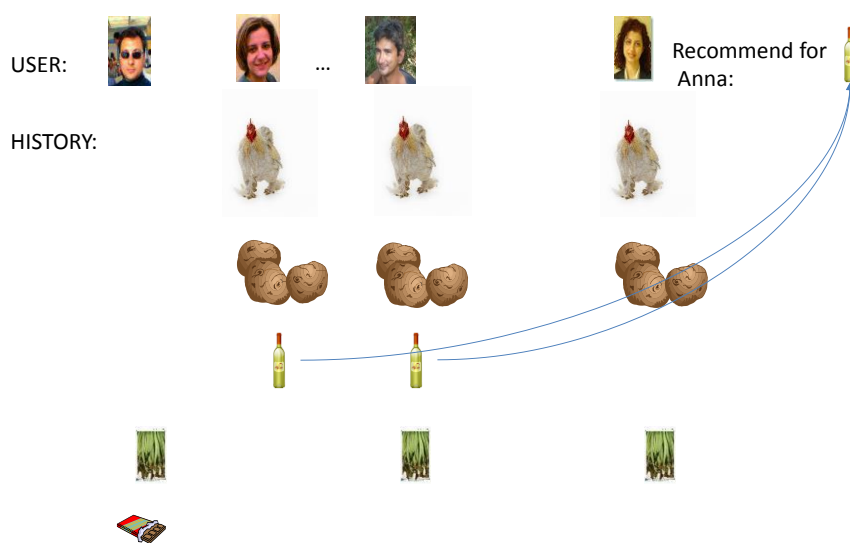


Figure 3: User-Based Collaborative Filtering.

Item-Based Collaborative Filtering (Karypis, 2001). Figure 4 illustrates item-based CF. To determine what Anna would be likely to purchase, figure out the “strength” of the “bond” of each item available for purchase with every other available item, as determined by the number of customers who purchase both those items. Once the, say 10 or 50, “closest” items to each item have been computed, recommend to Anna those items that she hasn't yet purchased that have the highest “bond” to the items that Anna has already purchased. The main advantage of this technique is that it scales better to millions of customers, whereas user-based CF algorithms would require much heavier computational power when faced with millions or tens of millions of customers.

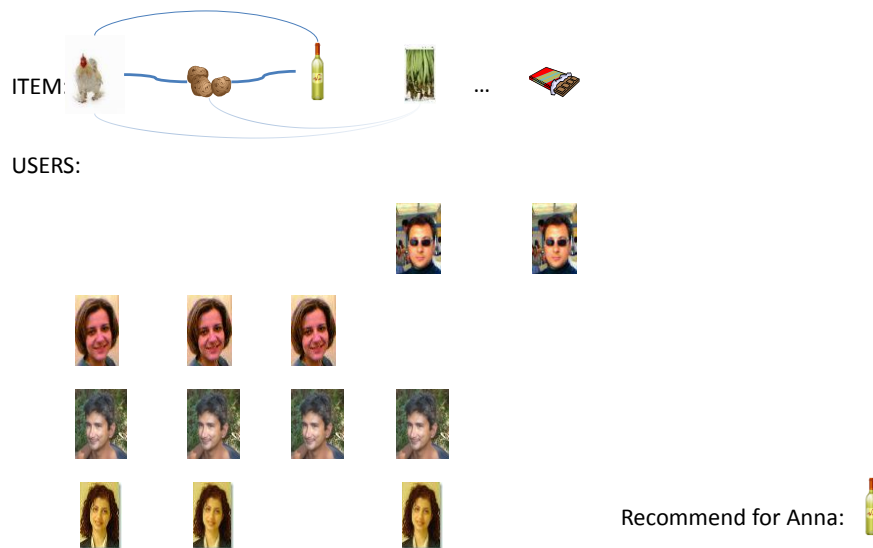


Figure 4: Item-Based CF.

Content-Based Recommenders [10]: Finally, the third major category of recommender approaches, is not based on Collaborative Filtering at all. In this case, the purchase histories of other users is not known to the algorithm when it makes its recommendations to an individual customer. It can therefore be used when, for example, the history of user choices cannot be transferred outside the user's device (e.g. their mobile phone) for a centralized server to process them. Figure 3 illustrates the idea.



Figure 5: Content-Based Recommender.

The recommendation algorithm determines what to recommend to Anna using the meta-data (attributes) of the items Anna has already purchased: the algorithm figures out what attributes appear the most in Anna's purchases (or what attributes appear in groups in Anna's purchases), and recommends to Anna other products that share the same individual attributes (or groups of attributes).

In the context of MASELTOV the choice of the recommender engine to implement requires a way of characterizing an event as an "item" that can be recommended or not to a user.

Moreover, a mapping of <key,value> pairs available in the context of the MASELTOV project, to <userid, event/itemid> pairs, and possibly to <event/itemid, Set<Attribute>> pairs for Content-Based recommendations is necessary.

3.7 RULE MANAGEMENT AND MAINTENANCE

DROOLS is a Business Rules Management System, which makes use of a set of rules that specify the preconditions for actions to be taken. Rules have the form

Predicate → Action

The interpretation is this: if the predicate is satisfied the rule fires and action is taken. The predicate may depend on a number of parameters; a Rule Based System keeps track of the predicate parameters, fires those rules whose predicates turn to true, and takes the corresponding actions.

In the context of MASELTOV, DROOLS is used as the underlying engine for the recommender. The rule predicates encode the user context. The parameters they contain come from two sources:

1. Events that are generated by other MApp applications and actually carry the user context
2. User data and preferences that are declared in the User Profile by the user.

So, when a specific user context is detected (the corresponding predicate turns to true) the rule fires and the recommender takes an action, i.e., produces a recommendation.

An example of a rule is shown in Table 1. In the example the predicate is what appears in the “when” clause and the action is what appear in the “then” clause of the rule. The particular rule makes use of an incoming event, which is referenced as \$e. Out of all possible events that may be fed to the recommender, the specific rule picks those that are generated by the component “User.Location” and the user for whom the event has been generated has a matching hobby with the POI’s category. The rule also shows that variables may be bound to values on the rule predicate and used in the rule action. The formalism is a bit involved but not difficult to follow. When the rule predicate becomes true, a recommendation is created based on the information collected by the matched POI (information temporarily bound to variable \$r).

```
rule "User Preferences and POIs"
  when
    $e: Event( $source : source == "User.Location",
              $r: getUser().checkUserPreferencesWithPOIs( getPoi() ) )
  then
    $e.createRecommendations( $r );
end
```

Table 1: DROOLS rule example.

Rule formulation is a challenging task and at first it may seem that supporting tools are necessary. For example, a GUI that would allow non-technical users to create and maintain sets of rules would be highly desirable and would make the recommender usable to a wider

group of users. In fact such a tool is difficult to construct and the reason is that besides the syntax, rules are very much application-specific. For example, in the above example the method `checkUserPreferencesWithPOIs` makes use of a number of other methods that cannot be described and implemented on the fly by a GUI, unless the GUI duplicates a typical programming language. In particular, method `checkUserPreferencesWithPOIs` uses the geolocation information described in the event and performs an http call to a known URI (at joanneum.at) in order to receive a list of POIs in a certain radius. Then each of these POIs is scanned on their category metadata in order to form a list of category keywords. Finally each one of these category keywords are compared with the user's hobbies as defined by the user under the Preferences in Mapp User Profile.

As a result, rules can only be developed by people with a technical background, like programmers, who, in addition, must implement a number of new methods or modify existing ones, to come up with a new rule. Moreover, in the context of MASELTOV it is not appropriate to have a GUI that will allow administrators to add copies of an existing rule and simply apply a different set of parameters on them.

Nevertheless, it should be noted that GUI tools do actually exist. For example, the DROOLS Guvnor and its replacer Drools Workbench are GUIs that a user can set new rules based on known (predefined) predicate methods and known action methods. These tools allow the user to set a number of parameters to create copies or combinations of rules.

3.8 AVOIDING REPEATED RULES FOR THE SAME USERS

Every time a recommendation is about to be created for a user the engine will also check if the recommendation is going to be a duplication of one that was issued previously.

The rule is that the recommender will not produce the same recommendation multiple times for the same user. An exception to this rule is implemented in order to be able to recreate a rule after the expiration of the previous rule and if it is triggered again by a new event.

For example, if the user receives 2 more coins from a MApp component, and at that moment the user has more than 20 coins, a recommendation will be created urging the user to visit the serious Game and spent these coins. For the next three days (expiration time for the coins) the recommender won't ask for user to redeem coins. However three days later if the user gains more coins from a MApp component (and if s/he still has more than 20 coins) s/he will get another recommendation for redeeming his/her coins.

One can easily envisage additional functionalities that may be incorporated in the recommender. For example, a user may be given the opportunity to classify recommendations as not relevant to their interests, effectively providing feedback to the recommender on what recommendations best fit their preferences. So if a user has set one of his/her hobbies as Music he/she will get a recommendation about the Athens Music Hall every time he/she moves near the Athens Music Hall. If the user marks the specific place as a POI that s/he would not like to get any more recommendations, the recommender can take into account the feedback and refrain from producing similar recommendations in the future.

3.9 READING AND MANAGING RECOMMENDATIONS BY THE USERS

The users can use the Recommendation component available from the MApp Dashboard. Users have a list of all recommendations produced in a descending order grouped by dates as shown in Figure 6.

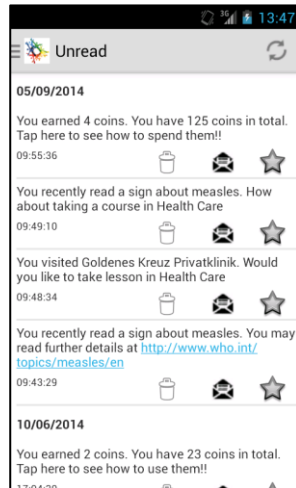


Figure 6: Recommendations grouped by date.

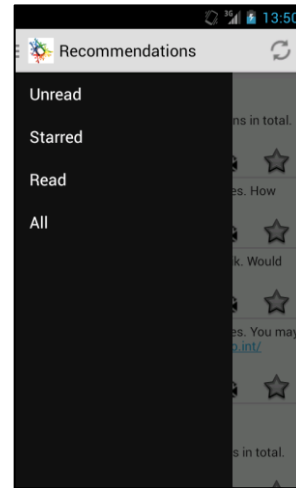


Figure 7: Recommendation filters (drawer menu).

There are three options for each recommendation, to delete the recommendation, mark it as read or unread and mark it as favorite (starred).

The user can delete the recommendation from the list. The result is that the recommendation is marked as deleted but remains invisible on the system. Deleted recommendations will not be displayed in the recommendations list in the future.

Moreover, users can mark a recommendation as read or unread. A drawer menu (as shown in Figure 7) is available in order to filter recommendations by the status of read or unread. Users can switch recommendations' status from read to unread and vice versa as many times as they want.

Finally, users have the option to define as favorite recommendations from the list. From the drawer menu users can also filter recommendations in order to get a list of favorites only. Users can favorite and un-favorite recommendations at will.

4. MAPP EVENTS AND NOTIFICATIONS TO THE USER PROFILE

Events are generated by MApp applications and are sent to the User Profile. The User Profile client application forwards the events to the back-end server for storage further and processing. Events carry user-contextual information, which is used by the recommender to generated personalized recommendations to the user. An event contains three pieces of information:

- Name of the source component
- Timestamp

- Set of <key, value> pairs that carry the actual information

In the context of the User Profile, three categories of event have been distinguished.

- Interesting things: these form the main category of events, which carry contextual information.
- Usage events: these events carry information about the usage of various MApp applications.
- Progress events: these events carry information about the user's progress in the select MApp applications.

The list of all events that can be sent to the User Profile is maintained alongside the prototype implementation. Table 2 shows the complete list of events that are defined in MASELTOV. The table shows the conditions under which each event is issued, as well as the contents the <key, value> pairs that carries with it. For example event eu1 carries a single <key, value> pair, in which the key is the string “duration” and value is the corresponding duration value that is carried with the event. Other events, like the progress and interesting things carry more complex information as shown in the table. It should be noted that event e2 even though defined in the table below is reserved for future extensions of the Language Learning component to trigger the corresponding rule and is not planned to be validated during the final field trial.

Event id	Mapp Component Name	Source	When event is issued	Values of <key, value> pairs
Usage Events				
eu1	Language learning	LanguageLearning	At stop	("duration", <duration>)
eu2	TextLens	TextLens	At stop	("duration", <duration>)
eu3	User Profile	User Profile	At stop	("duration", <duration>)
eu4	Recommendations	Recommendations	At stop	("duration", <duration>)
eu5	Info	fluinfo	At stop	("duration", <duration>)
eu6	Augmented reality	maseltov-arnav	At stop	("duration", <duration>)
eu7	Help Radar	GeoRadar	At stop	("duration", <duration>)
eu8	Navigation	flunav	At stop	("duration", <duration>)
eu9	Places of Interest	flupoi	At stop	("duration", <duration>)
eu10	Serious Game	SeriousGame	At stop	("duration", <duration>)
eu12		Social Forum	At stop	("duration", <duration>)
Progress Events				
ep1				
ep2	Language learning	Language learning	User completes an activity in a language lesson	("publication", <string: publication code>), ("lesson", <string: lesson id>), ("task", <string: task id>), ("progress", <number: percentage>)
ep3	Language learning	Language learning	User completes a test at the end of a language lesson	("publication", <string: publication code>), ("lesson", <string: lesson id>), ("score", <number: percentage>)
ep4	Language learning	Language learning	User rates a statement about their language lesson with a value of 1-5	("publication", <string: publication code>), ("lesson", <string: lesson id>), ("statement", <string: statement id>), ("rating", <number: integer 1..5>)
ep5	Serious Game	SeriousGame.CollectedExceptions	Currency collected	("coins" <number: integer>)
ep6	Serious Game	SeriousGame.ActivityCompleted	User completes an activity in a scenario and receives	("theme" <string: theme name>)

			a journal update	
ep7	Goal Setting	Goal Setting	User within 7 days of deadline	
Interesting Things				
e1	User Profile	GPS tracking	Every 1 minute provided user has moved	("Longitude", <long>), ("Latitude", <lat>)
e2		Language learning	Upon change of learning level (CEFR: A1-A2-B1)	("Language", <lang>), ("Course", <course>), ("Level", <new level>)
e3		Wiki search	Upon pressing of "search" button	("keywords", <keyword list>)
e4	Text Lens	TextLens	Between text detection and text translation (in case user corrects the detected text, event will carry corrected text)	("detectedText", <user corrected text in the image>)
e5		MaseltovContext. ModeOfTransportation	As soon as the mode of transportation changed	("type", <String>), ("confidence", <int>)
e6		MaseltovContext. ActivitySummary	Sent at the end of the day	("total_distance", <long>), ("distance_walking", <long>), ("distance_driving", <long>), ("distance_biking", <long>), ("distance_unknown", <long>), ("time_moving", <long>), ("time_still", <long>), ("time_walking", <long>), ("time_driving", <long>), ("time_biking", <long>), ("time_unknown", <long>)
e7		MaseltovContext. Interests	Sent at the end of the day	("interests", <String keyword list>), ("weights", <Integer weight list>)
e8		MaseltovContext. PlaceEntry	Sent if a person stays at least 5 minutes at one location.	("longitude", <double>), ("latitude", <double>)
e9		MaseltovContext. PlaceExit	Sent if a person leaves a location. Only occurs after MaseltovContext. PlaceEntry	("longitude", <double>), ("latitude", <double>)

e10		MaseltovContext. PlaceHistory	Sent at the end of the day	("coordinates", <Double coordinates list>), ("type", <String types list>), ("ts_entry", <Long ts_entry list>), ("ts_exit", <Long ts_exit list>), ("osm_id", <Integer id list>)
e11		MaseltovContext. PlacesOfInterest	Sent at the end of the day	("coordinates", <Double coordinates list>), ("type", <String>), ("visit_duration", <long duration list>), ("visit_count", <Integer visit_count list>), ("osm_id", <Integer id list>)
e12		MaseltovContext. SocialInteraction	Sent at the end of the day	(Only for illustration purposes in JSON) "contact_data": [{ "contact_id": "int", "calls" : ["call" : { "ts": "long", "duration" : "int", "type" : "String"}, ... }], "messages" : ["message" : { "ts": "long", "type" : "String"}, ...] }, {...}]
e13	Help Radar	GeoRadar. signupVolunteer	Sent when the user signs up himself/herself as volunteer	("username", <String>), ("knowledges", <String>), ("languages", <String>), ("status", <String>)
e14	Help Radar	Georadar. removeVolunteer	Sent when the user remove himself/herself as volunteer	("volunteerUsername", <String>)
e15	Help Radar	Georadar. statusVolunteer	Sent when the volunteer status changed	("volunteerUsername", <String>), ("status", <String>)
e16	Help Radar	GeoRadar. contactVolunteer	Sent when the user contacts a volunteer	("username", <String>), ("volunteerUsername", <String>), ("reqKnowledges", <String>), ("reqLanguage", <String>)
e17	Help Radar	GeoRadar. ratingAssistance	Sent when the user rates an assistance	("username", <String>), ("volunteerUsername", <String>), ("rating", <String>)

e18	Forum	Social Forum.sendPost	Message (i.e. post) posted	("username",<String>)
e19	Forum	Social Forum.sendReply	Replying to a message (i.e. post) another has posted	("username",<String>)
e20	Places of Interest	flupoi.PoiSearch	search keywords	("keyword",<String>)
e21	Info	fluinfo.CategoryTitle	viewed categories	("categoryTitle",<String>)
e22	Info	fluinfo.ArticleTitle	viewed article	("articleTitle",<String>)
e23	Navigation	flunav.RouteStart	start/destination	{ "pointAs": "start", "title": "<title>", "latlng": "<latitude>:<longitude>:WGS84:", "type": "<type>" }
e24	Navigation	flunav.RouteEnd	start/destination	{ "pointAs": "end", "title": "<title>", "latlng": "<latitude>:<longitude>:WGS84:", "type": "<type>" }

Table 2: List of MApp Events.

5. RECOMMENDATION RULES

A number of recommendation rules have been designed with the intention of providing useful and targeted recommendations to the user based on their context and their profile data and preferences. The list is maintained along with the prototype implementation. Table 3 contains the list of all rules that have been defined. Each rule defines the set of events that trigger it as well as the conditions on the information that is carried with the events, i.e., values of <key, value> pairs. Similar to the comments of the previous section, rule r2 is reserved for future extensions of the MApp applications.

Rule Id	Events that trigger rule	Conditions on <key, value> pairs	Conditions on User Profile data	Rule Action	Is Recommendation to be sent as Notification as well?	Expiration
r1	e1			L = getList ofPOIs(e1.<lat>, e1.<long>, radius) S=L.select(poi in L getMetaData(poi) intersect UserProfile.Preferences.Hobbies ≠ {}) for each s in S "The " + s.getName() + " is near you at " s.getAddress()		2 Hours
r2	e2		e2.<new level> > UserProfile.LearningSkills.(e2.<Language>).(e2.<Course>)	Well done for your improvement in + e2.<lang> + " in " + e2.<course> + "(you are now in level " + e2.<new level> + ")! You can now read " +getLearningURL(e2.<lang>, e6.<course>, e2.<new level>)		3 Days
r3	e22			for each kwd in splitToKeywords(e22.<articleTitle>) if kwd in WikiSeachKeywords and resource is aURL "Looking for " + <kwd> + "? Have a look at " + WikiSearchKeywords.getURL(<kwd>)		3 Days
r4	e22			for each kwd in splitToKeywords(e22.<articleTitle>) if kwd in WikiSeachKeywords and resource is a link to another Mapp component "Looking for " + <kwd> + "? Try a Learning Course about " + WikiSearchKeywords.getOpenActivityInfo(<kwd>)		3 Days
r5	e4			"You recently read a sign about " + e4.getText() + ". You may read further details at " + TextLensKeywords.getURL(e4.getText())		15 Days
r6	e4			"You recently read a sign about " + e4.getText() + ". Why don't you take some " + TextLensKeywords.getURL(e4.getLessonText()) + " lessons!		15 Days
r7	e8			U = getUser() L = getList ofPOIs(e8.<lat>, e8.<long>, radius) P = getListOfModules() not in U.completedLessons() S = L.select(p=poi.first in L where P.meta=poi.category)) if S not null then		5 Days

				"You visited " + p.getName() + "! Would you like to take the lesson about " P.getModuleName()		
r8	e1		GPSTracking setting is enabled and UserProfile.Hobbies is empty (no hobbies selected)	"Select your hobbies and interests from User Profile / User Preferences and we can recommend you related Points of Interest"		-
r9	e24					40 Minutes
r10	e16			"You asked the help of a Volunteer for Health issues. You may follow some Learning Language lessons on Health and Care"		2 Days
r11	ep5					2 Days
r12	ep7		UserProfile Goal setting service has a MyGoal with a deadline of 7 days	"You are now seven days from the deadline for one of your personal learning goals. Would you like to update your progress?"	Yes	1 day
r13	ep2	ep2.publication="mas_bas" && ep2.lesson="at_the_supermarket" && ep2.task="quiz:food_and_drink_practice"	UserProfile has "Cookery" selected	When Language Lesson Module: Basics/ Lesson2 At the Supermarket/Activity Food)= completed and User Profile: Entertainment/Hobbies = "Cooking". and THEN trigger recommendation "We see you like cooking and you have completed a language activity on food. Would you like to try out a British recipe? (include link to http://www.bbc.co.uk/food/recipes/search?keywords=&cuisines%5B%5D=british : the BBC recipes website, with a search for all British recipes).	Yes	5 days
r14	ep4	ep4.publication="" && ep4.lesson="" && ep4.statement="conf" && (ep4.rating = "1" "2")		At the end of a lesson, user rates their confidence of using the language more as 1 or 2, suggest that "You rate yourself as NOT being more confident with <Language> . Try looking in the forum for tips on how to increase your confidence."	Yes	1 day

Table 3: List of recommendation rules.

6. RECOMMENDER PROTOTYPE IMPLEMENTATION

This chapter presents the prototype implementation of the recommender. The overall architecture of the recommender, as presented above, contains two components, one that runs on the Android client and one that runs on the back-end server. The overall prototype implementation is based on the DROOLS rule engine, which runs on the JBoss 7.1.1 Application Server and uses the Drools 6.0.0 libraries. A set of rules have been written in order to process the input data and guide the system's output results.

6.1 WORKFLOW

The Recommender runs and receives triggers on an open socket on localhost at port 2346. This trigger triggers the recommender to check the database and get the new events that have been recorded by the User Profile's API. After receiving an event from a MApp application the User Profile component posts the event using the server's API and records the event data in the database. It then triggers the open socket of the recommender in order to inform it that new events are ready to be processed. Figure 8 shows the workflow of this process.

The Recommender checks the new data received from the new event against the data received from previous events and checks the preconditions of the rules it uses. If a rule fires the recommender takes the corresponding action, which produces a recommendation that is sent to the user. A produced recommendation is recorded to the database and is marked as ready to be sent to the user's device as a notification.

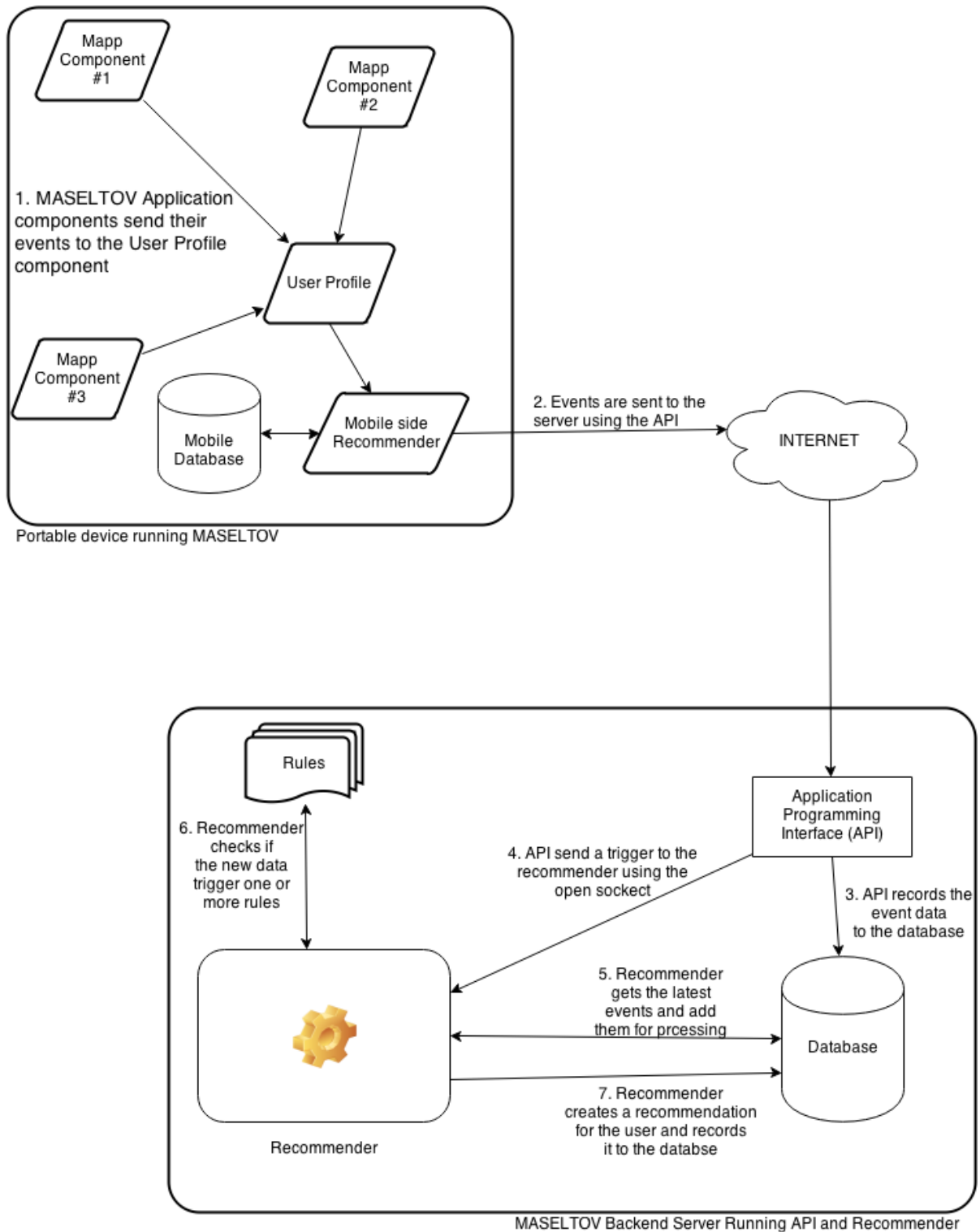


Figure 8: Events workflow; from the device to the recommender.

6.2 RULES

For the purposes of the recommender prototype a number of rules have been implemented. Table 4 is a quick reference to these rules.

Rule	Description
Location	This rule is triggered when the system receives the user's location and requests a number of points of interest (POI) based on the user's location. Then the system tries to match the meta tags of the POIs with the selected user profile's hobbies. If there is a match, a recommendation will be created accordingly.
LanguageLearning	This rule is planned to be triggered when the system receives an event signifying that the user achieved a new learning level in one of the courses. According to the new learning level achieved the system will produce a recommendation to the user. The rule is not planned to be tested during the final field trial.
.InfoSearch	This rule is triggered when the system receives keywords that the user reads an info article on Info component and this must trigger a notification-recommendation about websites that the user might like to search to find related information. It should be noted that no info search functionality is implemented by the project. Nevertheless, this rule is maintained in anticipation of future developments.
POI Entry	This rule is triggered when the system receives an event that a user is detected at a POI for more than five minutes. As soon as the event is triggered the system retrieves the POI's metadata and looks into the list of available lessons provided by the Language Learning component. In the case that a match is found then a recommendation is produced that will urge the user to follow on these lessons.
TextLens	This rule is triggered when the user uses the TextLens component and takes a picture of a text that is translated by the component. TextLens then sends an event with the translated text. If a match is possible between the words of the text and the keywords of available lessons, a recommendation is produced for the user to follow the related lesson in Language Learning component.
Coins	This rule is triggered when a Mapp component adds coins to the user's account and the total available coins of the user exceeds 20 coins. Then a recommendation is produced urging the user to visit the serious Game component and spend/redeem these coins.

Table 4: Quick Reference for Implemented Rules

6.2.1 USER.LOCATION

The recommender makes use of a set of rules, which specify actions (recommendations in this case) to be taken when certain conditions are satisfied. As mentioned above a general form of a rule is:

Predicate → Action [expiration specification]?

The interpretation of this rule is that when the predicate is satisfied the rule fires and the action is taken. An example rule is the following:

Predicate:

dist(current location, 37o58'51 12"N, 23o45'15 81"E) < 0.5Km
&&
User profile contains keyword "Music"

Action:

Recommend attending tonight's concert at Athens Music Hall

This rule will fire if the current user location (as communicated to the recommender by the geo-location service) is within 0.5 km away from the Athens Music Hall and also the user has indicated in their profile that "Music" falls in their interests.

So, this rule is triggered when the system receives the user's location as a set of values latitude, longitude (see example Table 5). The system then queries the POI provider service (run by partner JR) requesting the available POIs based on the user's location and given a radius of 500 meters.

```
{
  "source": "User.Location",
  "timestamp": "20131209100501",
  "info": {
    "lat": "51.516993",
    "long": "-0.126933",
  }
}
```

Table 5: JSON describing a User.Location event

The POIs provider returns a JSON-formatted dataset including a information about the POIs found. We make use of the following information (Table 6):

Key	Description
name	The name of the Point of Interest (e.g., The British Museum)
address	The address of the Point of Interest (e.g., Great Russell Street, London, United Kingdom)
categories	This includes a number of meta data (tag names) describing the Point of Interest (e.g., museum, establishment)

Table 6: Information found in the POI service response

The list of POIs returned by the service is then checked with the user profile's field Entertainment / Hobbies where the user can select his/her interests. If the user has selected, for example, Museums under Art then the POIs having a tag 'museum' (in categories) will be matched and will produce a new recommendation for the user.

```
package com.sample;

import function com.sample.Functions.*;
import org.json.JSONObject;
```



```
rule "User Preferences and POIs"
  when
    $e: Event( $source : source == "User.Location",
              $r: getUser().checkUserPreferencesWithPOIs( getPoi() ) )
  then
    $e.createRecommendations( $r );
  end
```

Table 7: DROOLS rule file for User.Location event

A recommendation matching the metadata “museum” when someone is around the British Museum:

“The British Museum is near you at Great Russell Street, London, United Kingdom”

Or a recommendation matching the meta data ‘movie_theatre’ when someone is around Leicester Square:

“Empire Cinemas Leicester Square is near you at 5-6 Leicester Square, London, United Kingdom”

All recommendations are recorded in the database and they initially marked as WAIT. For these recommendations the system will try to send a notification to the user’s application containing the recommendation text message.

6.2.1.1 EXPIRATION

There are cases where the event might not be received from the device in time (e.g. in cases where the user’s device is not connected to the internet when the MASELTOV component sends the event containing the GPS coordinates).

User.Location events that are received from the server 10 minutes after the events were created on the device generate recommendations but these are automatically marked as EXPIRED in the database and they are not sent as notifications to the user’s device.

6.2.2 LANGUAGELEARNING

This rule is triggered when the system receives an event signifying that the user achieved a new learning level in one of the courses. As noted before, the rule is not planned to be used during the final field trial.

The event has the following JSON format as shown in Table 8. For example if the user achieved level three on reading comprehension (id 36) in English (id 35) the JSON for the event will be:

```
{
  "source": "User. LanguageLearning",
  "timestamp": "20131209100501",
  "info": {
    "language": "35",
    "course": "36",
```

```

    "level": "3",
  }
}

```

Table 8: JSON describing a User.LanguageLearning event.

The ids are available from the content provider for the field id 24 (Learning Level). According to the new learning level achieved the system will produce a recommendation to the user.

```

package com.sample;

import function com.sample.Functions.*;

rule "Learning Level Recommendation"
  when
    $e: Event( $userLanguage: getUser().getLanguage(),
              $source : source == "User.LanguageLevel",
              $l1: getUser().setLearningLevel( getInfoValue("language"),
                                                getInfoValue("course"),
                                                getInfoValue("level") )
            )
  then
    $e.getUser().setLanguageLearningLevel( $l1.getLanguage(),
                                           $l1.getCourse(),
                                           $l1.getLevel()
                                         );
    $e.createUserLearningLevelRecommendations(
      $l1.getLanguageText(
        $l1.getLanguage(),
        $userLanguage
      ),
      $l1.getCourseText(
        $l1.getCourse(),
        $userLanguage
      ),
      $l1.getLevel(),
      $l1.getLinkAndText($userLanguage)
    );
End

```

Table 9: DROOLS rule file for User.LanguageLearning event

This event might produce the following recommendation to the user:

"Well done for your improvement in English in reading (you are now in level 2)! You can now read <http://www.newsnow.co.uk/h/Current+Affairs/Immigration>"

6.2.2.1 EXPIRATION

We have not set an expiration time for this rule.

6.2.2.2 KNOWLEDGE BASE STRUCTURE

For the LanguageLearning rule we use a single database table that holds the appropriate information about the resources that the system will recommend to the user upon a learning level achievement.

languagelearning_suggestions
id : BIGINT(AI, PK)
field_option_id : INT
level : INT
url : VARCHAR(500)
text : VARCHAR(500)

Thus, the administrator can add resources (URLs) that will be recommended to the user for a specific level and for a specific field (in case in the future we need to add another ontology of learning level). A text field is also available to describe the resource and it is currently only used for the administration system (in order to make easier to the administrator to distinguish the resources).

6.2.3 .INFOSEARCH

This rule is triggered when the system receives keywords that the user searched an info component. A number of keywords (a thesaurus) are organized in the database so that the system can understand the topic that the user is looking for. For each grouped set of keywords there might be a link to suggest to the user to follow and find related information. The mechanism searches to find a best match of the keywords in these concepts and provide recommendation links to the user.

For example when the user searches for keyword *dentist* the mechanism will send a recommendation with a URL of a website that includes dentists. The event must follow the JSON format as shown in Table 10.

```
{
  "source": "fluinfo.ArticleTitle",
  "timestamp": "20131209100501",
  "info": {
    "keywords": [
      "dentist",
      "pill",
      "headache"
    ]
  }
}
```

Table 10: JSON example of InfoSearch event

```
package com.sample;

import function com.sample.Functions.*;

rule "Info Search Recommendation"
```

```

when
    $e: Event (
        $source : source == "fluinfo.ArticleTitle",
        $keyword : getInfoValue("articleTitle")
    )
then
    $e. createUserInfoSearchRecommendations ( $e.getInfoSearch() );
end

```

Table 11: DROOLS rule file forInfoSearch event

The search for dentist will result the following recommendation that the user will receive as a notification:

"Looking for dentist? Have a look at <http://www.nhs.uk/Service-Search/Dentist/LocationSearch/3>"

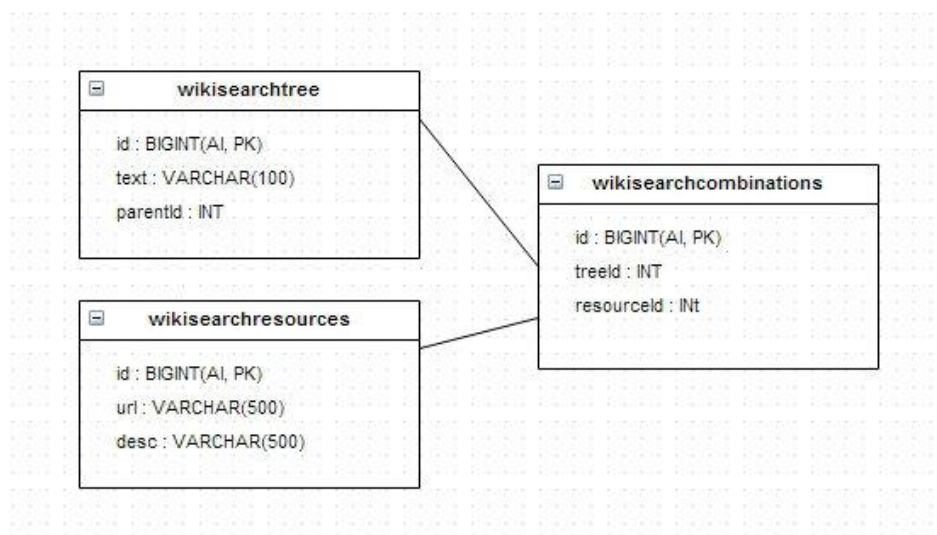
The above link will lead the user to a webpage of NHS about finding dentists service on this browser.

6.2.3.1 EXPIRATION

We have not set an expiration time for this rule, but the rule examines the searched keywords of the last 24 hours.

6.2.3.2 KNOWLEDGE BASE STRUCTURE

For the InfoSearch rule we use the following database structure to describe the structure of the recognized keywords and the recommended resources for each case.



A number of keywords are set by the administrator in a two level tree. The first level describes the group (Health, Sports, Transportation etc.) while the second level consists of keywords that belong to the specific group.

On the other hand there is a number of resources. Each resource consists of a URL and a description. Description is used mainly to help the administrator to spot and distinguish the resource within the backend administration tool. The URL is the actual URL of the resource that will be sent to the user inside the recommendation.

The administrator can create a connection between the keyword from the tree with a resource. More than one resource can be attached to a keyword as well as more than one keyword can be attached to a resource.

6.2.4 POI ENTRY

This rule is implemented in order to help the user when s/he enters a POI and s/he might need help with specialist terminology. For example when detecting that the user enters a hospital, the case might be that s/he needs to describe a health problem and the user might not be aware of some special terms.

This rule is triggered when the system receives the event that the user stayed within a POI for more than five minutes. The recommender then tries to match the POI's metadata with lesson keywords provided by the Language Learning component.

For example, when the user enters a hospital and stays within the building for five minutes, an event is produced and sent to User Profile and finally to the Recommender. The event must follow the JSON format as shown in Table 12.

```
{
  "source": "MaseltovContext.PlaceEntry",
  "timestamp": "20131209100501",
  "info": {
    {"longitude", 14.4170215},
    {"latitude", 50.0765468}
  }
}
```

Table 12: JSON example of POI Entry event.

```
rule "User Enters a POI Recommendation"
when
    $e: Event(
        $source : source ==
        "MaseltovContext.PlaceEntry",
        $longitude : getInfoValue("longitude"),
        $latitude : getInfoValue("latitude")
    )
then
    $e.createUserEnteredAPOIRecommendations( $longitude,
    $latitude );
end
```

Table 13: DROOLS rule file for POI Entry event.

Entering a hospital will result the following recommendation that the user will receive as a notification:

```
"You visited XYZ Hospital. Would you like to take lesson about Health Care?"
```

The above recommendation can be selected by the user and will open the Language Learning component in a list of lessons for Health Care.

6.2.4.1 EXPIRATION

The expiration time for this rule has been set to 5 days.

6.2.5 TEXTLENS

This rule is triggered when the recommender receives the event when the user has got a photo of a sign using TextLens component and translated the text on the sign. The recommender then tries to match the words in the translated text with the keywords on the list of lesson keywords provided by Language Learning component. Upon a match, the TextLens rule (Table 15) triggers and a recommendation is produced for the user.

So, when the user gets a photo of a sign that reads “MEASLES OUTBREAK: PARENTS WARNED”, an event is produced and sent to User Profile and finally to the Recommender. The event must follow the JSON format as shown in Table 14.

```
{
  "source": "MaseltovContext.PlaceEntry",
  "timestamp": "20131209100501",
  "info": {
    {"detectedText", "MEASLES OUTBREAK: PARENTS
    WARNED" }
  }
}
```

Table 14: JSON format for TextLens event.

```
rule "TextLens Recommendation"
when
    $e: Event (
        $source : source == "TextLens",
        $keyword : getInfoValue("keyword")
    )
then
    $e.createUserTextLensRecommendations(
        $e.getTextLens() );
end
```

Table 15: DROOLS rule file for TextLens event.

By reading a sign that is translated by TextLens as “MEASLES OUTBREAK: PARENTS WARNED” the recommender will produce the following recommendation for the user:

“You recently read a sign about measles. You may read further details at
<http://www.who.int/topics/measles/en>”

The above recommendation can be selected by the user and will open the Language Learning component in a list of lessons for Health Care.

6.2.5.1 EXPIRATION

We have set a 2 weeks expiration time for this recommendation.

6.2.6 COINS

This rule is triggered when the system receives the event that the user has gained some coins by using a Mapp component and his/her total coins exceed 20. The rule (shown in Table 17) then triggers and a recommendation is produced for the user.

The event must follow the JSON format as shown in Table 16.

```
{
  "source": "flupoi.RouteSearch",
  "timestamp": "20131209100501",
  "info": {
    {"addCoins", 2 }
  }
}
```

Table 16: JSON format of Coins event.

```
rule "Add Coins Recommendation"
when
    $e: Event (
        $source : source == "addCoins",
        $keyword : getInfoValue("coins"),
        $coins : getUserCoins()>20
    )
then
    $e.createUserCoinsRecommendations( $keyword, $coins
);
end
```

Table 17: DROOLS rule file of Coins event.

By receiving more coins the recommender will produce the following recommendation for the user:

"You earned 4 coins. You have 22 coins in total. Tap here to see how to spend them!! "

The above recommendation can be selected by the user and will open for him/her the serious Game component where the user can redeem coins.

6.2.6.1 EXPIRATION

We have set a 2 days expiration time for this recommendation.

7. REFERENCES

- [1] Forgy, Charles L. On the Efficient Implementation of Production Systems, Ph.D. Thesis, Carnegie-Mellon University, 1979.
- [2] Forgy, Charles L. "Rete: A Fast Algorithm for the Many Pattern/Many Object Match Problem," *Artificial Intelligence*, (19)1, Sept. 1982, pp. 17-37.
- [3] DROOLS, <http://www.jboss.org/drools/>
- [4] JESS, <http://herzberg.ca.sandia.gov/>
- [5] JESS rules, <http://www.jessrules.com/docs/71/rete.html>
- [6] Jena, <http://jena.apache.org/>
- [7] Androjena, <https://code.google.com/p/androjena/>
- [8] A. Aho, M. Lam, R. Sethi, J. Ullman, *Compilers: Principles, Techniques, and Tools*. Pearson Education, Essex, U.K., 2014
- [9] Amolochitis, E., Christou, I.T., Tan, Z.-H. 2014. Implementing a commercial-strength parallel hybrid movie recommendation engine. *IEEE Intelligent Systems*. 29(2):92-96.
- [10] Christou, I.T., Gekas, G. and Kyrikou, A. 2012. A classifier ensemble approach to the TV-viewer profile adaptation problem. *International Journal of Machine Learning and Cybernetics*. 3(4):313-326.
- [11] Karypis, G. 2001. Evaluation of item-based top-n recommendation algorithms. In: *Proc. 10th Conf. on Information & Knowledge Management (CIKM 01)*, pp. 247-254.
- [12] Shadnanand, U., and Maes, P. 1995. Social information filtering: algorithms for automating 'word of mouth'. In: *Proc. Human Factors in Computing Conference (CHI '95)*. Denver, CO.