A CONTEXT-AWARE MOBILE EVENT NOTIFICATION SYSTEM USING THE PUBLISH-SUBSCRIBE MODEL WITH A BUSINESS RULE ENGINE AND LINKED DATA

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF INFORMATICS OF MIDDLE EAST TECHNICAL UNIVERSITY

BY

MELİH GÜRGAH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2014

A CONTEXT-AWARE MOBILE EVENT NOTIFICATION SYSTEM USING THE PUBLISH-SUBSCRIBE MODEL WITH A BUSINESS RULE ENGINE AND LINKED DATA

Submitted by **Melih GÜRGAH** in partial fulfillment of the requirements for the degree of **Master of Science in the Department of Information Systems**, **Middle East Technical University by**,

Prof. Dr. Nazife Baykal ____________________
Director, Informatics Institute

Prof. Dr. Yasemin Yardımcı Çetin ____________________
Head of Department, Information Systems

Assist. Prof. Dr. P. Erhan Eren ____________________
Supervisor, Information Systems, METU

Examining Committee Members

Prof. Dr. Nazife Baykal ____________________
Information Systems, METU

Assist. Prof. Dr. P. Erhan Eren ____________________
Information Systems, METU

Dr. Nail Çadallı ____________________
KAREL A.Ş.

Assoc. Prof. Dr. Altan Koçyiğit ____________________
Information Systems, METU

Assoc. Prof. Dr. Alptekin Temizel ____________________
Work Based Learning, METU

Date: **11/09/2014**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Melih Gürgah

Signature: ___________________

ABSTRACT

A CONTEXT-AWARE MOBILE EVENT NOTIFICATION SYSTEM USING THE PUBLISH-SUBSCRIBE MODEL WITH A BUSINESS RULE ENGINE AND LINKED DATA

Gurgah, Melih
MSc., Department of Information Systems
Supervisor: Assist. Prof. Dr. P. Erhan Eren

September 2014, 58 Pages

Context-awareness has become an important feature of event recommendation and notification systems. So far, several studies in tourism and education domains have provided good results on using different context data and delivering messages based on this context-aware environment. Although many context data are gathered, the analysis of these context data for a proper recommendation still remains insufficient. Even if the recommendation itself is said to be successful, delivery performance, in other words, notifying the message recipient under appropriate conditions, is still inadequate. We propose a publish-subscribe based event notification system enhanced with a business rule engine for context data evaluation, and linked data for semantic analysis. We aim to improve event notification performance by aggregating various context data, making complex inferences and finding the most suitable time to deliver messages for the subscriber by applying the business rule concept. Furthermore, in order to semantically analyze event details and infer new relationships, we utilize semantic analysis by using linked data. To validate our proposed system, we implement a working prototype incorporating event publishers, an event management server composed of a business rule engine, a semantic analysis module powered with linked data and an event dispatcher component, as well as internal and external context sources. The applicability of the system is demonstrated by evaluating it against several sample scenarios.

Keywords: Mobile Computing, Context-Aware Notification System, Publish - Subscribe Model, Business Rule Engine, Linked Data

ÖZ

İŞ KURALI MOTORU VE BAĞLI VERİ İLE YAYINLA-ABONE OL MODELİNİ KULLANAN BAĞLAM BİLİNÇLİ MOBİL ETKİNLİK BİLDİRİM SİSTEMİ

Gürgah, Melih
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. P. Erhan Eren

Eylül 2014, 58 Sayfa

Bağlam bilinçlilik, etkinlik tavsiyesi ve bildirim sistemlerinde önemli bir özellik olmuştur. Şimdiye kadar, turizm ve eğitim alanlarında birçok çalışma farklı bağlam verilerini kullanmada ve bu bağlam bilinçli ortama dayanarak mesajları iletmede iyi sonuçlar sağlamıştır. Birçok bağlam verisi toplanmasına rağmen, bu bağlam verilerinin analizi iyi bir tavsiye için hala yetersiz kalmaktadır. Tavsiyenin kendisi başarılı olarak nitelense bile, iletim performansı, diğer bir deyişle, mesaj alıcısına uygun koşullarda mesaj bildirmek hala yetersiz olmaktadır. Bu noktada, bağlam verileri değerlendirme için iş kuralı motoru ve anlamsal analiz için bağlı veri ile geliştirilmiş yayınla-abone ol tabanlı etkinlik bildirim sistemi sunmaktayız. Buradaki amaç, iş kuralı kavramını kullanarak, çeşitli bağlam verilerini birleştirip ve kompleks çıkarımlar yapıp aboneye mesajları iletmede en uygun zamanı bularak, etkinlik bildirim performansını geliştirmektir. Ayrıca, etkinlik detaylarını anlamsal olarak analiz etmek ve yeni ilişki çıkarımlarında bulunmak için, anlamsal analizden bağlı veri aracılığıyla faydalanırız. Sunulan sistemi doğrulamak amacıyla, etkinlik yayınlayıcısı modülü, etkinlik abonesi modülü; iş kuralı motoru, anlamsal analiz modülü ve etkinlik dağıtıcısı alt modüllerinden oluşan etkinlik yönetim sistemi modüllerinin yanı sıra, içsel ve dışsal bağlam kaynaklarının dahil olduğu çalışan bir prototip geliştirilmiştir. Sistemin uygulanabilirliği, sistemin birkaç örnek senaryo ile değerlendirilmesi ile gösterilmiştir.

Anahtar Kelimeler: Mobil Hesaplama, Bağlam Bilinçli Bildirim Sistemleri, Yayınla - Abone Ol Modeli, İş Kuralı Motoru, Bağlı Veri

This thesis is dedicated to my family.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Recently, technology has provided significant capabilities for effective use of computing power through mobile devices. By means of wireless sensors, easy Internet access from mobile devices, and enhanced mobile operating systems, the systems can receive many types of context information related to the mobile user and the environment. With this utilization, studies in related domains have improved capabilities to propose and provide context-aware systems.

One of the significant context-aware systems is context-aware event recommendation and message delivery systems. These systems generally concentrate on the tourism domain that includes recommendation of places to visit or events to attend. Additionally, some studies focus on the university campus domain in order to increase quality of in-campus information delivery about academic, non-academic events, in-campus café/restaurant offers and in-campus news. Furthermore, some research studies provide successful results in context-aware movie or music recommendation. The context-aware recommendation systems utilize user profile information, user activities, user preferences, location, and weather information as contextual data, and then create new knowledge in order to best describe the user and act accordingly.

With the technical improvements in mobile devices, location finding via GPS and other wireless sensors has become a basic feature of mobile phones. In context-aware event recommendation and message delivery systems, location information is the fundamental context data used for recommending geographically related events. For example, a tourist wants to visit nearest places and closest events, so utilizing location context data carries a very important role in event recommendation. Moreover, time data also have a significant effect on recommendation, so that a tourist or a student will not have unfortunate experiences while arranging event schedules, with the help of time-aware recommendations. Furthermore, in event recommendation, weather forecast data are important, especially for outdoor events. Making recommendations by considering weather forecast information will provide better results improving the quality of the recommendation for the users. There are many studies, such as STS [7], which includes weather data as context to use it in recommendation, and the system may alert the user to avoid attending an event, due to bad weather at the event date.

In context-aware event recommendation and notification systems, semantic analysis may create a value for additional relevant event service and event item discovery. For instance, in the tourism domain, systems utilize tourism related ontologies to categorize tourism items for better recommendation results. Accordingly, the computer becomes easily aware of the event information and classifies it by means of semantic analysis. When classification information is not available in the domain, the system may become inadequate in trying to find semantic relationships. Therefore, ontologies may not be successful in such a domain. Linked data [40] is another approach in semantic knowledge representation that is used to display and connect data on the Web from various resources. Linked data method enables machines to easily read the knowledge unit and find related information about the unit. To achieve this, the computer uses the Web to find the most related data with the specified information by exposing the connections of the data. Therefore, this method may prove to be useful when ontology is insufficient or any information classification method is not available. When everything is considered, using semantic analysis with linked data method has the potential to improve related information discovery about events and other items in the specified domain.

Another property of context-aware event recommendation and notification systems is that they enhance their recommendation quality with case based reasoning by evaluating complex conditions. They achieve this by defining rules to evaluate the conditions and trigger actions. Business rules [47] are used in software systems to make a separation between software implementation and business logic so as to modify the conditions with low maintenance effort. One advantage of business rules is that they have exact time and date to be executed, so the business rule activates only at specified times, and there is no need for spending effort on arranging a job scheduler [49] to trigger an action at a particular time. This feature may have significant role in delivery time of the recommendation message.

In message delivery, context-aware recommendation systems can use the publish-subscribe approach [46] to filter messages and send only related messages to related subscribers who are registered to interested topics only. In the publish-subscribe approach, the publishers create content and add related topics to the content. The subscribers specify topics that are interesting for them. Then, the subscriber receives related messages from the subscribed topics. Another possibility is that they can specify message content constraints to retrieve related messages. The publish-subscribe approach provides successful message filtering so that the subscriber views only related messages. In mobile devices, push notification services can be used for message delivery. Push notification [50] is publish-subscribe based message delivery by a centralized server to an endpoint client without specific request from the client. Therefore, with this service, mobile devices easily receive messages anytime without requesting information from the server.

As an enhancement to the former studies, we propose a publish-subscribe based design to present a context-aware event notification system. The presented system provides added value with a business rule engine that can gather all contextual data such as location, time, weather forecast data, user preferences and user interests. The business rule engine deduces valuable knowledge from the contextual data, and control the time to deliver the message at the best moment for the subscriber. Secondly, the other added valued feature is offering semantic knowledge with the linked data approach, in order to allow automatically setting the related topics of the event, as a contribution to the main task of the publisher. This feature may prove to be significant where ontology is not available. In our proposed system, the subscribers set their interests (tags) in order to receive related events. In order to plan message delivery time and arrange message delivery conditions, they can set and modify their message delivery preferences through the event management system that takes into account time, location and/or weather forecast context data while sending messages. The publisher creates events with the tags they specify, and publish it to the event management server. According to the preferences and the interests of the subscriber, finally, the subscriber can be notified with the related event messages.

This thesis is structured as follows:

- Chapter 2 gathers information about related studies and discusses context aware event recommendation and message delivery systems.
- In Chapter 3, we propose the conceptual design of our context aware event notification system and describe it in detail.
- In Chapter 4, we describe our prototype based on our conceptual design. We also explain use case scenarios of our implementation.
- In Chapter 5, we create test sets and sample scenarios for evaluation of our prototype. By disabling the modules in the system individually, we evaluate the results at the absence of the specified module, in order to highlight the importance of the corresponding proposed module. We also describe the significance of the full working system. Furthermore, we discuss applying the proposed system at METU Informatics Institute environment.
- In Chapter 6, we briefly summarize the related work, current problem, the proposed system and the prototype. Finally, we list future plans regarding our current work.

CHAPTER 2

RELATED WORK

Context-awareness is related to sensing environmental conditions or situations as well as user parameters, and helps evaluate current conditions and act according to the situation. In context-aware event recommender systems, there are many environmental and situational inputs to consider and evaluate. In addition to location awareness, the systems might use time of day information, weather information, user preferences, user history and demographic info (e.g. age, gender). In delivery and recommendation mechanisms, some systems use business rules. They define business rules and the system can change its behavior dynamically, without updating the system programmatically. The systems may use user history, i.e., track previous activities of the user and infer knowledge. Moreover, by looking at the past user activities, the system might categorize the experienced activities and appoint a profile for the user accordingly. For example, in the tourism recommender system m-To-Guide[1], the system sets a profile for the user according to the travel type of the user, leisure travel or business travel, and recommends services according to that profile.

In context-aware event recommender systems, location is the most significant factor, since these systems offers places or events in the nearest area to make the user be able to join. Location information is obtained generally from a mobile phone via GPS. Speta [2], Gulliver's Genie[3][36], LoL@[4], Maiden[5] use GPS to locate user's location and find the nearest services. LoL@[4] is a location based mobile application that provides tourism information in Vienna. It aims to give information about predefined tours, about visiting places and also provides routing to navigate tourists to the interesting places. Speta [2] gives tourism related recommendations, aggregates social networks, semantic Web and context awareness (location, weather forecast, time, user preferences, friend's recommendation and history). The recognition of user preferences is done by two ways. One is explicitly specified interest of the user, and ratings to the attractions are taken into account. In addition to this, social media interests, such as favorite painter, are considered for user preferences. Another way is inferred from user behavior. For instance, the type of the museums, which are always visited by the user, is taken into account. The social network integration part is limited to adopting OpenSocial API[11], and only People (information about people and their relationships with each other) and Activities (information about what people are doing.) elements are used.

In context-aware recommendation and notification systems, in order to gather semantic information about the recommendation entities, such as place, events and interests, the systems may utilize semantic analysis through the use of ontologies. Ontology [39] is a structure that aims to represent knowledge as a hierarchy of notions in a domain using a common word to define types, properties and relationships between these notions. In the semantic part of Speta [2], the ontology consists of entities, relations and the axioms. There are abstract entities such as 'City', 'Hotel', 'Restaurant', and instances such as 'Paris' and

Hotel Ritz'. After all the context information is gathered, a recommender algorithm, which is based on hybrid filtering [12] approach, is used to give the relevant information to the user. In this system, there is no event driven and notification component. The recommender runs if the user queries the system.

SigTur/E-Destination [38] has shown a good feature on using semantic environment that is settled on tourism ontology to explicitly classify the activities to suggest among predefined tourism concepts. We can give a sample for the tourism ontology that SigTur uses:

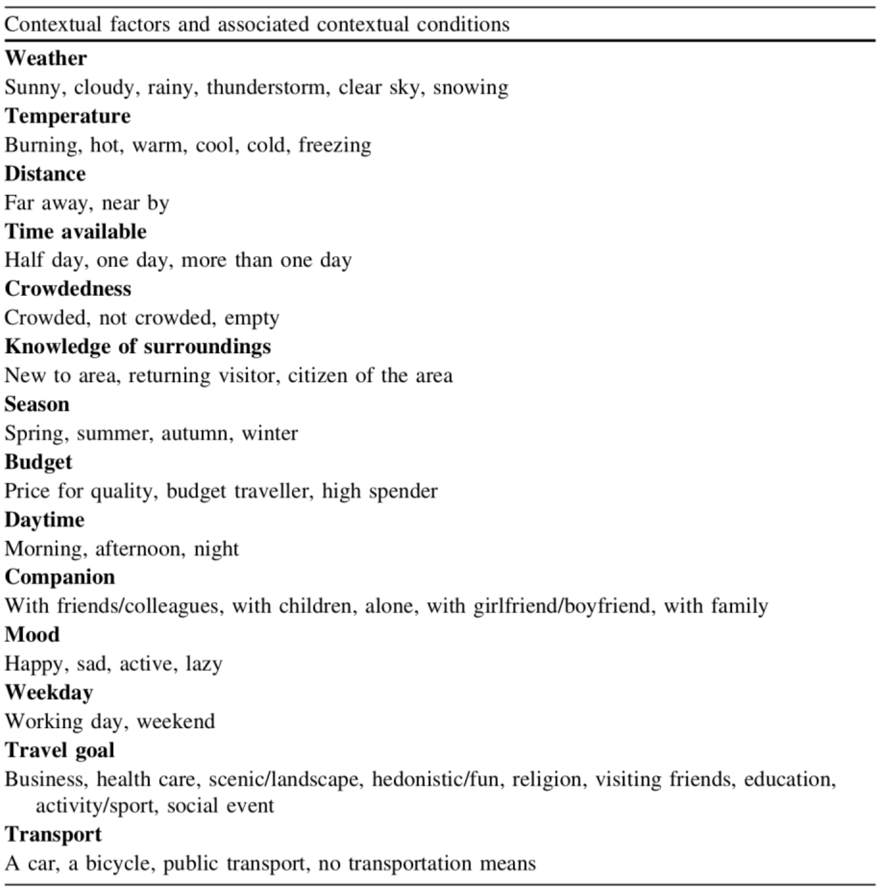
Sailing –> AquaticSports ->Sports - > ItemType
As a second example*: HistoryMuseums* -> *Museums* -> *Culture* -> *ItemType*

Moreover, Sem-Fit [13] shows a good example on hotel recommendations by combining hotel ontology and fuzzy engine in order to increase precision on recommendation success. When the activity name or hotel name is available in the ontology, semantic analysis will have good performance on categorization. However, when an activity/hotel name is not available in the ontology, the name will not give any idea about what is related to or what it means. Thus, setting ontologies might sometimes remain inadequate in event/place recommendations. When ontology remains insufficient, some studies use alternative ways, for instance, they might use linked data method. A study about context-aware music recommendation [51] shows an example of creating inter-ontologies using linked data method, in addition to music ontology. It achieves this by aggregating related data from Last.fm [52], Yahoo! Local [53], Twitter [54] and LyricWiki [55], then find the relations between terms, lyrics and other words to recommend related music for users, so that the system will be able find up to date related information about music, songs and their lyrics, while music ontology alone cannot provide that much information.

Gulliver's Genie [3] [36] is another context-aware tourism guide system. It utilizes user defined interests and location information and user demographic data such as age, gender and nationality. An important feature of this system is that it uses push notifications. When a user is near a sight or a visiting place, the information about this place is pushed to the user's mobile device as notifications. Only push trigger of this system is location based, meaning that, the push notification is triggered only if the user is in the neighborhood of the particular place. MobiDenk [5] provides current information about places of interest, by utilizing GPS location information. In addition to the GPS data, it also considers the movement speed and direction of the user.

Generally, it is assumed that the places to visit and the activities to join are outdoor locations. However, some interesting places could be indoor locations, and GPS alone may be insufficient. At this point, some studies include different technologies for indoor positioning. eAgora [6] (Agent based architecture for context-aware and personalized event recommendation) system, proposes Wi-Fi, Bluetooth and ZigBee [14] technologies for indoor positioning. It recommends academic or cultural events based on location, time and profile information by using spreading activation method [28]. In this system, there is no delivery mechanism with notifications or event driven approach.

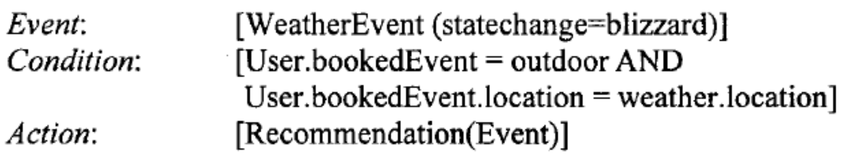
In event recommendation systems, weather and time information may be very significant especially for outdoor events. The awareness of weather forecast gives an extra added value to the system and it may be very important in such a pervasive environment. The system STS [7] offers context aware points of interest suggestions based on weather conditions. The system recommends touristic places with suitable weather conditions. If the user wants to go to a recommended place, he/she can bookmark the place. If the weather changes from e.g. sunny to rainy, the system alerts the user to revise his/her choices. The system uses rich context factors to determine recommended places as in Figure 1, however many of these context info is taken by the user, thus, the inference engine could be weak, since the learning is done by ratings of the places in addition to the user profile information, and the alert mechanism is limited to weather change conditions. Finally, the system cannot recommend new places against bad weather conditions.



Contextual factors and associated contextual conditions
Weather
Sunny, cloudy, rainy, thunderstorm, clear sky, snowing
Temperature
Burning, hot, warm, cool, cold, freezing
Distance
Far away, near by
Time available
Half day, one day, more than one day
Crowdedness
Crowded, not crowded, empty
Knowledge of surroundings
New to area, returning visitor, citizen of the area
Season
Spring, summer, autumn, winter
Budget
Price for quality, budget traveller, high spender
Daytime
Morning, afternoon, night
Companion
With friends/colleagues, with children, alone, with girlfriend/boyfriend, with family
Mood
Happy, sad, active, lazy
Weekday
Working day, weekend
Travel goal
Business, health care, scenic/landscape, hedonistic/fun, religion, visiting friends, education, activity/sport, social event
Transport
A car, a bicycle, public transport, no transportation means

Figure 1 Used Context Factors

Moreover, in CAIPS (Context-Aware Information Push Service in Tourism) [8], the system stores planned trips and scheduled events for the user. If the weather is bad for the specified event time, the system sends a notification to the user that the event is not suitable for the bad weather. Instead of the planned event, the system offers new recommendations for those weather conditions. An example of how new event is offered due to bad weather condition as in Figure 2 is shown.



```
Event:        [WeatherEvent (statechange=blizzard)]
Condition:    [User.bookedEvent = outdoor AND
               User.bookedEvent.location = weather.location]
Action:       [Recommendation(Event)]
```

Figure 2 In bad weather and outdoor condition, new recommendations are offered

Business rules are another significant factor for context-aware event recommender and delivery systems. In past studies, we can talk about business rule integrations to the systems. In CAIPS [8], the system offers improvements by the business rules defined by the tourism experts. The tourism experts create rules by means of a graphical user interface to arrange push messages to the users. For example, the expert creates a rule that pushes information about new recommendations against bad weather conditions, according to the user profile. In terms of learning and improvement of the system itself, the capability generally depends on the tourism experts. The business rules are limited to arranging push notifications and the system itself cannot infer knowledge by considering the business rules.

We can discuss notification mechanisms in past studies. A former study about notification system [9] has a different approach in time, location and user profile awareness. It proposes an in-campus notification system that delivers the right information at the right time and place to the right person. It succeeds in the approach by using RFID feature of the student cards. The student can define his/her preferences and interests from a web application. Some RFID reader terminals connected to the PCs are put at important places, such as sport center, faculty buildings, cafés etc. When a user tag is read by a terminal, he/she can get related information for him/her, at the right time and place. The information may be an academic announcement in the faculty building, which informs the students about a cancelled morning course, or it may be a sport event notification to invite the student to join the event in the sport center. The system can be assessed as successful in delivering right information to the correct person at the most suitable right and time. However, the user interaction level with the system is very low and there is no feedback from the user except for arranging the preferences from the web application. Thus, there is a low quality learning mechanism and weak dynamic adaptation of the system. The sample delivery rule process is as follows [9].

Time = '5 pm'
Identity = '1038'
Location = 'Sports Complex'
Preferences = 'Sports'

Then the rules will be presented:

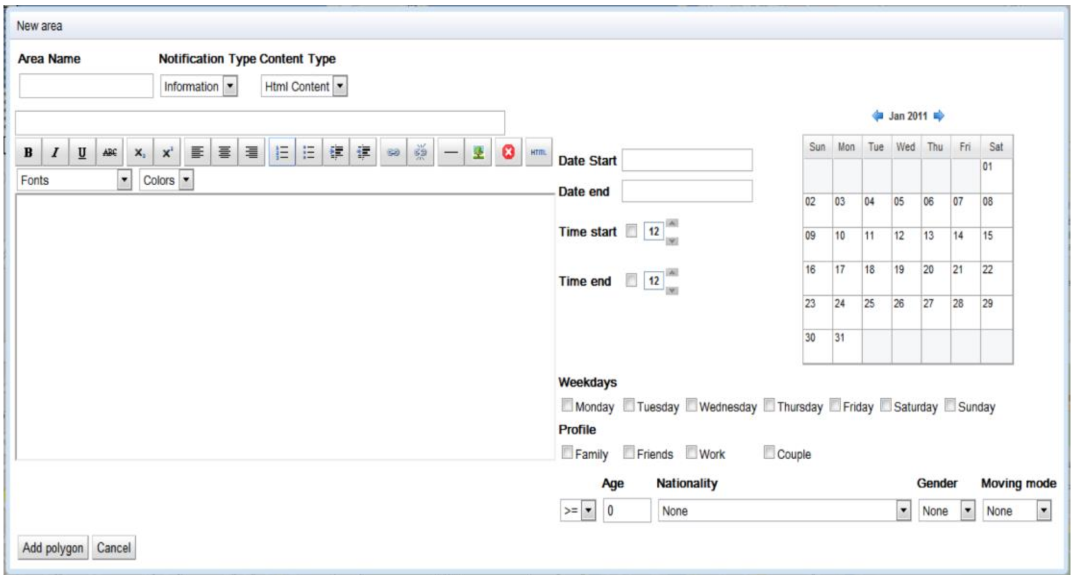
IF hour=5
AND ampm = 'pm'
AND location = 'sports_complex'
AND preferences = 'sports' THEN display_notification = "inter-varsity football league is on now"

Furthermore, "liveCities" [10] proposes an improved notification system by including all tourism entities such as a restaurant or a bus company, as a notifier, meaning that, a restaurant company is able to send notifications to the related people by specifying notification message and the person type that might be interested. The system takes the user profile info from the user, a notifier can specify the user profiles of the recipient to send the notification, and the system sends the related info to the user according to location and time. A scenario can briefly describe the system [10].

Scenario: If a user is walking with the friends around a restaurant at lunchtime and the weather is good, the system pushes an offer to have a lunch at the terrace.

Tourism entity: fast food restaurant
Notification: a discount to have lunch at the terrace.
Areas: An area has been created around the restaurant.
Context parameters: location, moving mode(walking), time(from 12 to 14), age (<30), social context (friends) and temperature(>20 °C)

As seen in the scenario, the system uses location, time and weather data, user profile and social context (user scans nearby devices via Bluetooth and assign a role: family, friend, workmate or couple) data.


New area

Area Name
Notification Type: Information
Content Type: Html Content

Fonts
Colors

Date Start
Date end
Time start 12
Time end 12

Jan 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						01
02	03	04	05	06	07	08
09	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Weekdays: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

Profile: Family, Friends, Work, Couple

Age: >= 0
Nationality: None
Gender: None
Moving mode: None

Add polygon
Cancel

Figure 3 Notification Manager of liveCities system [10]

A tourism entity, such as a restaurant, can describe the notification settings, for example, what property of the users to send, e.g. age, nationality, gender, moving mode, and also the time can be determined. All settings are defined in the notification creation phase, and there is no complex inference engine and business rule engine except for translating defined settings into rules. Moreover, there is no user activity tracking and learning mechanism, since most of context data are defined explicitly or pulled from some external service such as weather. Nevertheless, it can be assessed as successful in terms of personalization, context awareness, and in terms of sending right information to the right person.

There are remarkable studies in publish-subscribe approaches for context-aware systems. Caglar [25] shows a good work on proposing context-aware reminder system based on publish-subscribe model. The system, which uses time and location as context data, composes reminder patterns for reminding events to the subscriber that it registered before. The publisher component dispatches the reminder information, such as event detail or event cancellation info, via the message dispatcher.

CHAPTER 3

PROPOSED CONCEPTUAL DESIGN

In this part, conceptual design of our context-aware event notification system is explained in detail. The aim of the proposed system is to deliver event notifications to the right users at the right place, right time, and under suitable conditions. To achieve this, the proposed system should be able to use various information about the user and the environment for evaluation of the conditions. While receiving information about the users and the environment, the users must not be interrupted much, so the system should have the ability of autonomously receiving information and deducing knowledge from such data. Another aim is that the system information source should be open and flexible to be accessible everywhere and provide easy information flow especially in composing event information. Thus, the system knowledge pool can be extended easily and the users can retrieve more number of event recommendations. Furthermore, the users should receive only related messages under suitable conditions according to context data such as time, weather, and place. They must not get messages regarding unrelated events that they may not be able to attend or enjoy. While allowing fast growing information source of the system, there should be mechanisms for filtering data and inspection of the relationship between user interests and the events.

3.1. Main Features of the System

Main features of the system include Message Delivery, Business Rule Management System for Context Data, and Semantic Analysis Using Linked Data, as explained next.

3.1.1. Message Delivery

In order to satisfy the aim of sending messages to the right users as explained above, a publish-subscribe based approach is utilized in the proposed system. In systems incorporating message delivery functionality, all messages are typically not sent to all recipients. There are various methods for selective sending of the messages to the related recipients. Publish-subscribe model [46] is one of them and it provides a message filtering approach to send the message to the related recipients, in other words, the subscribers. In this model, the subscribers can register to the topics that are defined by the publisher, or they may specify the message content constraints to receive related messages. The advantage of this approach is that it presents loose coupling between publisher and subscriber. Publisher does not need to know who the subscriber is or whether the message is sent to the subscriber or not. With this weak requirement, another advantage is that the model is scalable, meaning that, publisher can send the message to a large number of subscribers without intensive effort. In contrast, publish-subscribe model does not guarantee that the message is sent to all related subscribers. In a system that aims assured message delivery, the model might need to be supplemented accordingly. Moreover, the model may assume that the subscriber is listening, when it is not. The information of whether the subscriber is listening will be missed in this approach.

There are successful studies in proposing context-aware publish-subscribe model based message delivery systems. The study of Caglar [25] proposes a publish-subscribe model based context-aware reminder system with reminder patterns. The proposed system uses time and location as context, and uses them in the reminder pattern in order to remind an upcoming event to the subscriber. For instance, the publisher can create a generic reminder template by specifying the location and the radius to remind the event in or out of the radius, or specifying the time interval to trigger event reminder. The subscriber registers to the topics in order to receive related reminder information of these topics. Then, the publisher creates an event specifying the topic and the reminder pattern. Finally, the subscriber receives reminder messages from subscribed topic channels. In the study, the system is successful in using time and location as context, and takes action accordingly so as to deliver information to the right person. In terms of context-awareness, this approach of proposed system can be extended and can be made more flexible in using and managing context data, rather than just utilizing predefined patterns limited to time and location info. We intend to extend this approach by allowing managing different context data from different sources without depending on the predefined templates and without changing the system. Furthermore, we aim to make a contribution to this publish-subscribe approach by semantically analyzing the published items and inferring new topic matches in order to add them to the published item. Automatically setting related tags for the published items is valuable as the publisher may miss some information regarding related tags or topics when publishing an item.

We propose to use the topic-based [46] publish-subscribe approach in publishing and delivering event notifications to correct users. The publisher has the ability to compose the events by attaching event related user interests and publish it for the subscriber. The subscriber is another user of the system, which is able to receive event messages by specifying user interests (tags) and which allows the system to gain knowledge about itself to get appropriate events with minimum interruption. The broker [46] is responsible for presenting the user interests (tags), publishing and delivering the events to the related subscriber coming from the publishers. Our system is composed of publisher, subscriber, event management system, which can be described as an enhanced broker in the publish-subscribe model [46], and context sources, which are used in retrieving subscriber and environment information. To arrange message delivery conditions, the subscribers set their message delivery preferences such as time, location, and weather forecast context data related to sending messages through the event management system. The publisher composes the events with the tags they specify, and publish them to the event management server. According to the preferences and the interests of the subscriber, finally, the subscriber is notified with the related event messages. As additional contributions to the work of Caglar [25], we propose to use a business rule engine, which allows easy evaluation of different context data and inferring knowledge, in order to utilize different context sources and to allow removing the limit of utilizing only predefined patterns that use place and time. Furthermore, we propose to incorporate the ability of semantically analyzing published items by using linked data, in order to discover all related tags, when the publisher does not fully specify all related tags. These two features of the proposed system are described next.

3.1.2. Business Rule Management System for Context Data

In software systems, business rules [47] present a solution that can handle dynamic business logic by separating business knowledge and the implementation. Thus, it provides a flexibility that if the business knowledge has to be changed, there is no need for modifying the implementation, so this decreases the software maintenance costs and provides more dynamic behavior of the system itself. The rules are used to help decision-making, to deduce new knowledge or to trigger actions based on the conditions. While achieving this, newly entered input data and newly entered events are evaluated dynamically without modifying or restarting the system. Business rule engine is a software system to execute different business rules. Business rule engines are used as a pluggable software component, therefore, this separation provides that running and modifying the rules to change business logic does not need an IT (Information Technology) system intervention. This situation allows changing the system without need of a technical person. Business Rule Management System (BRMS) [48] is a software system for business rules, which is used to deploy, execute and maintain the decision logic. It is a standalone whole system that contains a business rule engine and business rules to execute. Many BRMS applications exist in the software industry. For instance, Drools [33] is an open-source Java programming language based BRMS of JBoss [21]. It presents core business rule engine, a web authoring and rules management application. Drools BRMS can run independently from the actual implementation and defining Drools rules keeps simplicity. Therefore, it aims flexibility, cost effectiveness and ease of use.

In the related studies, we have witnessed that business rules are used in event recommendation and message delivery. The business rules, which are proposed, generally focus on one particular work. For example, the rules are defined only for message delivery such as in specifying delivery time, or only for evaluating conditions such as checking weather condition. In our system, we propose a flexibility to use the business rule management system for evaluating different context data, trigger actions according to them and arranging message delivery for the most suitable conditions for the subscriber.

3.1.3. Semantic Analysis Using Linked Data

In some cases, the publisher may not have enough knowledge for defining related tags of the event that is about to be created. The event may be published with missing information so the related subscribers cannot receive the event recommendation that actually would be suitable for themselves. The solution to this problem could be creating a semantic environment for the event detail in order to analyze it semantically and infer new knowledge. With new semantic knowledge, the system will be able to match new tags as a contribution to task of the publisher. As a method of semantic categorization of the event info, we can consider creating ontologies of the related domain. Ontologies are successful in categorizing the events only if event name is available in the ontology. If we think of semantic environment with ontologies, the ontologies may remain unsuccessful in matching names that does not exist in the ontology. In such cases, the words do not mean anything to the system in order to add new tags into the published event. Against this problem, we can provide an alternative method that infers and lists words related to the descriptions in the event detail. For this purpose, linked data method seems appropriate to use, as the similar technique is used in context-aware music recommendation [51]. Linked data is a method in semantic Web domain that is used to expose and interlink data on the Web from various resources. Berners-Lee [40] describes linked data principles as:

1. Use URIs [41] as names for things.
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards. (RDF [42], SPARQL [43])
4. Include links to other URIs, so that they can discover more things.

As seen in the principles, each name has a URI and links to other URIs, so this situation enables computers to easily read the information of the things and connections among them in Internet. In other words, it eases listing the related words of one specific word for computers as humans can already semantically do. There are many applications and datasets that compose linked data feature. For instance, DBpedia [16] is a large dataset that contains extracted data from Wikipedia. It links different data sets on the Web to Wikipedia data. FOAF [44] is a dataset containing persons, their features and relationships. GeoNames [45] presents RDF descriptions of more than 7,500,000 geographical features. These datasets are used for extracting relational information about current data. For instance, when we want to get geographic relational information about "Ankara" city, we can receive position (longitude, latitude) of Ankara, its counties, and population information via GeoNames as Table 1 shows the examples.

Table 1 Geonames Output Samples

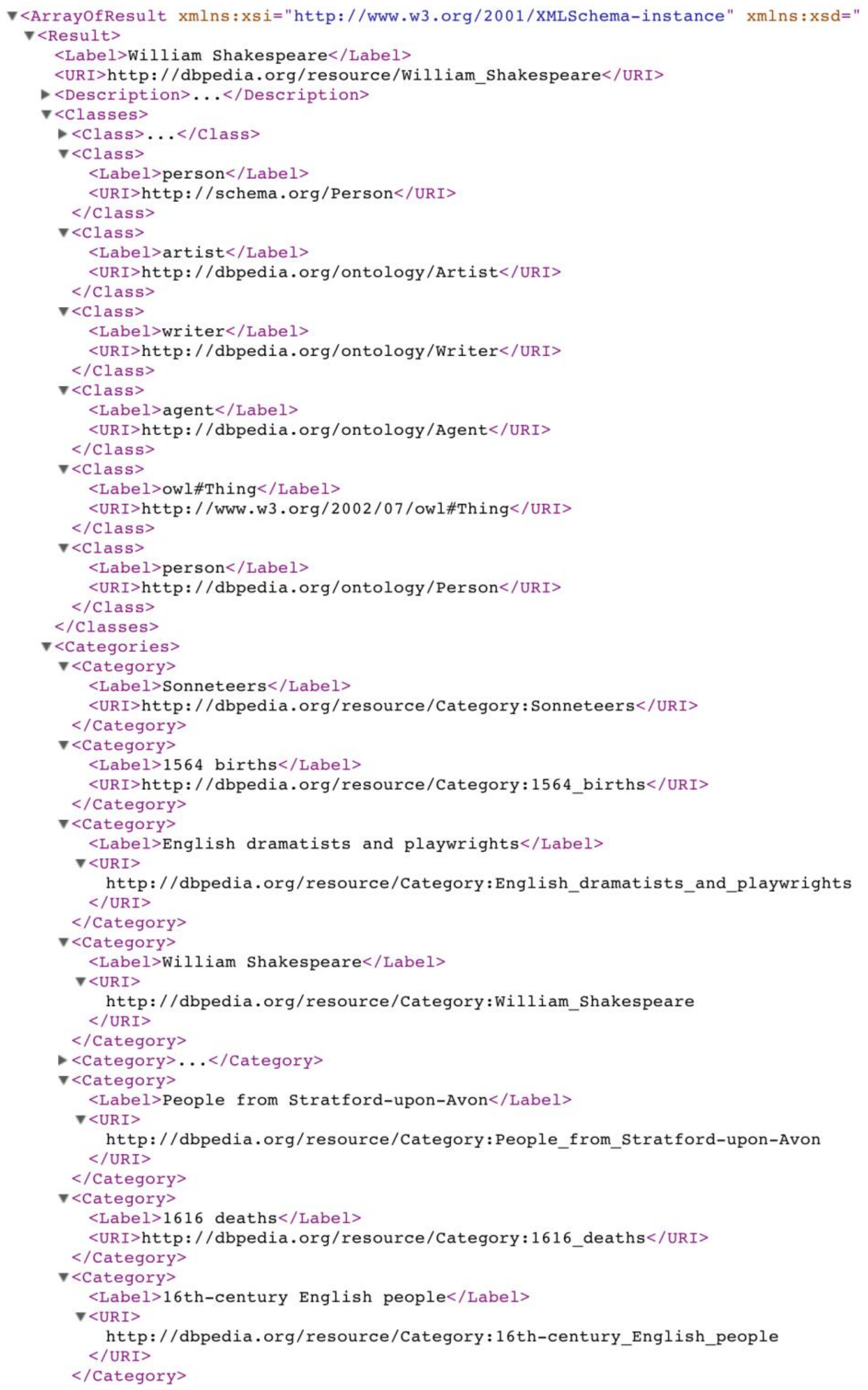
GeoNames Query Input	**GeoNames Results**			
Name	**Name**	**Country**	**Feature class**	**Latitude & Longitude**
Ankara	Ankara NK,Anakara, Ancara, Ancyra, Ang-ka-la,Angkara, Angora, Anguriyah, Ankar,Ankara, Ankara khot,Ankaro ,Ankuara...	Turkey, Ankara	capital of a political entity, population 3,517,182, elevation 850m	N 39° 55' 11", E 32° 51' 15"
London	London City of London,Gorad Londan,ILond on,LON, Lakana, Landen,Ljond an,Llundain, Londain,Lond an,Londar,Lon de,L...	United Kingdom, England Greater London	capital of a political entity population 7,556,900	N 51° 30' 30", W 0° 7' 32"
Taksim	Taksim	Turkey, Istanbul	section of populated place	N 41° 2' 13", E 28° 59' 11"

When we want to ask relational information about "skiing", via DBpedia, we can get many categories and classes of "skiing" from different sources, such as "Olympic sports", "activity", "winter sports", not just the information of "winter sports" as it generally exists in sports ontology. As a different example, we can connect to DBpedia Lookup Service [17], query about "William Shakespeare" word to list categories and classes of William Shakespeare. We can achieve this job by arranging the URL with "William Shakespeare" input, then we retrieve an XML file as output.

URL for William Shakespeare:

http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?MaxHits=1&QueryString=William+Shakespeare

The XML file output will be as follows:



```
▼<ArrayOfResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
 ▼<Result>
    <Label>William Shakespeare</Label>
    <URI>http://dbpedia.org/resource/William_Shakespeare</URI>
  ▶<Description>...</Description>
  ▼<Classes>
    ▶<Class>...</Class>
    ▼<Class>
       <Label>person</Label>
       <URI>http://schema.org/Person</URI>
     </Class>
    ▼<Class>
       <Label>artist</Label>
       <URI>http://dbpedia.org/ontology/Artist</URI>
     </Class>
    ▼<Class>
       <Label>writer</Label>
       <URI>http://dbpedia.org/ontology/Writer</URI>
     </Class>
    ▼<Class>
       <Label>agent</Label>
       <URI>http://dbpedia.org/ontology/Agent</URI>
     </Class>
    ▼<Class>
       <Label>owl#Thing</Label>
       <URI>http://www.w3.org/2002/07/owl#Thing</URI>
     </Class>
    ▼<Class>
       <Label>person</Label>
       <URI>http://dbpedia.org/ontology/Person</URI>
     </Class>
   </Classes>
  ▼<Categories>
    ▼<Category>
       <Label>Sonneteers</Label>
       <URI>http://dbpedia.org/resource/Category:Sonneteers</URI>
     </Category>
    ▼<Category>
       <Label>1564 births</Label>
       <URI>http://dbpedia.org/resource/Category:1564_births</URI>
     </Category>
    ▼<Category>
       <Label>English dramatists and playwrights</Label>
      ▼<URI>
         http://dbpedia.org/resource/Category:English_dramatists_and_playwrights
       </URI>
     </Category>
    ▼<Category>
       <Label>William Shakespeare</Label>
      ▼<URI>
         http://dbpedia.org/resource/Category:William_Shakespeare
       </URI>
     </Category>
    ▶<Category>...</Category>
    ▼<Category>
       <Label>People from Stratford-upon-Avon</Label>
      ▼<URI>
         http://dbpedia.org/resource/Category:People_from_Stratford-upon-Avon
       </URI>
     </Category>
    ▼<Category>
       <Label>1616 deaths</Label>
       <URI>http://dbpedia.org/resource/Category:1616_deaths</URI>
     </Category>
    ▼<Category>
       <Label>16th-century English people</Label>
      ▼<URI>
         http://dbpedia.org/resource/Category:16th-century_English_people
       </URI>
     </Category>
```

Figure 4 William Shakespeare Output XML File

As the output, the system receives an XML file (as seen in Figure 4) that includes information about William Shakespeare such as description, categories, classes etc. After selecting the categories and classes from the XML file, we can get:

- person
- artist
- writer
- Sonneteers
- 1564 births
- English dramatists and playwrights
- 16th-century English people
- People of the Tudor period
- English poets
- English Renaissance dramatists

As seen from the datasets and the examples, linked data applications and datasets help machines to easily find and expose the relations of words from different sources. We can say linked data method has an extended feature of finding a set of related words that are both available and not available in ontology. On the contrary, linked data feature may sometimes mislead the computer to find the relations of words. Since this feature extracts the relations that are mostly referred, sometimes, maximum number of references may not be the actual relational information. This may happen when there is a word that has two meaning, such as both as a place name and a person name.

3.2. System Architecture

After considering these two main features above, we can explain event management server in which the corresponding modules provide these two features. In order to store event details, topics (tags), manage the whole event processes and the relationships between the modules we propose to use the event management server which incorporates three sub modules: business rule management system, semantic tag matcher and event dispatcher. So as to deliver messages at the most suitable time, and gathering all context information in one place then infer knowledge special to user, and to satisfy the first feature, we propose business rule management system. We can use internal and external context resources connected with the business rule management system. For our system, we can use location data of the subscriber, time information and weather forecast data as context. In order to analyze event names semantically and automatically specify tags of the event, and to satisfy second feature, we propose a semantic tag matcher utilizing linked data. This module will be helpful in case the publisher misses specifying all related tags for the created event and the subscriber will not be able to view the related event. Linked data approach allows finding additional related data about the created event and then tries to match the related data with the existing tags. In order to deliver messages to the subscribers at the right time, we also include event dispatcher module. We propose event publisher to publish events and set the topics for the events. Finally, we propose event subscriber to register to the topics and receive related event notifications. You can see below figure of our conceptual design architecture:

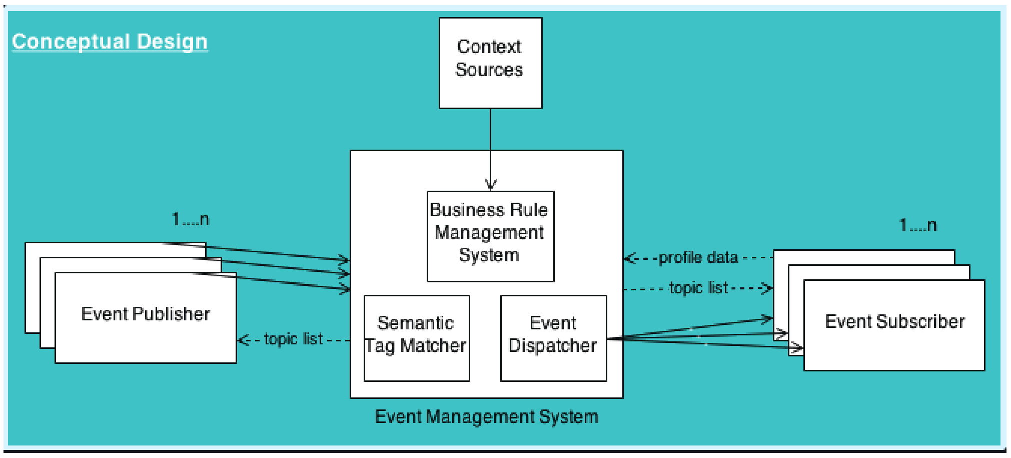


Figure 5 Architecture of Proposed Conceptual Design

3.2.1. Event Publisher

In our proposed conceptual design, event publisher has the main responsibility of event creation and event publishing for the subscriber. It queries event topics from event management system and publish the event with specified topics in order to be sent as recommendation to the related subscribers. Event publisher has one connection. It connects to the event management system to query as well as create new topics and sends the event details to be published. Event publisher has to pass the authorization phase to query or create topics, and create events.

In our proposed system, the publisher does not need to consider the context data such as environmental, social and time situations. For instance, the publisher does not have the responsibility of setting time to deliver the event message, however, it has chance to optionally set the time, location or weather data conditions to trigger sending events.

The event to be published has these content fields.

- Event Title: Name of the event.
- Start Time: Start date and time of the event.
- Finish Time: Finish date and time of the event.
- Description: Extra information about the event.
- Place: Location of the event.
- Event Type: Indoor or outdoor event info.
- Event Id: Unique identification info of the event.

The publisher has the opportunity to set trigger conditions via business rule management system. The conditions are as follows:

- Weather forecast condition: The publisher may set the weather conditions to send the event messages. For instance, the publisher can arrange that the event message is sent only if the weather condition is clear sky on the event day.
- Time condition: The publisher may set the event delivery time specifically. For example, the publisher can set an event message delivery time to 09:00.
- Location: The publisher may set the message delivery action according to neighborhood of the subscribers. For example, the publisher can set that the event message is only delivered to the subscribers who are in Ankara.

3.2.2. Event Management System

In our proposed conceptual design, event management system, which can be referred to as an enhanced broker in the publish-subscribe model, has the main responsibility of managing events coming from publishers, and distributing the events to related subscribers. The topics (tags), events and their details are held in event management system.

Event management system has three sub modules, business rule management system, semantic tag matcher and event dispatcher. Semantic tag matcher helps to find more related topics for created event by analyzing event info semantically. Even if the publisher does not specify all related tags for a created event, semantic tag matcher tries to specify the tags related to the event. Semantic tag matcher uses linked data method to find related words for the event. It receives the related words by querying linked data datasets with event details as input. Then, it tries to match the new received words with existing tags. If related words of the event title have a successful match with a tag name, the event is now ready to be delivered to the subscribers who subscribed to the matched tag or tags, in addition to previously specified tags that the publisher sets.

Secondly, event dispatcher sub module has the responsibility to distribute the event messages to the related subscribers with the trigger from BRMS. Event dispatcher stores the identification info of the subscriber devices in order to send the message to the right subscribers. Event dispatcher also has the ability to register new identification info of the device when a new subscriber device is registered to the system.

Thirdly, business rule management system (BRMS) is a subsystem in the event management system that infers extra information from context sources such as time, place, and weather information, and trigger actions according to the inferred information. It determines time to publish the messages. The system achieves this by defined business rules. In our proposed BRMS, new business rules can be added and the existing business rules can be modified via business rule editor interface. BRMS presents flexibility to trigger sending messages by considering any single or multiple context data. BRMS helps the event manager system when to deliver the message, help about what the message content will be and help about how the event recommendation will be. To achieve this, BRMS uses preferences data of the subscriber. There are three main preferences that the subscriber can set; time, location and weather forecast.

- Time preference: The subscriber has an option to retrieve messages in particular times. When an event is created, the message may not be directly sent to the subscriber. The message can wait for the selected time or time interval that the subscriber prefers. For instance, the subscriber may set his/her time preference in order to receive message in out of business hours. In this case, if the event is created in business hours, the notification waits until the business hours pass, then the subscriber receives the notification.
- Location preference: The subscriber has an option to receive event messages that is arranged in the subscriber’s neighborhood, or he/she can turn off this preference in order to receive all related messages that do not depend on location.
- Weather preference: The subscriber can set his/her preference to receive event messages or not receive messages depending on weather data in the event date. If the event is outdoor event and the weather is rainy, or we can say the weather is bad, the subscriber is avoided to receive the message. He/she can turn off this preference in order not to depend notifications on weather data.

BRMS can use these preference data in order to evaluate the conditions and trigger actions accordingly. For instance, by means of BRMS, we could set time triggered message delivery, such as sending message every day at in business hours (08.00-16.00). An example algorithm can show a time triggered event message. Let’s say BRMS has a rule that controls sending the message about Event 1 to the Subscriber A in business hours. In this kind of approach we do not need a job scheduler [49] to trigger action in the specified time. BRMS is able to easily handle time-triggered events. Here is the example of a business rule:

```
When
(Time is in business hours)
Then
Send message (Event 1) to Subscriber A
```

We could set weather info triggered message delivery by means of BRMS. Or we can set location info, weather info and time triggered message delivery, if we define that kind of business rule without changing any code from the other parts of the event management system. Therefore, we present more flexibility relative to predefined templates in order to set trigger conditions to deliver the message. We can give an example of a business rule algorithm that evaluates different context data.

```
When
(Time is in business hours)
AND (Event 2 location is near to Subscriber B location)
AND (Weather will be good in Event 2 date)
Then
Send message (Event 2) to Subscriber B
```

For that kind of rule, BRMS will avoid sending the event message to Subscriber B in bad weather conditions, or BRMS won’t send event message when Subscriber B is far away from Event 2 location. However, Subscriber B may turn off location, time and weather dependent recommendation that we will talk about in Event Subscriber part. Therefore, the example rule will work for the Subscriber 2 without considering location, time and weather data.

```
When
()
Then
Send message (Event 2) to Subscriber B
```

In addition to specific time definition, BRMS is able to use time information and able to categorize it in two ways: Day categorization and hour categorization. Day categories are weekend and weekdays. Hour categories are business hours, non business hours, and night hours. We can give a categorization example of a selected date:

Table 2 BRMS Time Inference

Input Date	BRMS Inference
August 29, 2014 09:00	Friday, Weekday, Business hours
July 7, 2014 20:00	Monday, Weekday, Non Business Hours
July 6, 2014 01:00	Sunday, Weekend, Night hours

BRMS achieves receiving weather information by connecting to weather information services and deduces the knowledge that the weather is good or bad. We can give an example of connecting to weather services and deducing information.

Table 3 BRMS Weather Condition Inference

BRMS Request Input	Weather Forecast Service Response	BRMS Inference
Ankara August 29, 2014 09:00	Clear Sky 19 °C - 32 °C	Good
London August 31, 2014 15:00	Heavy Intensity Rain 19 °C - 25 °C	Bad

After retrieving the weather data, BRMS will use the weather inferences in evaluating conditions and sending messages to the subscribers.

Another feature of BRMS is deducing subscriber behavior and set a time profile for the subscriber by tracking the interaction time of the subscriber with the event message, as we explain above in “Time preference” item. We can visually show the principle of assigning of time profile. In addition to manually setting of time preference that we will talk about in Event Subscriber part, BRMS can automatically set a time profile for the subscriber as in Table 4.

Table 4 BRMS Subscriber Interaction Inference

Event Message Sending Time	Subscriber Interaction Time With Sent Event	BRMS Inference
August 28, 2014, Friday 10:00	August 28 2014, Friday 10:02	Event is sent in weekday business hours, the subscriber interacts with the event message in business hours, too. No new time profile assignment for the subscriber.
August 28, 2014, Friday 10:00	August 28, 2014, Friday 20:37	Event is sent in weekday business hours, the subscriber interacts with the event message in out of business hours. It is learnt that the subscriber is more interactive in out of business hours. Then, set time profile as non business hours in order to send messages in out of business hours, and in order not to make the subscriber miss the notifications in business hours.
August 28, 2014, Friday 10:00	August 29, 2014, Saturday 14:07	Event sent in weekday business hours, the subscriber interacts with the event message at weekend. Then, set time profile as weekend in order to send messages in weekends, and in order not to make the subscriber miss the notifications in weekdays.

All BRMS inferences depend on defined business rule sets. Therefore, the inferences and actions of BRMS are allowed to modify by changing the rules or by adding new rules. For example, time context perception of BRMS is allowed to change by adding new categories into hour or day categories. Moreover, weather data inference can be also changed. For example, in snowy weather conditions, the weather inference could be set to “Good” while the inference was “Bad” before. Furthermore, by means of BRMS, the context data is not limited with time, location and weather info. Various context data could be evaluated by aggregating the other context data.

BRMS sub module has one connection. The sub module communicates with the context sources to gather all related data from external or internal context sources, in order to evaluate the conditions and run the rules.

Event management server has three connections; these are with, event publisher, context sources and event subscriber. Firstly, the event management server has communication with the subscriber to present the topics to be subscribed and to deliver the event messages via event dispatcher module. Secondly, event management server connects to the context sources to use the context data in BRMS for evaluation of the conditions. Finally, it has connection with the event publisher that we talked about in the event publisher part.

3.2.3. Context Sources

We propose internal or external context sources to define the existing situation better. From context sources, the system receives extra information to return it to valuable knowledge. It is done by BRMS sub module that communicates with the context sources and that gathers the context data as input to start new inferences. Context sources in our conceptual design will be time data, location and weather information.

3.2.4. Event Subscriber

Event subscriber can subscribe to the topics that the event management system provides. The subscriber can select the topics that may attract the subscriber in order to receive related event recommendations. Furthermore, the activities of the subscriber are tracked by the event management system, so that event management server could maintain context-awareness and the event management server is able to keep presenting related events to the subscriber. Event subscriber has one connection with event management system in order to receive the related events and topics.

In our conceptual design, event subscriber has chance to modify its preferences and interests (tags) so that it will receive more customized notifications. Firstly, the subscriber may add new interests or modify its interests about the events. We can give a sample list of some user interests about the event that the subscriber may select or deselect:

- Music
    - Pop
    - Rock
    - Jazz
    - R&B
- Movie
    - Action
    - Drama
    - Science Fiction
    - Romance
    - Comedy
- Sport
    - Football
    - Basketball
    - Tennis
    - Golf
    - Volleyball

- Shopping
    - Clothing
    - Electronics
    - Food
    - Drink

By selecting the interests like above tag list, the subscriber filters the event messages that are only interesting to it. This main feature of publish-subscribe approach provides customized messages for every single subscriber and avoids spam event messaging. The subscriber is able to change its interests both in the system beginning and later. Moreover, the subscriber can set message delivery preferences that depend on time, location and weather forecast data.

Firstly, the subscriber has the opportunity to add or remove location dependent event notification. For instance, the subscriber may want to receive events only in Ankara where it is located. Or it can remove location dependency to receive event notifications that are in or not in the location of the subscriber. If the subscriber sets location dependent notification, the subscriber will have the event messages in a location, when it enters the specified location. In other words, the system triggers sending geographically related event, when the subscriber enters into the event neighborhood.

Secondly, the subscriber can utilize weather sensitivity of the system to receive better event notifications. The event management system is able to receive current and future weather forecast data for a specified location so that the system will have better recommendation skills dependent to weather forecast. By means of weather context data, the subscriber will have the opportunity to be avoided in bad weather situations for outdoor events. The system will not suggest an event that is an outdoor activity and in bad weather circumstances. The subscriber may turn off the weather sensitive notification so that it can receive all events related to the subscriber without considering weather context data.

Finally, time awareness could be significant for a context-aware system. In our conceptual design, BRMS module has the ability to be aware of time notion. The subscriber has an option to receive messages according to its specified time preferences. For example, the subscriber may receive the event message in a specified hour interval, such as everyday in out of business hours by setting a time profile. Furthermore, it can receive the messages in a day range, such as on weekends. In addition to manual setting of the subscriber, the BRMS can automatically assign a time profile for the subscriber, according to its activity time with the event notifications. For instance, BRMS can have the knowledge about that the subscriber interacts with the system in out of business hours, so BRMS may set a time profile for the subscriber to send the messages in only out of business hours. BRMS achieves this job by getting the information of the interaction time (the time of opening the event message) of the subscriber, then compares the interaction time with the event creation time, finally, BRMS could set a time profile according to the comparison. Of course, the subscriber may change its time profile manually.

CHAPTER 4

IMPLEMENTATION

As a prototype application of the proposed system, an Android based mobile application is implemented. While developing the application, the architecture of the proposed system is taken into account, so the implementation architecture is parallel with the proposed system structure. Before explaining the implementation architecture, used technologies and services in the developed applications are described, then the architecture is explained in detail. Finally, sample usage of the system and sample use cases are mentioned in order to express the system and its features better.

4.1. Used Technologies and Services

4.1.1. REST architecture / Restful Web Services

REST (Representational State Transfer) is a web architecture that uses http requests to easily connect client and server. We can simply use GET, POST, PUT or DELETE http methods and get the response over json format. Additional overheads in SOAP [26] architecture are not found in REST, such as the Envelope, which specifies what is in the message and how to process it, encoding rules for data types, layout procedure calls. Since the REST is lightweight, the architecture could be suitable for mobile devices, in terms of low CPU effort and low battery consumption. Another property of REST is that we can access resources via URI. For example, if we want to get item with id 15 we just use GET HTTP request by writing the URI: http://example.com/resource/item/15, than we could get the item in json format (or other Internet media types like XML, Atom etc.). To sum up, we can list the features of the REST architecture;

- Client-server
- Stateless: Independent pairs of request and response. Previous request has no relation with the further requests.
- Cacheable
- Layered System
- Code on demand: Servers might customize the functionality of a client by transmission of the executable code.
- Uniform interface

4.1.2. ASP.NET Web API

ASP.NET Web API [29] is a framework that uses REST architecture, and eases to create HTTP services for many types of clients such as browsers and mobile devices. In our system, we use ASP.NET Web API with C# programming language to construct our main application server (event management server).

4.1.3. Microsoft Visual Studio 2013 and Azure Cloud Services

In implementation of our application server, we use Visual Studio 2013[30] Ultimate platform for programming with C#. For easy and fast deployment we use Microsoft Azure [31] cloud services.

4.1.4. Google Cloud Messaging Push Service

GCM [32], Google Cloud Messaging, is a Google service that provides developers to send message from servers to Android applications. For push notifications, we use GCM push service.

4.1.5. Drools - Business Rule Management System

Drools [33], is an open source Java based business rule management system provided by JBoss [21], used in business rule engine server for our proposed system.

4.1.6. Java, Android and Eclipse

For mobile application development (client side), we use Java programming language and Android [34] development with Eclipse [37] platform.

4.1.7. DBpedia & DBpedia Lookup Service

DBpedia [35] is a platform that provides extracted structured information from Wikipedia. It also allows users to make complex queries from Wikipedia. DBpedia Lookup Service is used to look up DBpedia URIs by related keywords. We use this service in our system to utilize as linked data feature. For example, when we look up for ‘metu’ word, the results and some parsed linked data are shown below.

- Organization
- Educational institution
- Agent
- University
- Educational institutions established in 1956
- Education in Ankara
- State universities and colleges in Turkey
- Technical universities and colleges
- Middle East Technical University

4.1.8. OpenWeatherMap API

OpenWeatherMap [17] API is both free and paid service for the developers to integrate weather info into their apps. The developer can easily get up to 16-days weather information for a selected city via OpenWeatherMap API. In our system, we utilize OpenWeatherMap API to use weather data as context source.

4.2. Architecture Overview of the Implemented System

As a concrete output of our conceptual design we provide a real system, as the architecture is shown below.

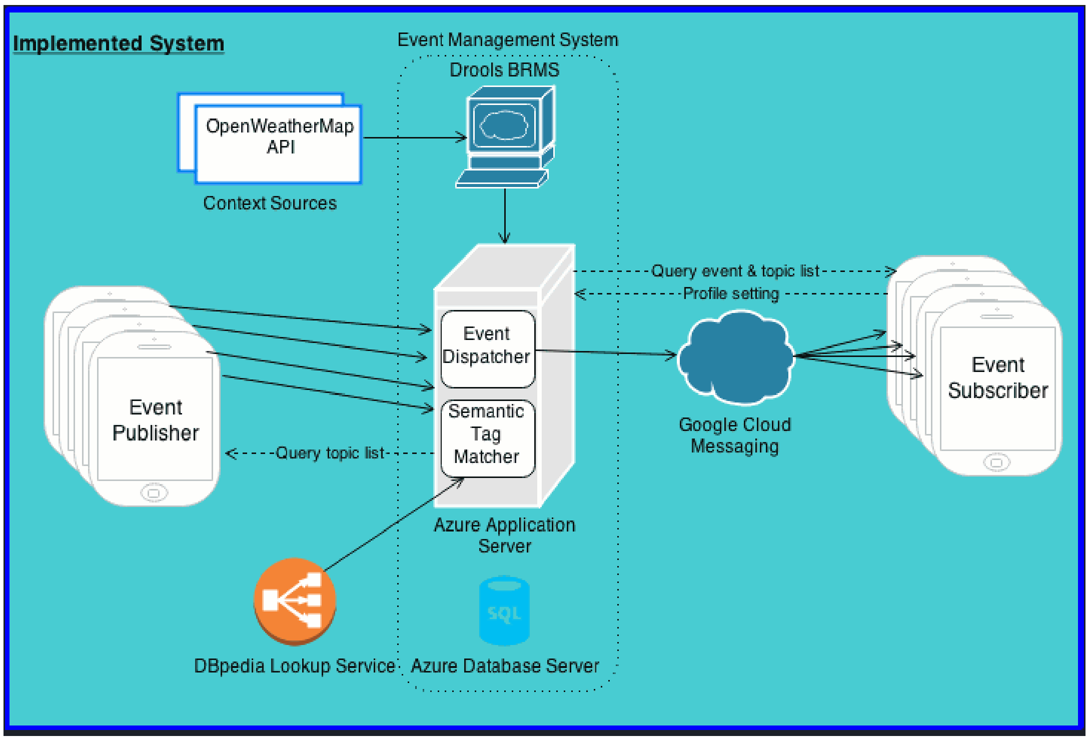


Figure 6 Architecture of Proposed Implementation

As we describe the conceptual design before, we can see reflection architecture of the implementation in Figure 6. We have Event Publisher working on Android device, Event Subscriber working on Android device, and Event Management System that composes of Azure Application Server, Azure Database Server and Drools BRMS. As external services and sources, we utilize DBpedia Lookup Service for Semantic Tag Matcher so as to implement linked data method for related tag matching. We described DBpedia dataset and its Lookup Service before. Drools BRMS uses Context Sources to receive context data. In our implementation, we utilize OpenWeatherMap API to retrieve weather data. Moreover, BRMS uses location data of the subscriber and time of day info. In order to deliver messages to subscriber Android devices, we utilize Google Cloud Messaging service on Event Dispatcher.

The events, event details, existing all user interests (tags) are stored in Azure application server with Azure Database Server. The publishers create events by specifying the tags related to the events via Android user interface. The subscriber Android mobile client can enter the system and specify its interests and view the events. Drools BRMS receives the data from the context sources such as weather data from OpenWeatherMap API externally, location and time data internally, then creates valuable knowledge for the event recommendation and waits for the best time of message delivery. Semantic tag matcher deduces semantic inferences for new user interest discovery by linked data method via DBpedia lookup service.

In our proposed system, we aim sending right information to correct person at the right conditions, while considering rich context construction by utilizing improved internal and external context services. For rich context construction and efficient information delivery, we bring together; time, weather, place, user history and user profile information criteria with smart services, semantic environment, business rule management system.

For related context, the backbone of our system is the tags, in other words, the user interests. To find related content with the user, we mainly utilize user interest and process jobs using tags. In Figure 7, the user can arrange his/her interests then receives notifications according to them. Furthermore, the subscriber can set her/his message delivery preferences according to time, place and weather forecast information so that the message will be sent by considering time, place of the subscriber and weather information in event date. The publisher sets the tags while he/she is creating an event, but he/she might not have enough knowledge to set all related tags; even he/she might not specify any tags. At this point, we propose a helper solution that we discuss in semantic tag matcher.

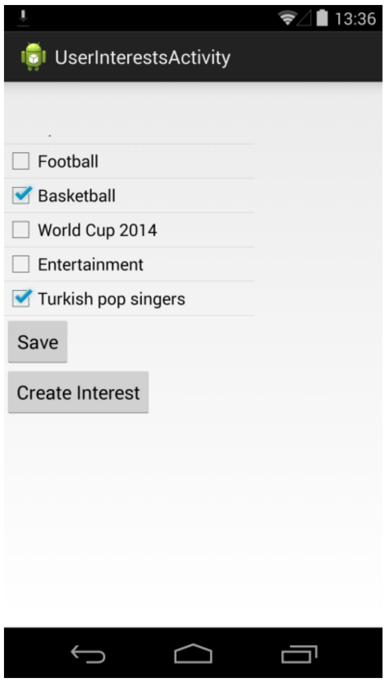


Figure 7 Interests can be specified in this screen

4.2.1. Event Management System

Event management system composes of three sub systems, Azure application server, Azure database server and Drools BRMS.

Azure Application Server is one of the main components responsible from event publishing that has the connections with Drools BRMS, database, publisher and the subscriber. The user information, topics that are called tags i.e. user interests, and the user histories are hold in this event management system via Azure database server. The communications with the other components are done via suitable sub modules inside the event management system. Firstly, for event message delivery, the event management system uses the message dispatcher sub module in Azure application server to send messages to registered subscribers via Google Cloud Messaging service. Secondly, in order to use context sources to check conditions, process events and trigger message sending, the system utilizes Drools BRMS. Thirdly, so as to infer semantic relations for event name to match more tags, semantic tag matcher uses linked data method by connecting DBpedia Lookup Service. Moreover, before connecting to the DBpedia Lookup Service, word-filtering algorithm, which is working in semantic tag matcher, helps refining the word in order get more successful results coming from DBpedia Lookup Service. Finally, Azure Application Server is built on Web API to interact with the subscriber and publisher Android Mobile Clients via REST framework. We will talk about semantic matcher in part 4.2.2., and we will talk about Drools BRMS in part 4.2.4. in detail.

4.2.2. Semantic Tag Matcher via DBpedia Lookup Service

In order to make semantic contextual inferences, we use linked data [15] method in semantic tag matcher. In content delivery, the aim is trying to reach highest number of subscriber who may be interested in the event. While an event is created, the publisher might not add the related tags to the event. At this point, semantic tag matcher searches for linked words with the specified word in the event title to match the related words to the tags. For that purpose, the system utilizes DBpedia [16] to find related words. For example, if a publisher creates an event named "William Shakespeare", the semantic tag matcher connects to lookup service [17] of DBpedia and look up categories and classes of the word as related words.

Some linked data are listed below for William Shakespeare coming from DBpedia as the example is visually explained in conceptual design part.

- person
- artist
- writer
- Sonneteers
- 1564 births
- English dramatists and playwrights
- 16th-century English people
- People of the Tudor period
- English poets
- English Renaissance dramatists

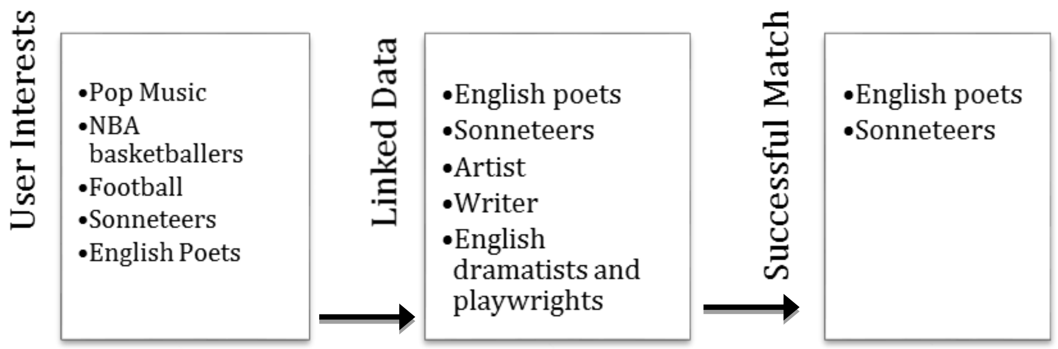


Figure 8 Linked Data-User Interests Successful Match

This kind of output could create a value that; there is no need for exact word matching for the specified tag. There are some alternatives like these related words to help find interest matching. An example may clarify the situation, a subscriber is interested in sonneteers and he/she adds “English Poets” and “Sonneteers” to his/her interests via his/her Android device. Another person as publisher creates an event titled “William Shakespeare Seminar Event” without specifying any tags related to the event. Semantic tag matcher connects to the DBpedia and receives the XML file about William Shakespeare, parses the classes and categories about William Shakespeare, finds “English Poets” and “Sonneteers” classes, and binds the “English Poets” and “Sonneteers” existing tags with this event. Finally, the system is able to notify the user who is interested in sonneteers and English poets, as shown in Figure 8. The benefit of this feature is providing automated tag matching and reaching more subscribers who may be interested. Moreover, for healthier keyword lookup, a basic word filtering mechanism works in our system. There are some reserved words to filter event name, for example, “event, party, festival, night, of, the, movie, seminar” words are discarded for refined search. Thus, when we type “William Shakespeare Seminar Event”, the filtering mechanism works and removes “Event” and “Seminar” words, then connects to DBpedia with “William Shakespeare” in order to get linked words about William Shakespeare. The word filtering mechanism, as the examples are shown in Figure 9 and Figure 10, is a sub module in semantic tag matcher, and it could be improved in the future.

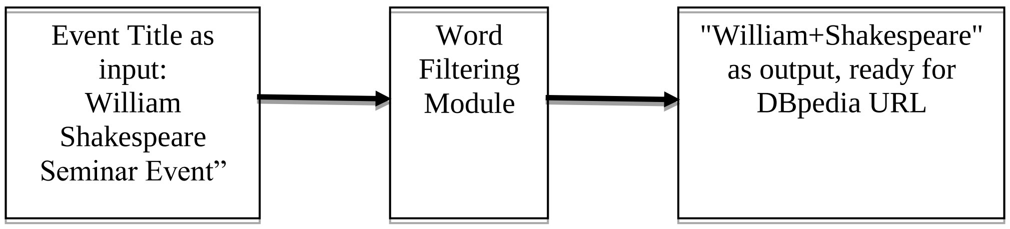

Event Title as input: William Shakespeare Seminar Event”
Word Filtering Module
"William+Shakespeare" as output, ready for DBpedia URL

Figure 9 Word Filtering Module first example

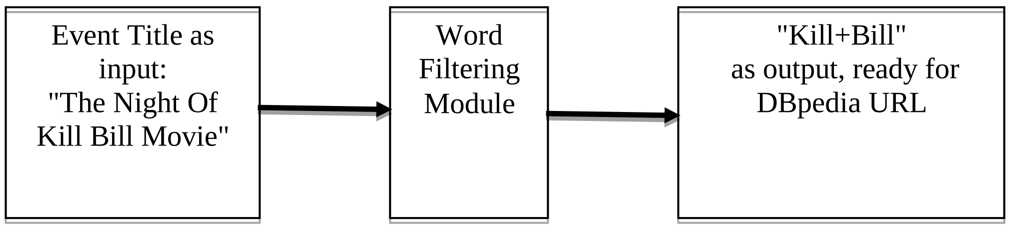

Event Title as input: "The Night Of Kill Bill Movie"
Word Filtering Module
"Kill+Bill" as output, ready for DBpedia URL

Figure 10 Word filtering module second example

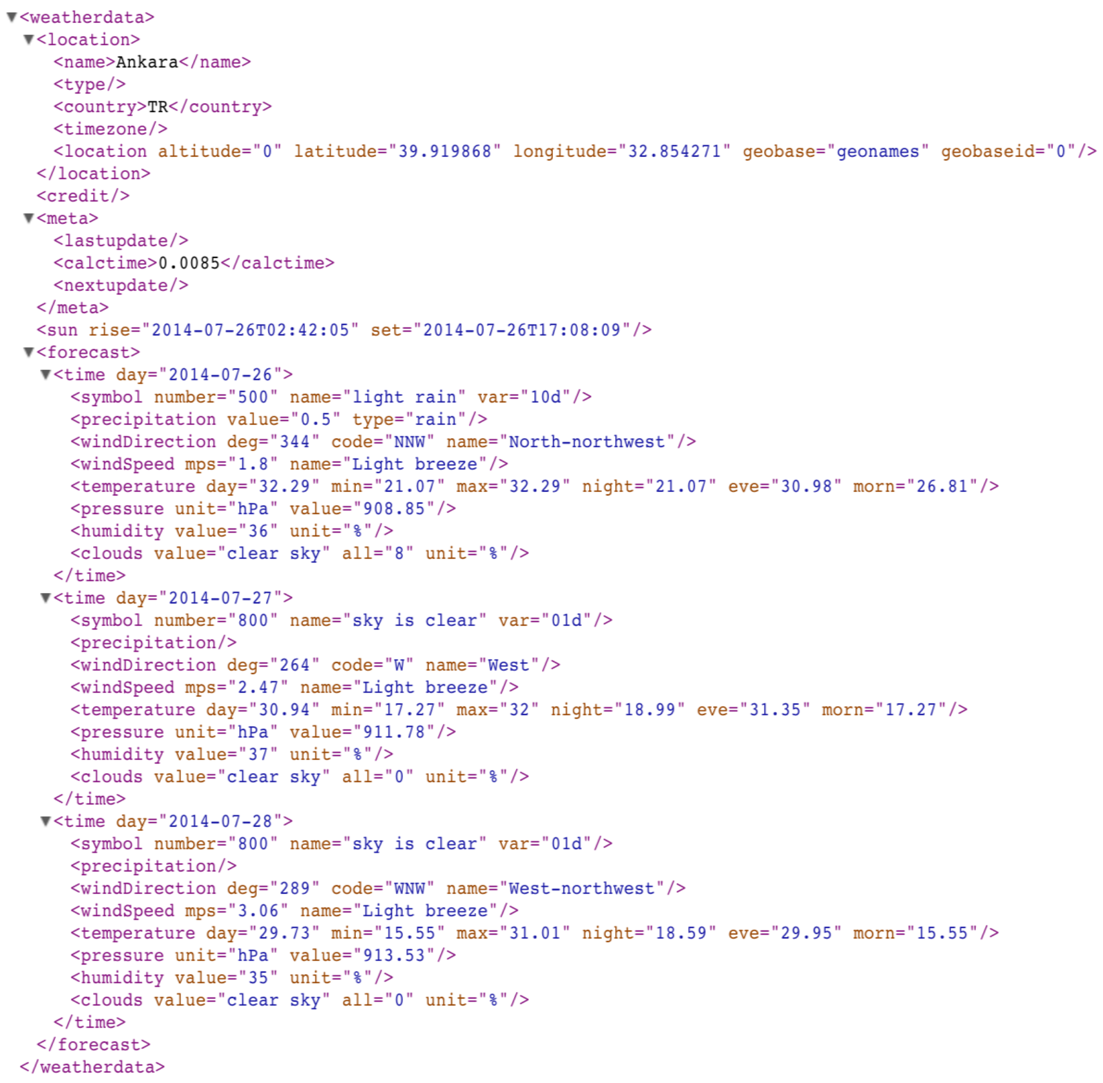
4.2.3. Context Sources

As context source, we utilize time of data to perceive time. The time data is resided in Drools BRMS. There are some predefined calendars in BRMS to become aware of time concept, which we will talk about in Drools BRMS part.

Secondly, we use weather context data for our prototype. In context-aware event recommender and message delivery systems, weather information could be very significant especially for outdoor events. For instance, in rainy situations, an outdoor event might not be that enjoyable. For that purpose, we add weather forecast context in our system by utilizing OpenWeatherMap API [17] to receive current and future weather data according to where the event will be. OpenWeatherMap presents a free service for developers to integrate weather information to their applications. With this service, we basically type the URL with the city name, or GPS information (longitude and latitude), and then receive current weather data and also up to 16-days forecast information in JSON or XML format. For more readability, we can give an example for 3-days forecast of Ankara.

URL for 3-days weather forecast of Ankara:
http://api.openweathermap.org/data/2.5/forecast/daily?q=ankara&mode=xml&units=metric&cnt=7

The output XML file is in Figure 11 below:



```
▼<weatherdata>
  ▼<location>
     <name>Ankara</name>
     <type/>
     <country>TR</country>
     <timezone/>
     <location altitude="0" latitude="39.919868" longitude="32.854271" geobase="geonames" geobaseid="0"/>
   </location>
   <credit/>
  ▼<meta>
     <lastupdate/>
     <calctime>0.0085</calctime>
     <nextupdate/>
   </meta>
   <sun rise="2014-07-26T02:42:05" set="2014-07-26T17:08:09"/>
  ▼<forecast>
    ▼<time day="2014-07-26">
       <symbol number="500" name="light rain" var="10d"/>
       <precipitation value="0.5" type="rain"/>
       <windDirection deg="344" code="NNW" name="North-northwest"/>
       <windSpeed mps="1.8" name="Light breeze"/>
       <temperature day="32.29" min="21.07" max="32.29" night="21.07" eve="30.98" morn="26.81"/>
       <pressure unit="hPa" value="908.85"/>
       <humidity value="36" unit="%"/>
       <clouds value="clear sky" all="8" unit="%"/>
     </time>
    ▼<time day="2014-07-27">
       <symbol number="800" name="sky is clear" var="01d"/>
       <precipitation/>
       <windDirection deg="264" code="W" name="West"/>
       <windSpeed mps="2.47" name="Light breeze"/>
       <temperature day="30.94" min="17.27" max="32" night="18.99" eve="31.35" morn="17.27"/>
       <pressure unit="hPa" value="911.78"/>
       <humidity value="37" unit="%"/>
       <clouds value="clear sky" all="0" unit="%"/>
     </time>
    ▼<time day="2014-07-28">
       <symbol number="800" name="sky is clear" var="01d"/>
       <precipitation/>
       <windDirection deg="289" code="WNW" name="West-northwest"/>
       <windSpeed mps="3.06" name="Light breeze"/>
       <temperature day="29.73" min="15.55" max="31.01" night="18.59" eve="29.95" morn="15.55"/>
       <pressure unit="hPa" value="913.53"/>
       <humidity value="35" unit="%"/>
       <clouds value="clear sky" all="0" unit="%"/>
     </time>
   </forecast>
 </weatherdata>
```

Figure 11 3-days weather forecasts for Ankara [18]

In addition to temperature value, we consider the weather condition parameters [19] in our system:

- clear sky
- few clouds
- scattered clouds
- broken clouds
- shower rain
- rain

- thunderstorm
- snow
- mist

Drools BRMS uses these parameters to make healthier recommendations and healthier information delivery by regarding weather info as context. For instance, the system may discourage the user to attend while presenting an outdoor event with a rainy weather data, by showing a warning message about the situation. This condition can be scaled with the BRMS that we will talk about below.

Thirdly, we utilize location context data of the subscriber. Drools BRMS use location info about the subscriber and determine triggering of message delivery. In our prototype, we retrieve direct location input of the subscriber instead of using GPS and other wireless sensors.

4.2.4. Drools BRMS

Business rules are another factor that could create value in context-aware systems. We integrate Drools BRMS [20] into our system. Drools is open-source, Java programming language based business rule engine and enterprise framework for the construction and maintenance of business logics in an organization, application or service supported by JBoss [21]. In addition to business rule engine, Drools has the feature of complex event processing (CEP [22]). CEP is an event processing that aggregates data from multiple resources to deduce patterns and events that propose complicated circumstances. The aim of CEP is to identify significant events and react accordingly to them quickly. Therefore, Drools BRMS helps to find most suitable conditions to deliver message specific to the subscriber.

In Drools platform, we can define business rules by adding conditions (like if-else conditions) and time-based conditions. For our system, we can define time rules to run the rule only in the selected time interval or specified future time. For instance, we create some time intervals named calendars such as “business-hours” and “non-business-hours” calendars. (Figure 12 and Figure 13)

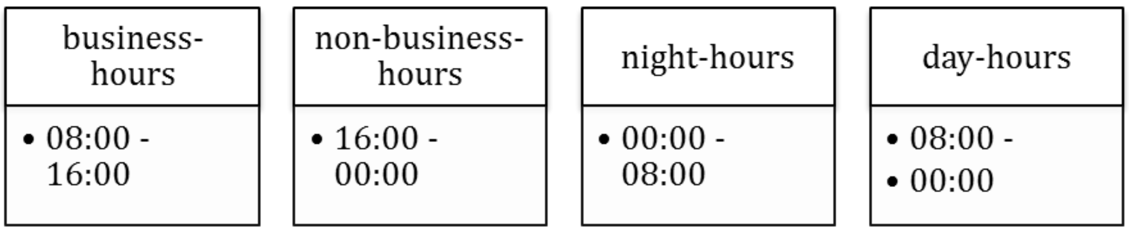


Figure 12 Daily Calendars

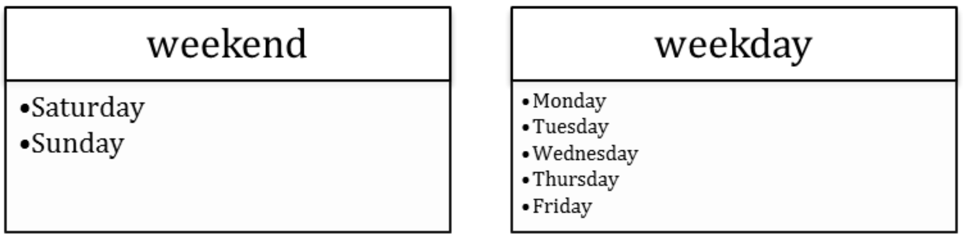
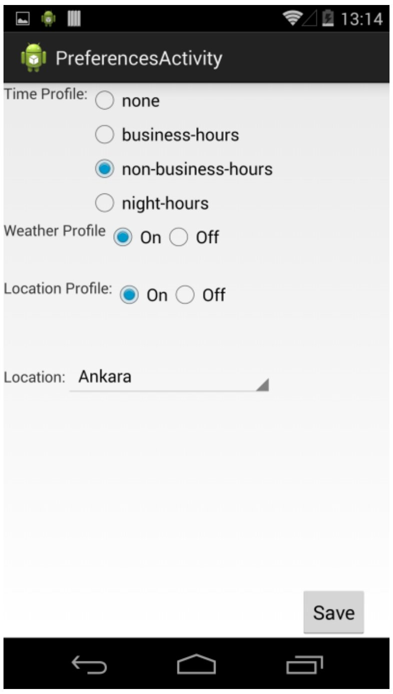


Figure 13 Weekly Calendars

We use these calendars in order to evaluate subscriber time preferences. The subscriber may want to receive notifications in certain time intervals such as in business hours, out of business hours, night hours. You can see below figure that displays subscriber time preferences.


PreferencesActivity

Time Profile: ( ) none ( ) business-hours [✓ non-business-hours] ( ) night-hours

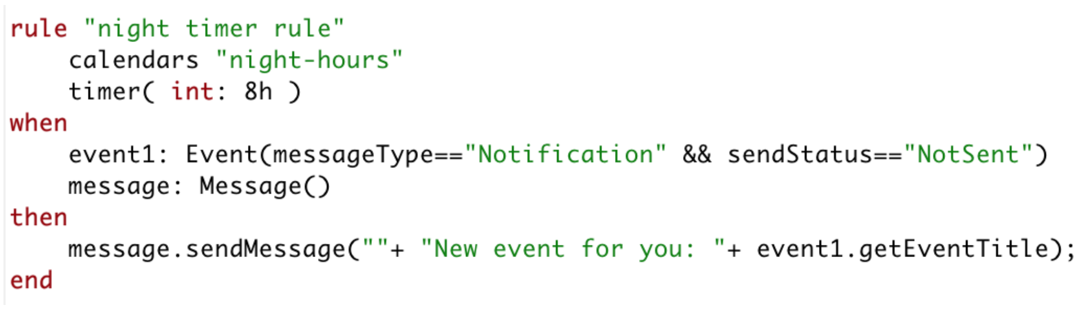
Weather Profile [✓ On] ( ) Off

Location Profile: [✓ On] ( ) Off

Location: Ankara

Figure 14 Subscriber Preferences Screen

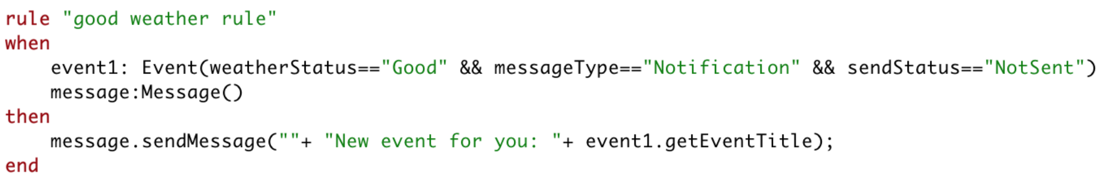
If we integrate one of the calendars into a rule depicted in Figure 12 and Figure 13, the rule will work only in the specified times. The calendars can be any time interval that can be defined, such as, business hours between 08:00 and 16:00 (as shown in Figure 14), or a day interval such as, weekends. Furthermore, we define timers to trigger actions, when single or multiple conditions occur. For example, we can run the rule action after 8 hours, or in a specified date such as 10/10/2014 09:00. Additionally, we can run a rule periodically, such as, every day at 13:00. These are time-based conditions. We can also define case based conditions. Weather data could be integrated into rules to suggest events. For instance, if an outdoor event is created for a rainy day (the weather information is taken from OpenWeatherMap API), the rule engine works and gives an alert to not suggest this event. Furthermore, if a person attends to event A and event B, by means of rule engine, the system offers event C. The rule engine can be dynamically changed and modify its behavior by adding new rules into our system, by allowing to upload a DRL file. The benefit of this process is to make the system work fluently without deploying new code. Thus, without maintenance effort, the system provides new rules that can be added later.



```
rule "night timer rule"
    calendars "night-hours"
    timer( int: 8h )
when
    event1: Event(messageType=="Notification" && sendStatus=="NotSent")
    message: Message()
then
    message.sendMessage(""+ "New event for you: "+ event1.getEventTitle);
end
```

Figure 15 Night Timer Rule that works only night hours

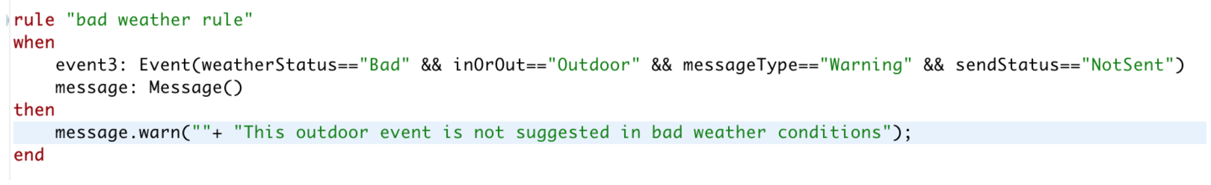
In timer based rules, in the rule definition phase, one of the specified calendars could be selected, such as, night-hours, business-hours, weekends or weekdays, to define the rule in the selected time interval without affecting any other time range. This arrangement provides independent rules in different times. Figure 15 shows a time-based rule. The person who defines the rules, specified “night-hours” calendar, meaning that, the time range is between 00:00 and 08:00, and he/she sets a timer to run the rule after 8 hours (timer (int: 8 h)). It means running the rule after the “night-hours” calendar time is passed. The successful scenario in Figure 15 is that, if an event is created in the night-hours, wait for passing of night hours, and then send event information to the user.



```
rule "good weather rule"
when
    event1: Event(weatherStatus=="Good" && messageType=="Notification" && sendStatus=="NotSent")
    message:Message()
then
    message.sendMessage(""+ "New event for you: "+ event1.getEventTitle);
end
```

Figure 16 Good Weather Rule

We can add context data based condition rules such as weather context. Figure 16 shows an example of sending an event notification in good weather circumstances. In here, all fine weather situations are abstracted as “Good” in the specified time from the server before, then the rule only checks the whether the conditions is good or bad. The weather parameters clear sky, few clouds, scattered clouds, broken clouds are said to be as “Good”, and shower rain, rain, thunderstorm, snow, mist weather parameters are said to be as “Bad” from the server. Of course, specific weather parameters could be also added to the business rules without abstraction.



```
rule "bad weather rule"
when
    event3: Event(weatherStatus=="Bad" && inOrOut=="Outdoor" && messageType=="Warning" && sendStatus=="NotSent")
    message: Message()
then
    message.warn(""+ "This outdoor event is not suggested in bad weather conditions");
end
```

Figure 17 Bad Weather Rule

Figure 17 shows a bad weather warning example. If an outdoor event is created and the weather is bad during the event time, the server sends a warning message to the user, while he/she is viewing the specified event.

Finally, we can say that new rules can be defined later into our system without deploying new code or without having to change whole system, since the rule framework is based on a modifiable .drl file. Furthermore, Drools BRMS has the opportunity to be modified visually by the utilization of Drools Guvnor [24] service of Drools mechanism, in order to ease business rule creation for non-developer people.

In our system, there are two system roles: Standard User as Subscriber, Event Notifier as Publisher. Standard User is the subscriber role of the system. He/she can specify his/her interests, create new interests (tags), and view the events that may be interesting. He/she can be notified and view the push messages when an event, which might be interesting, is created. Event Notifier is the publisher role of the system that can create tags, create events with related tags. Event Notifier could be any character such as a restaurant/cafe manager, restaurant/cafe customer, student, academician, hotel receptionist or even a person who walks on the street, who wants to notify the subscribers who may be interested in the specified event/place. For rich content creation, there is no constraint to be an event notifier but registration and authorization phase, so that anybody could create an event in order to notify other users.

4.2.5. Event Publisher Android Client

We implement event publisher Android client responsible from event creation with attached from Android user interface. In our implementation, the publisher must fill these fields as in Figure 23:

- Event Title: Name of the event.
- Start Time: Start date and time of the event.
- Finish Time: Finish date and time of the event.
- Description: Extra information about the event.
- Location: Location of the event.
- Event Type: Indoor or outdoor event info.
- Tags: Related tags of the event.

4.2.6. Event Subscriber Android Client

We implement event subscriber Android client responsible from setting preferences, subscribing to user interests, viewing events and registering into events from Android user interface.

Event subscriber can modify his/her interests in order to retrieve related notification and to view related events. Moreover, event subscriber has the opportunity to set his/her preferences so as to get notification in appropriate times for the subscriber. We define 3 types of message delivery preferences as in Figure 14.

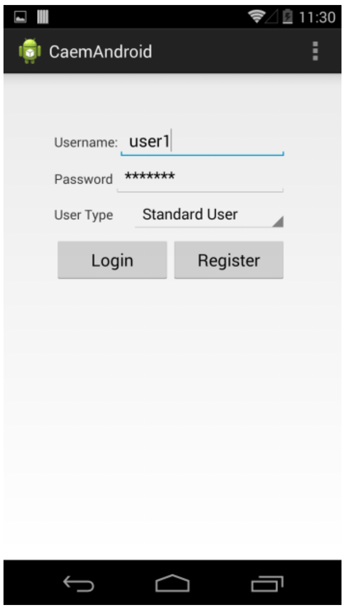
- Time preference: The subscriber can receive the notification in specified time intervals.
    - none: The subscriber can receive the message without any time interval constraint. He/she receives the message whenever the related event is created.
    - business-hours: The subscriber can only receive the notifications in business hours. (Default interval: 08:00-16:00)
    - non-business-hours: The subscriber can only receive the notifications in out of business hours.(Default interval: 16:00-00:00)

    - night-hours: The subscriber can only receive the notifications in night hours. (Default interval: 00:00-08:00)
- Location preference: The subscriber can receive the event notifications whose event place is only in the neighborhood of the subscriber. Or the subscriber may turn off this preference in order to receive the event notifications whose event place could be anywhere.
- Weather preference: The subscriber can receive the event notifications only if the weather is good in the event date. The subscriber has the option to turn off this preference in order to receive the event info without checking the weather data.

4.3. Sample Usage of the System

4.3.1. Interest Setting

1. User logins to system as Standard User (Subscriber).



Username: user1
Password *******
User Type Standard User

Figure 18 Login as Standard User

2. After successful login, the user taps on the “Your Interests” button.

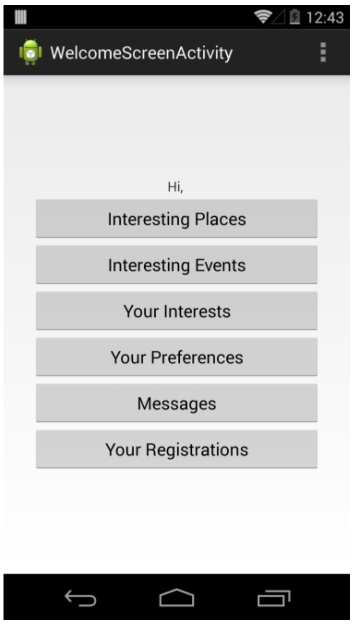


Figure 19 Standard user main screen

3. User selects the interests “Turkish pop singers”, “Football” and “Action Movies”, and taps to “Save” button then exits. Now, the user will have chance to receive related event notifications.

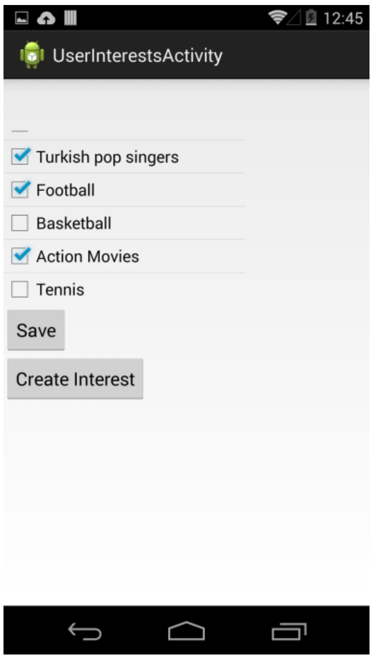
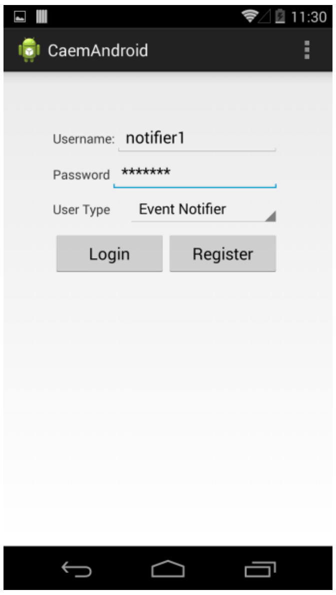


Figure 20 Interest setting

4.3.2. Event Creation and Message Delivery

1. User logins to system with the role of Event Notifier (Publisher).



Username: notifier1
Password *******
User Type Event Notifier

Figure 21 Login as Event Notifier

2. User taps on the "Create Event" button.

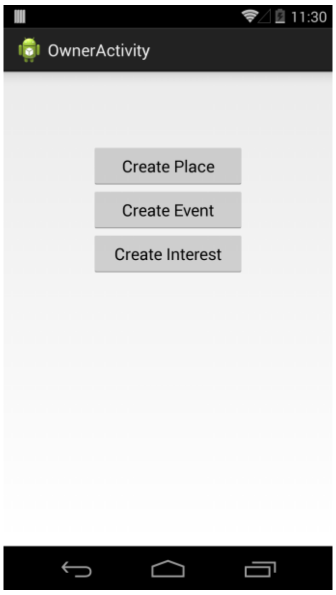
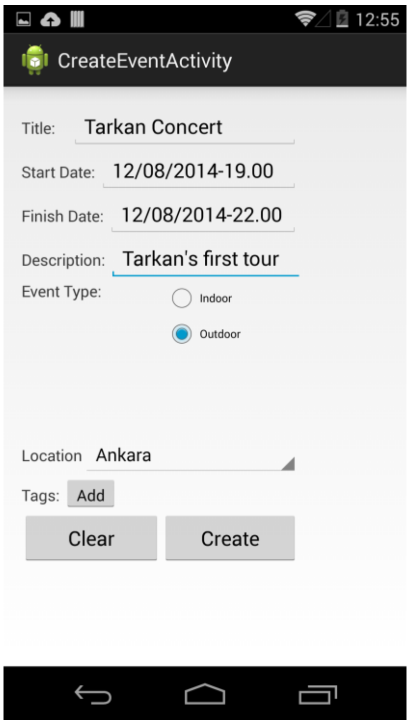


Figure 22 Main Screen of Event Notifier

3. User fills the form specifying title as "Tarkan Concert", event type as "Outdoor", start date as "12/08/2014 19:00", finish date as "12/08/2014 22:00", description as "Tarkan's first tour", Location as "Ankara", and the tag as "Turkish pop singers". Then creates the event.


12:55

CreateEventActivity

Title: Tarkan Concert

Start Date: 12/08/2014-19.00

Finish Date: 12/08/2014-22.00

Description: Tarkan's first tour

Event Type: Indoor [✓ Outdoor]

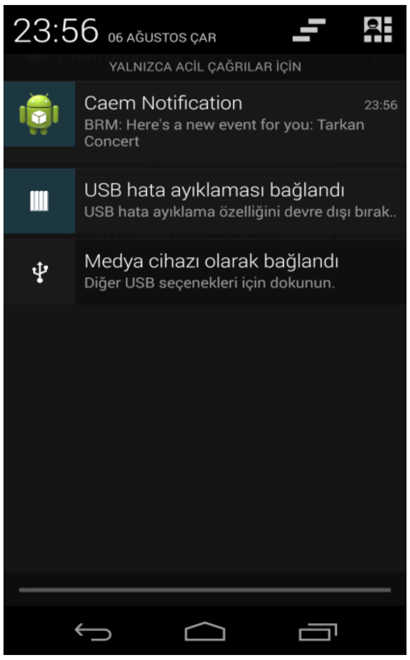
Location Ankara

Tags: Add

Clear Create

Figure 23 Event creation

4. After creating the event, the system finds the related users and pushes the event information to the subscriber who registered to "Turkish pop singers". We can see the event message (with title Caem Notification) in a subscriber's mobile phone. (BRM: Business rule engine message)


23:56 06 AĞUSTOS ÇAR

YALNIZCA ACİL ÇAĞRILAR İÇİN

Caem Notification 23:56
BRM: Here's a new event for you: Tarkan Concert

USB hata ayıklaması bağlandı
USB hata ayıklama özelliğini devre dışı bırak..

Medya cihazı olarak bağlandı
Diğer USB seçenekleri için dokunun.

Figure 24 BRM Notification

4.3.3. Event Detail Display, Registering to Event and Setting Time Profile

1. The user logs into system as the role of Standard User. After successful login, the user taps on the “Interesting Events” button in order to view the related events.

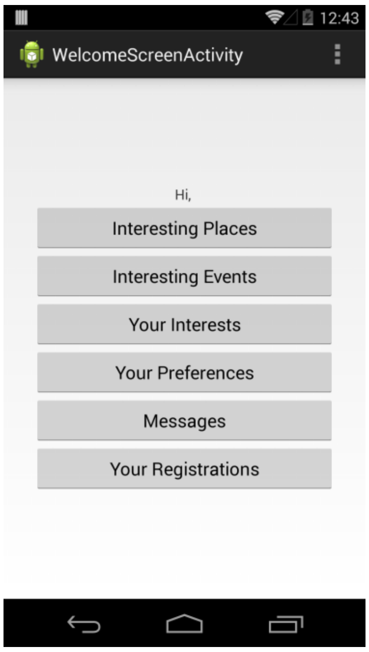
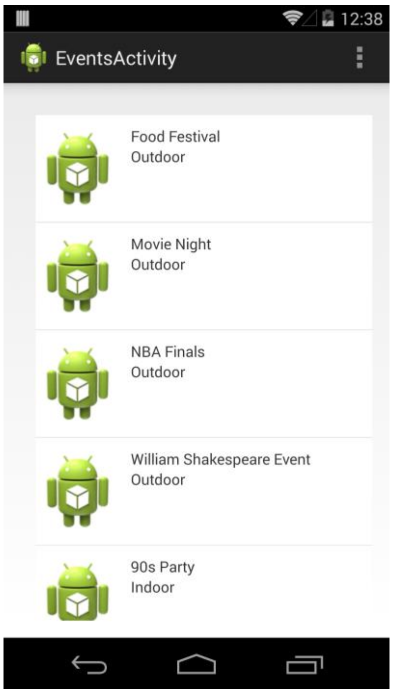


Figure 25 Standard User main screen

2. The user views the listed events and selects an event to view the details.



Food Festival
Outdoor

Movie Night
Outdoor

NBA Finals
Outdoor

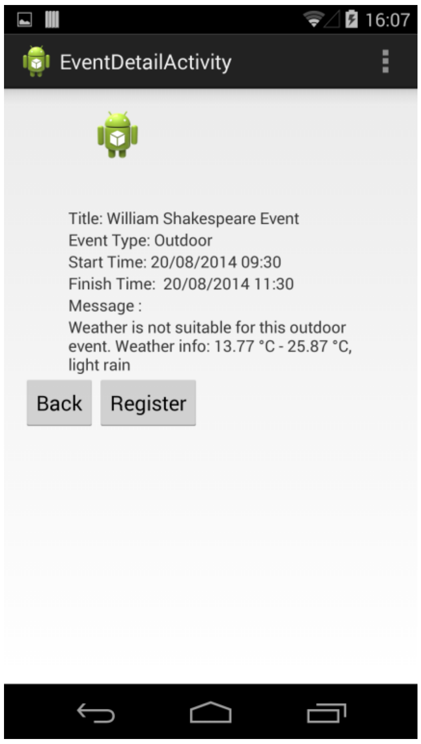
William Shakespeare Event
Outdoor

90s Party
Indoor

Figure 26 Suggested events

3. In the next screen, the system gathers all information about the event. It checks the weather information sends the weather info to the business rule engine. The engine assesses the conditions and sends a message to application server. The application

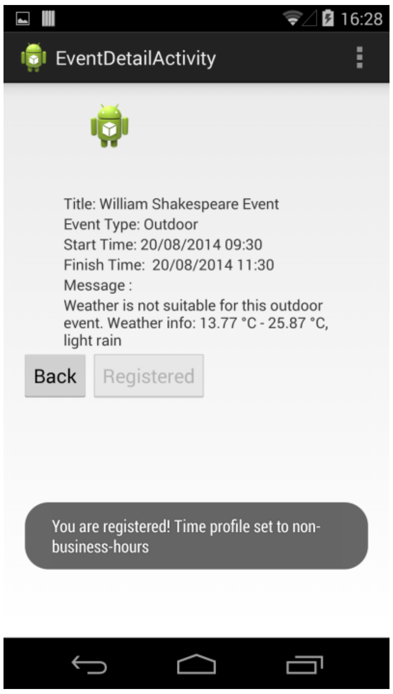
server sends a message about the event. In here, the message from the business rule engine is that the weather will not be suitable in the event date.



Title: William Shakespeare Event
Event Type: Outdoor
Start Time: 20/08/2014 09:30
Finish Time: 20/08/2014 11:30
Message :
Weather is not suitable for this outdoor event. Weather info: 13.77 °C - 25.87 °C, light rain

Figure 27 Event description with message

4. After viewing the event information and the message, the subscriber finds the event interesting and registers to the event anyway, even if the system does not suggest the event. The system checks the conditions and the interaction time of the user. The event was actually created in business hours, but the user interacts with the event in out of business hours. System infers the knowledge that the user is more active in out of business hours, then, sets time profile of the user to “non-business-hours”, in order to send the messages only in out of business hours. The subscriber may change his/her profile later if s/he wants to.



Title: William Shakespeare Event
Event Type: Outdoor
Start Time: 20/08/2014 09:30
Finish Time: 20/08/2014 11:30
Message :
Weather is not suitable for this outdoor event. Weather info: 13.77 °C - 25.87 °C, light rain

You are registered! Time profile set to non-business-hours

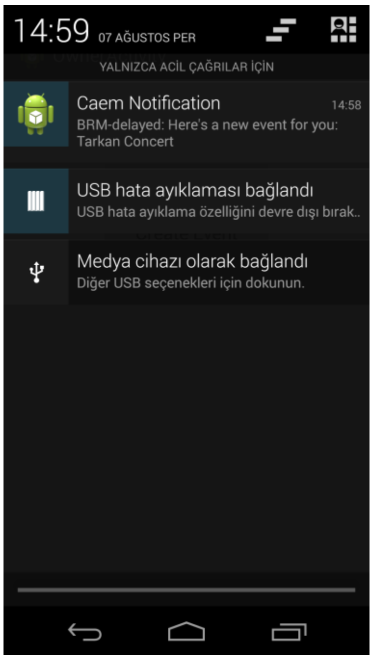
Figure 28 Time profile set to non-business-hours

4.3.4. Message Delivery By Means Of Linked Data

When an event notifier adds an event information to the system without specifying related tags, the system connects to DBpedia lookup service and tries to match the tags with the received categories and classes of lookup service. For example, there is an event created, named as “Tarkan Concert”. However, there is no tag specified about Tarkan Concert event. In that case, the system connects to the DBpedia lookup service and receives “Turkish pop singers” category and matches the category with the tag that is already available in the system. Finally, the system is able to send the messages to the users who are interested in Turkish pop singers by means of linked data feature. (This use case refers to Sample Use Case 1 in part 4.4.1.)

4.3.5. Message Delivery Delay

Assume that the user time profile is set to “non-business-hours” (as explained in 4.2.6. Event Subscriber part), meaning that, the messages are sent to the user only in out of business hours. Event Notifier creates an event in business hours. The system sends the event information to the related users immediately, except for the users whose time preference is set to “non-business-hours”. The messages of “non-business-hours” profiled users are delayed until the end of the business hours. Then, they receive the messages by means of the business rule engine by attaching “BRM-delayed” word into the original message, meaning that, Business Rule Message-delayed.


14:59 07 AĞUSTOS PER

YALNIZCA ACİL ÇAĞRILAR İÇİN

Caem Notification 14:58
BRM-delayed: Here's a new event for you: Tarkan Concert

USB hata ayıklaması bağlandı
USB hata ayıklama özelliğini devre dışı bırak..

Medya cihazı olarak bağlandı
Diğer USB seçenekleri için dokunun.

Figure 29 BRM-delayed message

4.4. Sample Use Cases

4.4.1. Sample Use Case 1

Ali is a computer engineering student in Middle East Technical University, Ankara. He registers and logins to the system. He adds “Turkish pop singers”, “Football” and “Action Movies”, as he is interested in these tags, and logs out. He sets his message delivery preferences as “Time profile: none, Location profile: on, Location: Ankara and Weather profile: on”. Mehmet is industrial engineering student in METU, Ankara. An event brochure is given to him in METU shopping center. He sees in the brochure that there is a Tarkan Concert in Ankara in 12/08/2014 at 19:00. Mehmet logins to the system and add the event details by specifying the place and date, and he forgets to add Tarkan Concert related interests. Finally, he pushes the create button. The system checks the event details, such as event title, date and place. Firstly, the system filters the words and gets the word “Tarkan”, and connects to the DBpedia with the “Tarkan” word. As a result, it receives the related words: “person”, “Turkish pop singers”, “Musical artist”, “Turkish-language artists”. It finds the tags available and tries to match the tag names with the related words. It finds “Turkish pop singers” category is a match. It queries the users who added the “Turkish pop singers” to their interests before. Secondly, the system gets the weather information of Ankara from OpenWeatherMap API for the specified date. As a result, the system receives “23 C, clear sky” weather data, and abstracts the data as “Good” weather. Thirdly, Business rule engine checks the conditions, and then gives a response about to send the notifications to the related users according to preferences of the users. Finally, the system sends the notification to the related users, “Here’s a new event for you: Tarkan Concert”. Ali realizes that he has a push notification from the system about Tarkan Concert, he finds it interesting and registers to the event in order to get event details and news about the event.

4.4.2. Sample Use Case 2

Melek is research assistant in English Literature department in METU and arranging a seminar event about William Shakespeare. Melek logins into system as publisher, and create the event with name “William Shakespeare Event” and with tags “English Literature” and “Sonneteers”. Zeynep is sociology student in METU, Ankara. She is interested in English poets and sonneteers. She registers to the system as subscriber and adds “Sonneteers” to her interests. Then, she arranges her message delivery preferences as follows: The weather profile is on, location profile is on and location is Ankara and time profile is on. When Melek as publisher creates the event, even if Zeynep is a related subscriber for the event, Zeynep does not receive the notification because the weather will be bad. Thus, she is not aware of the event. While using the mobile app, she taps the “Interesting Events” button from her Android phone. She realizes there is an event named “William Shakespeare Event” in Ankara, four days later. She finds it interesting and taps on the event to view detailed description about the event. System receives the current weather data about the event date, deduces the specified outdoor event is not suitable due to bad weather circumstances. While Zeynep is viewing the detailed description about the event, she realizes a warning message that the event is not suitable because of rainy weather. Then, she refuses to register to the event in order not to encounter a bad surprise.

4.4.3. Sample Use Case 3

Rana has a job in Ankara and is working very hard in business hours. Generally, she could not answer the phones and view the messages in business hours. She is interested in musical instruments and she always wanted to play a musical instrument. She logs into system and adds “Musical Instruments” tag into her interests. She misses to set the preferences and the time profile is set as “none” by default so that Rana has no time preference to receive the notification in certain times. An event is created named “Piano Training” in Ankara 5 days later. A push notification about this event is sent to Rana’s Android phone in business hours. She realizes she received message, but she couldn’t view the message. Later, she forgets to view the message and she misses the event. After the event time has passed, Rana realizes that she misses that interesting event. Thus, she taps on “Your Preferences” button and sets the time profile as “non-business-hours” in order to receive all messages in out of business hours. One week later, there is another event created in business hours, named “Guitar Training” in Ankara. The notification waits for business hours to pass, then goes to Rana’s phone in out of business hours in the evening. She finds the event interesting and registers to the event. When a new event, which Rana might be interested, is created in business hours, by means of business rule engine, the message waits until the business hour is passed then the system sends the push notification to Rana in order to maximize interaction rate with Rana’s attracted events.

CHAPTER 5

RESULTS AND DISCUSSION

In this section we discuss efficiency, importance and applicability of our implemented system. In order to demonstrate the value added by the system modules, we compare several different conditions of the system that are with the system modules and without system modules. The system modules of interest are semantic tag matcher, business rule management system and event dispatcher. For the assessment, we create test data set and sample scenarios. The sample scenarios are shown below. (Default values for unspecified inputs are, Event Type: “Indoor”, Weather Info: “Clear sky, Good”, Event Creation Time: “Out of business hours”, Subscriber Time Profile: “non-business-hours” meaning that the user receives all messages in out of business hours, Event Start Time: “20/08/2014 09:00”, Event Finish Time: “20/08/2014 13:00”, Event Place: “Ankara”, Subscriber Location: “Ankara”, Subscriber Interests: “Computer Science”, “Turkish Pop Singers”, “Sonneteers”, “Entertainment”, “Musical instruments”, and “American actors”. Standard user turns on all message delivery preferences, time dependency, location dependency and weather information dependency by default.

With the scenarios below, we test our system with or without the modules. We run first as a whole, “Full Active System”, then “Without Semantic Tag Matcher”, “Without Business Rule Management System” and “Without Event Dispatcher” as we explain in the scenarios.

In Table 5, we show the outputs from the scenarios. We assess the scenarios and create the table. In this assessment, we consider whether the subscriber is satisfied or unsatisfied with the event notification and the event experience. We say Satisfied (✓),

- If the subscriber is satisfied with the event that he/she receives its notification in appropriate conditions according to the subscriber preferences.
- If the subscriber is prevented from receiving unrelated events that the subscriber is not interested, because of geographically unrelated event that is far from the subscriber, because of an outdoor event that will have a bad weather, or because of an unsuitable time that the subscriber receives.

We say Unsatisfied (✗),

- If the system could not send the event message that the subscriber is actually interested in.
- If the system cannot avoid sending the event which could be an unrelated event that the subscriber is not interested, which could be a geographically unrelated event that is far from the subscriber, which could be an outdoor event that will have a bad weather, or which could be sent in unsuitable time for the subscriber.

5.1. Sample Scenarios

S1. Event Notifier creates an event named “Java Programming Language Training” and specifies the tags as “Computer Science” and “Programming”. Since the subscriber has “Computer Science” interest, the subscriber receives the notification in appropriate conditions with full active system. The subscriber is satisfied with this related event notification. Only if we remove event dispatcher and start the system, the subscriber cannot receive notification and cannot be satisfied. (The aim of this scenario is to show successful working of the system without the help of semantic tag matcher and BRMS)

S2. Event Notifier creates an event named “Tarkan Concert” without specifying any tags. The semantic tag matcher is able to find Tarkan related tag such as “Turkish Pop Singers”. Since the subscriber has “Turkish Pop Singers” interest and the semantic tag matcher is able to find “Turkish Pop Singers” interest from Tarkan, the subscriber receives the notification in appropriate conditions. The subscriber is satisfied with the related notification. Subscriber cannot receive the notification if we remove semantic tag matcher module, because the system cannot find related interests. If we remove event dispatcher, the notification cannot be delivered. (The aim of the scenario is to prove that the semantic tag matcher finds related topics in case the publisher is not able to set the related topics.)

S3. Event Notifier creates an event named “William Shakespeare Event” without specifying any tags. The event type is “Outdoor” and the weather will be rainy in this event. The subscriber does not receive the event notification because of bad weather, so the subscriber is satisfied, since the system avoids the subscriber from bad experience. If we remove BRMS that uses context sources such as weather data, the system will not prevent the subscriber from bad weather experience, so the subscriber will be unsatisfied. If we remove event dispatcher, the notification cannot be delivered. (The aim in this scenario is to show how BRMS uses weather context data and evaluates it to decide whether to send or not to send the notification.)

S4. Event Notifier creates an event named “Guitar Training” and specifies the tags as “Musical instruments” and “Music”. The event is created in business hours. The subscriber receives the message in out of business hours, since he/she set his/her time preference as “non-business-hours” before, in order to receive all related messages in out of business hours. Therefore, the system makes the messages wait if the event is created in business hours. If we remove BRMS module which actually handles the notification timing, the subscriber will not be satisfied, since he/she receives the notification in a busy hour when he/she does not want to receive any notifications. (The aim in this scenario is to show how BRMS uses time context data and evaluates it to decide appropriate time for sending message to the subscriber.)

S5. Event Notifier creates an event named “Tomato Festival” without specifying any tags. Semantic tag matcher remains unsuccessful in analyzing “Tomato Festival” semantically and finding related terms for current interests. Thus, the system cannot send the notification to the subscribers who might be interested in this event. The related subscribers such as the subscribers who are interested in “Entertainment” cannot receive the notification, so the related subscribers could not be satisfied, as they will not receive a related event notification. (The aim in this scenario is to show full active system might remain unsuccessful in sending notifications to the related subscribers in some cases.)

S6. Event Notifier creates an event named “Spam Event” with specifying all of the tags. The subscriber receives the event however; the subscriber is unsatisfied because the event does not mean anything to him/her. (The aim in this scenario is to show weakness of the full active system in filtering the event notifications and in blocking spam notifications.)

S7. Event Notifier creates an event named "Bradley Coming to Ankara" without specifying any tags and with the description, "American footballer Michael Bradley is going to play in the friendly match in Ankara." Semantic Tag Matcher tries to find tags about "Bradley". However, it finds "Brad Pitt" instead of "Michael Bradley". It matches the tag "American actors" with user interest "American actors", then the event is sent to the subscriber. The subscriber is not interested in "Football". Thus, the subscriber will be unsatisfied that the notification is unrelated with him. (The aim of this scenario is to point that linked data method in semantic tag matcher may sometimes bring wrong results that do not make the subscriber happy.)

We can show the outputs of the scenarios, with or without modules.

Table 5 System Evaluation With/Without Modules

	Full Active	**Without Semantic Tag Matcher**	**Without BRMS**	**Without Event Dispatcher**
S1: Correct Tags specified, other conditions nominal	✓	✓	✓	✗
S2: Tags not specified, semantic tag matcher is able to find tags	✓	✗	✓	✗
S3: Correct tags specified, outdoor rainy event	✓	✓	✗	✗
S4: Correct tags, event created in business hours	✓	✓	✗	✗
S5: Tags not specified, semantic tag matcher is unable to find tags	✗	✗	✗	✗
S6:Wrong tags specified	✗	✗	✗	✗
S7: Tags not specified, linked data brings wrong results	✗	✗	✗	✗

- Full Active System: Full active system works successfully on all scenarios except for Scenarios 5, 6, and 7. In Scenario 5, semantic tag matcher remains weak in finding new tags that the publisher is unable to specify. Semantic tag matcher could not find any related words for "Tomato Festival". In Scenario 6, the subscriber receives a spam event notification that he/she is not satisfied with. Therefore, a weak point of the system is that there is no spam filtering, so a user may create spam events. In Scenario 7, semantic tag matcher finds wrong results that are not related with the subscriber's interests. Thus, the subscriber will not be content with the event notification.

- Without Semantic Tag Matcher:
    - S1, S3 and S4 (Satisfied): The system will remain successful in keeping the subscriber satisfied since the tags are already specified, so there is no need for semantic tag matcher in these scenarios.
    - S2 (Unsatisfied): Since the publisher does not set any tags, without semantic tag matcher, the system cannot reach to the related subscribers. The subscriber will be unsatisfied due to missing a related event.
    - S5, S6 and S7 (Unsatisfied): Same reasons explained in Full Active System part above.
- Without BRMS:
    - S1 and S2 (Satisfied): Since default good conditions, which do not need context data such as weather data these scenarios will remain successful in making the subscriber satisfied.
    - S3 (Unsatisfied): In this scenario, we have an outdoor event and the weather will be rainy. Even if the context data service is open, the system cannot infer knowledge from context data without BRMS. Thus, the system will keep sending notifications even though the event is on a rainy day and not enjoyable. We can say the subscriber will be unsatisfied.
    - S4 (Unsatisfied): The event is created in business hours but the subscriber sets his/her time preferences as "non-business-hours" in order to receive notifications in out of business hours. Without BRMS, the subscriber receives his/her notification in business hours when he/she does not prefer to receive. Thus, the subscriber will be unsatisfied.
    - S5, S6 and S7 (Unsatisfied): Same reasons explained in Full Active System part above.

- Without Event Dispatcher: Absence of this service cause not to deliver any messages to the subscriber. The subscriber, of course, is still able to see the events from the user interface by manually querying the events.

5.2. Evaluation of the Results

Firstly, we realize from the scenarios that the semantic tag matcher tries to identify additional related subscribers. In case of not specifying any tags, semantic tag matcher searches for new categories to match the tags as a contribution to the publisher's task. In our system, semantic tag matcher utilizes the DBpedia source. However, as in seen in Scenario 5, DBpedia source remains insufficient in finding the related tags. For more accurate results, the external sources can be extended, and also an ontology would be helpful in addition to the linked data method.

Secondly, BRMS is very useful for inferring new knowledge for the users and the events, by aggregating the inputs so that the event message could be more meaningful for the user. For instance, BRMS, which uses the context data coming from another external sources such as weather service, deduces knowledge that whether the event is suitable for the user or not. In Scenario 3, we showed the power of BRMS by giving example of using weather context data. Additionally, BRMS can use location and time context data to arrange sending the notification to the subscriber. For instance, assume an event is created in business hours, the user might miss the messages if the message does not wait for out of business hours. Right here, BRMS does this job and sends the message in the most suitable time for the user by inferring the suitable time for each user, by considering time as in Scenario 4, or location and weather information. Therefore, without BRMS many of the event messages could be seen as spam or could be left unseen if received at an unsuitable time.

Finally, regarding the event dispatcher that handles message delivery, absence of this service results in undelivered messages. The user is only able to query the events and able to view the info manually without the event dispatcher. Thus, this service is significant in reaching the user on time.

It can be said that the system has weaknesses in some cases. Firstly, the publisher may create spam events with attached unrelated tags. Hence, there is a need for a mechanism in monitoring relationships between created event and the tags. Secondly, if the publisher does not specify any tags, and the semantic tag matcher was unable to find related tags, the event is not sent to any subscriber. Thus, the weak point of the system is that the existence of attached tags is mandatory. Thirdly, the linked data method used in semantic tag matcher may sometimes identify wrong tags, so, this situation leads the subscriber to be unsatisfied with the notifications.

If we talk about the assumptions in the scenarios, firstly, we assumed that the Standard User (subscriber) might be a student or might have a job. Thus, we created some predefined calendars for business rule engine. However, a user may not be that active in business hours or in weekdays. In the rule file, the calendars are static but we can create new rules with these calendars without changing the system itself. If want to change the calendar times or add a new one, we must deploy new rule definitions into the business rule management system (BRMS). Thus, these calendar times are our assumptions. Secondly, we assume clear sky and cloudy weather conditions as “Good” weather; otherwise the weather is “Bad”. In some conditions, for example, snowy conditions for some events may be suitable. In this case, because of our assumption, the event could not be notified because it is seen as “Bad” weather. However, in BRMS, we could define weather-specific rules to override this situation without deploying new code. For example, we could add a rule that has snowy condition:

```
When
event1 : Event (weatherSpecific == "Snow")
```

Or for more accurate results, we can define the condition with weather code.

```
When
Event2 : Event (weatherCode >= 600 && weatherCode < 700)
```

We give weather code between 600 and 699 since it describes snowy conditions for OpenWeatherMap API[19].

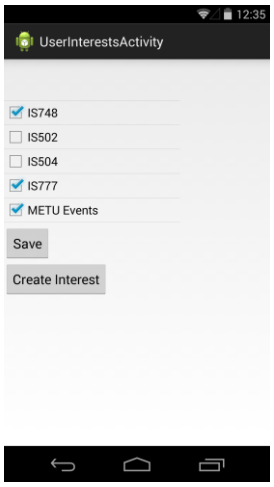
5.3. A Proof of Concept Application at METU Informatics Institute

As proof of concept, a sample application for METU Informatics Institute is presented by using Institute related data. For instance, we create course topics, course related events, nonacademic events in METU, and some entertainment activity announcements at the Informatics Institute. After arranging such an appropriate environment, we pick 12 users among MSc/PhD students, research assistants and non-academic personnel in METU Informatics Institute, and ask them to use and evaluate the system. We explain the aim of the system and show the mobile application for both publisher's and subscriber's views. Then, users register to the system as subscribers, and set related user interests such as course topics that they are taking, or social topics in the institute or in METU, as seen in figures 30, 31 and 32. The users also set their preferences in order to receive messages in appropriate conditions. As a publisher, we create topic related events. Finally, the users receive the event notifications. After using and evaluating the mobile system, a survey and an interview is conducted with the users. There are 5 questions in the survey and the users give scores to the system from 1 to 5. Many of the issues are explained and there is no missing knowledge regarding the system and from the survey questions. In the interviews, we ask for feedbacks and suggestions to improve the system. The questions, minimums and maximums, averages and standard deviations for the scores for each question are given in Table 6.

Table 6 Survey Results

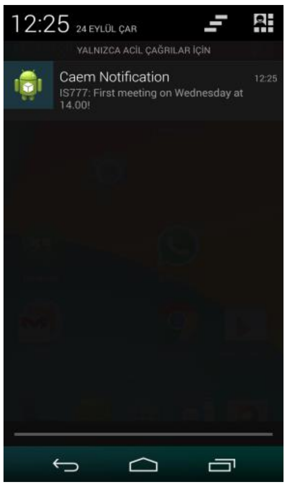
Questions	**Min**	**Max**	**Standard Deviaton**	**Average**
Q1. Are the notifications related with your interests? (1: Generally Unrelated, 5: Generally Related)	2	5	0,89	4,00
Q2. Are the notifications sent according to your preferences? (1: Rarely, 5: Generally)	2	5	0,90	4,27
Q3. How would you score the user interface of the mobile app? (1: Unsatisfactory, 5: Satisfactory)	1	3	0,60	1,82
Q4. How would you score your general satisfaction with the mobile application? (1: Unsatisfied, 5:Satisfied)	3	5	0,60	3,82
Q5. How much applicable and usable is the mobile app at METU Informatics Institute? (1:Hard to apply, 5: Easy to apply)	3	5	0,75	4,18

- For question 1, results show that the system generally sends related messages. This situation may be affected by that the publisher did not create unrelated events for each topic. The system also shows in this proof of concept application that, Semantic Tag Matcher did not find any unrelated tags to the events (similar to the situation described in Scenario 7 above), so the users were not disappointed.
- For question 2, the users generally receive their messages based on their preferences. The assumption could be effective on gathering high scores in here that if the users set time preferences such as non-business hours, the system is said to be successful if the related notification is immediately received with prefix “delayed”, instead of waiting for the specified time interval. It is meant that the notifications would be actually sent in selected time interval, when the users are unable to wait and track the situation.
- For question 3, the users generally give low scores to the user interface of the mobile application. In the mobile application, we mainly focus on functionality, so the user interface remains weak.
- For question 4, the users are generally satisfied with the mobile application. They appear satisfied more with the functionality of the system. However, the score in question 4, is lower than in questions 1 and 2. The reason might be that the user interface remains insufficient.
- For question 5, the users give high scores in applying the system at METU Informatics Institute. In this grading, the users might assume that if the mobile app is improved according to the feedbacks they give, the system seems more usable in METU Informatics Institute.



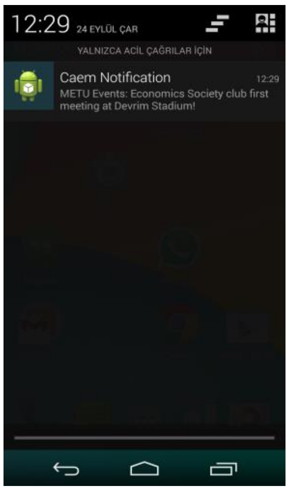
- [✓] IS748
- [ ] IS502
- [ ] IS504
- [✓] IS777
- [✓] METU Events

Figure 30 User Interests



Caem Notification
IS777: First meeting on Wednesday at 14.00!

Figure 31 Notification about IS 777 Course



Caem Notification
METU Events: Economics Society club first meeting at Devrim Stadium!

Figure 32 Notification About METU Events

After the survey, we interview some of the METU Informatics Institute research assistants, MSc /PhD students and non-academic personnel in order to evaluate the system and discuss the applicability and usability of the system at METU Informatics Institute. They generally find it feasible to use the proposed mobile system. We can list some main topics of the feedbacks and suggestions from the interviews:

1. New course notifications using topics: If a new course is opened in METU Informatics Institute, this new course topic can be added easily into system to send new course related notifications. (Actually, the feature is supported, but none of the users used it.)
2. Notification details: In the notification content, the publisher name and related topics of the notification should be available, so that the subscribers can gain knowledge about who the publisher is and what the topics are.
3. Authorization issue: The subscribers can be anybody who registers into the system. However, in publishing event notifications, there should be a constraining mechanism in order to prevent unwanted and unrelated messages. Two approaches are presented in the interviews. Firstly, a restriction for being a publisher can be applied. The publishers should only be teaching members, teaching assistants and other academic or non-academic personnel in METU. Secondly, a limit on setting academic tags can be applied. In other words, everyone can be a publisher, however, not everyone can create an event attached with e.g. IS 501 course tag. Academic tags must be only set by METU personnel.
4. Admin role: There should be an admin role in addition to the publisher and subscriber, in order to authorize METU personnel to set academic tags or to be a publisher.
5. Rate the publisher: There should be a rating mechanism by giving scores by the subscribers to the publishers, in order to increase the quality of the notifications.
6. Option to select attending/not attending: While viewing the event notification detail, there can be an option for the subscriber to specify attendance condition of the subscriber. It could be helpful, for instance, when an instructor plans an additional course and wants to learn the attendance situation of the students, this attendance information could help to cancel the course hour and arrange another time.
7. Context data can be helpful only in non-academic events: Using weather, location and time context data for the subscriber, and sending notifications according to them, can be very important and beneficial especially for non-academic events. However, it might not be that preferable for the academic event notifications. Thus,

there could be a separation between academic events and non-academic events in order to send every academic message to related subscribers.

8. Integration to METU OIBS: The system might be integrated into METU OIBS (Student Affairs Information System), so that taken current course topics can be automatically registered for the subscriber.
9. Statistics and reports: Analysis and statistics of created events can be presented visually by the system. For instance, the report may include the number of total events tagged by IS501 in the current semester, the number of posts of the selected publisher, the attendance percentage of the events, the number of total subscribers registered to a specific topic.
10. Friend list and social network integration: Friend list should be integrated into the system so that the subscriber can share the event with other subscribers who do not actually receive the event notification since they did not register to the related topics. Friend list can be gathered from phone contact list, email contact list or social network contact list. Moreover, the event should be shared via a social network application.
11. Integration with Google Calendar [56]: The events that the subscriber is attending can be integrated into Google Calendar so that external and internal events are combined in one place and the subscriber can display whole schedule.
12. Publish important events via publisher confirmation: The students as subscribers may sometimes find out about an event that has not been published into the system yet. In this case, the subscriber may have the opportunity to create an event draft with related course tags in order to send to the related publisher’s screen. If the publisher sees the event as important and confirms the related tags are correct, she/he confirms the event to be sent to the related subscribers. Therefore, the subscribers can contribute to event notification in addition to attending events. For instance, the CEO of a global IT company is coming to Ankara to attend a seminar. The seminar might not be course related but a subscriber, who finds out about the event, may think that IS 501 course students may be interested in this event. Since she/he cannot create an event or cannot create an event with academic tags, she/he creates the event by specifying the date and IS 501 topic. Then, he/she sends the event to the publishers to be confirmed. A publisher sees the event that could be significant, and confirms the event to publish the event for the IS 501 subscribers.

As we learn from the feedbacks, the users understand the system, and also they really want to use the system, since they describe concrete situations and requirements in an academic environment. They generally give feedbacks about improvements that they want to utilize. The system provides adequate functionality according to the interests and the preferences of the subscribers, but the user interface remains weak that we intend to improve first. However, we deduce that weakness of the user interface did not very much affect the functionality and value provided by the system.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In context-aware event recommender and notification systems, many successful studies have been conducted so far. For rich context and knowledge creation, time of day information, location, weather information, user preferences and demographic info (age, gender etc.), semantic environments, business rules have been used. Many studies have shown a good example of context-aware information delivery. For instance, some systems use complex algorithms to recommend accurate items. Some studies use social media to have more interaction with the people. There are some works using publish-subscribe approach with push notification. In addition to push notifications, there are business rule engines to send a message to the user in a specified time. The business rule engines in former systems focus on single job such as only message delivery. There are also some successful semantic Web features in new service discovery. Many studies have provided a solution with a recommendation engine, business rule engine, in semantic analysis, or in context-aware environment. However, the studies remain inadequate in combining many of these features into one solution, for instance, both successful recommendation, successful delivery of message at the right time to the correct person and immediate behavior altering of the system. Additionally, semantic analysis may prove to be insufficient, if ontology of the application domain does not exist. Furthermore, the studies do not provide adequate functionality and a feasible proof-of-concept application in finding the user at the most suitable time to deliver the info, when it finds a related info about the user.

Caglar [25] proposes a publish-subscribe based context aware reminder system that utilizes reminder patterns. The reminder patterns help to trigger reminding message by time or location defined by the publisher. Message trigger is limited to time or location info and limited to the offered patterns. Here, as a contribution to the study of Caglar, we propose a publish-subscribe system that is empowered with a business rule management system and semantic analysis using linked data. Business rule management system uses the context sources to create a context-aware medium for the subscriber, by applying complex event processing, assessing complex conditions and deciding suitable message delivery time. BRMS can work independently and modify itself quickly without any other system change. The advantage of our proposed BRMS is that we can use any context source as input and evaluate it to infer valuable knowledge. This could be time, user profile info, user location, user history, weather conditions or any other context data. We can define time based, location based and weather condition based events so that the subscriber will be able to receive the message in the most suitable time. Moreover, we use a semantic analysis with linked data through the semantic tag matcher component, in order to get linked words related to an event name. The aim of using semantic tag matcher is to maximize the success of sending messages to the related subscribers, when the publisher misses to specify all related tags on the created event.

In the results and discussion part, we apply the selected scenario to our implemented system and observe the results. Event dispatcher has the most significant effect on pushing notifications to the subscriber. In terms of message delivery, we realize that semantic tag matcher improves the performance by finding more tags to match. In terms of recommendation suitability performance, we can observe that BRMS has an important role in using context data and inferring valuable knowledge for recommendation and message delivery. On the contrary, as we realize from Scenario 6 in Results and Discussion part, the message can be delivered successfully, but the notification will not be meaningful to the subscriber, since the event is created as spam and it has unrelated tags on it. At this point the system has a weakness of not catching unrelated and spam events.

Furthermore, we consider the applicability at METU Informatics Institute. A proof of concept application at METU Informatics Institute is presented. To achieve this, the related tags and events are created. METU Informatics personnel and Informatics students are selected to use and evaluate the system. We find out that the system appears applicable, after gathering the feedbacks, requirements and the suggestions. However, for a proper usage of the system, there must be some improvements made, according to the selected users.

As future work, we intend to improve some modules of the system, by adding new features, new context sources and new refined algorithms in message delivery and recommendation. Firstly, semantic tag matcher can be improved to get more successful related results. We are planning to improve word-filtering algorithm by adding new word manipulation algorithms not only in English but also in Turkish and other languages. Thus, the event details will be more meaningful to the event management server to categorize the events, and the word similarity results will be more successful. Moreover, we are planning to add new semantic services, for instance we can use ontologies to improve the semantic tag matcher in addition to the DBpedia Lookup Service. As mentioned in the evaluation part, the linked data method may sometimes bring unrelated and wrong matching. Thus, adding new services and applying filtering to the queries from many sources may help reduce the wrong results. Secondly, we want to add new context sources to increase context awareness performance of the system. For instance, for indoor and outdoor positioning, we are planning to use GPS and also wireless sensors to locate the user accurately. Finally, we plan to add spam filtering module into system to filter unwanted messages, and also add the capability to block the corresponding user.

REFERENCES

[1] Kamar, A. (2003). Mobile Tourist Guide (m-ToGuide). Deliverable 1.4, Project Final Report. IST-2001-36004.

[2] García-Crespo, A., Chamizo, J., Rivera, I., Mencke, M., Colomo-Palacios, R., & Gómez-Berbís, J. M. (2009). SPETA: Social pervasive e-Tourism advisor.Telematics and Informatics, 26(3), 306-315.

[3] O'Hare, G. M., & O'Grady, M. J. (2003). Gulliver's Genie: a multi-agent system for ubiquitous and intelligent content delivery. Computer Communications,26(11), 1177-1187.

[4] Umlauft, M., Pospischil, G., Niklfeld, G., & Michlmayr, E. (2002). LoL@, A mobile tourist guide for UMTS. Information Technology & Tourism, 5(3), 151-164.

[5] Krosche, J., Baldzer, J., & Boll, S. (2004). Mobidenk-mobile multimedia in monument conservation. MultiMedia, IEEE, 11(2), 72-77.

[6] Neves, A. R. D. M., Carvalho, Á. M. G., & Ralha, C. G. (2014). Agent-based architecture for context-aware and personalized event recommendation. Expert Systems with Applications, 41(2), 563-573.

[7] Braunhofer, M., Elahi, M., Ricci, F., & Schievenin, T. (2013). Context-aware points of interest suggestion with dynamic weather data management. In Information and Communication Technologies in Tourism 2014 (pp. 87-100). Springer International Publishing.

[8] Beer, T., Fuchs, M., Höpken, W., Rasinger, J., & Werthner, H. (2007). Caips: A context-aware information push service in tourism. Information and Communication Technologies in Tourism 2007, 129-140.

[9] Haron, N. S., Saleem, N. S., Hasan, M. H., Ariffin, M. M., & Aziz, I. A. (2010). A RFID-based campus context-aware notification system. arXiv preprint arXiv:1003.4080.

[10] Martin, D., Alzua, A., & Lamsfus, C. (2011, January). A contextual geofencing mobile tourism service. In ENTER (pp. 191-202).

[11] Website of OpenSocial. (n.d.), Retrieved August 14, 2014, from http://opensocial.org/

[12] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

[13] García-crespo, Á., López-cuadrado, J. L., Colomo-palacios, R., González-carrasco, I., & Ruiz-mezcua, B. (2009). Sem-Fit : A semantic based expert system to provide recommendations in the tourism domain, 1–10. doi:10.1016/j.eswa.2011.04.152

[14] Information about Zigbee (n.d.), Retrieved August 14, 2014, from

http://www.zigbee.org/

[15] Information about Linked Data. (n.d.), Retrieved August 14, 2014, from http://linkeddata.org/

[16] Information about DBpedia. (n.d.), Retrieved August 14, 2014, from http://wiki.dbpedia.org/About

[17] Information about OpenWeatherMap API. (n.d.), Retrieved August 14, 2014, from http://openweathermap.org/API

[18] Sample OpenWeatherMap URL for 3-day weather information of Ankara as XML file. (n.d.), Retrieved July 26, 2014, from http://api.openweathermap.org/data/2.5/forecast/daily?q=ankara&mode=xml&units=metric&cnt=3

[19] List of weather conditions of OpenWeatherMap API. (n.d.), Retrieved August 14, 2014, from http://openweathermap.org/weather-conditions

[20] Description of Drools business rule engine. (n.d.), Retrieved August 14, 2014, frım http://drools.jboss.org/

[21] Description of JBoss Middleware. (n.d.), Retrieved August 14, 2014, from http://www.jboss.org/

[22] Tutorial document of complex event processing feature of Drools. (n.d.), Retrieved August 14, 2014, from http://docs.jboss.org/drools/release/6.0.1.Final/drools-docs/html/DroolsComplexEventProcessingChapter.html

[24] Description of JBoss Guvnor. (n.d.), Retrieved August 14, 2014, from http://guvnor.jboss.org/

[25] Caglar, O., (2013), A Context-Aware Reminder System Based On Publish And Subscribe Model, Retrieved August 14, 2014, from Middle East Technical University, Informatics Institute Web site: http://etd.lib.metu.edu.tr/upload/12616083/index.pdf

[26] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet computing, 6(2), 86-93.

[27] Collins, A. M., & Loftus, E. F. (1975). A spreading-activation theory of semantic processing. Psychological review, 82(6), 407.

[28] Nilsson, N. J. (1998). Artificial Intelligence: A New Synthesis: A New Synthesis. Morgan Kaufmann Publishers, Inc., 121-122

[29] Microsoft .NET Web API. (n.d.), Retrieved August 14, 2014, from http://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx

[30] Description of Microsoft Visual Studio 2013. (n.d.), Retrieved August 14, 2014, from http://www.visualstudio.com/tr-tr/downloads/download-visual-studio-vs.aspx

[31] Website of Microsoft Azure. (n.d.), Retrieved August 14, 2014, from https://azure.microsoft.com/tr-tr/vis

[32] Description of GCM. (n.d.), Retrieved August 14, 2014, from http://developer.android.com/google/gcm/index.html

[34] Android Developer Web page. (n.d.), Retrieved August 14, 2014, from http://developer.android.com/index.html

[35] Description of DBpedia.. (n.d.), Retrieved August 14, 2014, from http://wiki.dbpedia.org/About

[36] O'Grady, M. J., & O'Hare, G. M. (2004). Gulliver's Genie: agency, mobility, adaptivity. Computers & Graphics, 28(5), 677-689.

[37] Website of Eclipse platform. (n.d.), Retrieved August 14, 2014, from https://www.eclipse.org/

[38] Moreno, A., Valls, A., Isern, D., Marin, L., & Borràs, J. (2013). Sigtur/e-destination: ontology-based personalized recommendation of tourism and leisure activities. Engineering Applications of Artificial Intelligence, 26(1), 633-651.

[39] Gruber, T. R. (1993). A translation approach to portable ontology specifications.Knowledge acquisition, 5(2), 199-220.

[40] Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far.International journal on semantic web and information systems, 5(3), 1-22.

[41] Definition of URI. (n.d.), Retrieved August 23, 2014, from http://www.w3.org/TR/uri-clarification/

[42] Definition of RDF. (n.d.), Retrieved August 23, 2014, from http://www.w3.org/TR/PR-rdf-syntax/

[43] Hebeler, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2011). Semantic web programming. John Wiley & Sons.

[44] FOAF project web page. (n.d.), Retrieved August 23, 2014, from http://www.foaf-project.org/

[45] GeoNames web page. (n.d.), Retrieved August 23, 2014, from http://www.geonames.org/

[46] Liu, Y., & Plale, B. (2003). Survey of publish subscribe event systems.Computer Science Dept, Indian University, 16.

[47] Nagl, C., Rosenberg, F., & Dustdar, S. (2006, October). VIDRE--A Distributed Service-Oriented Business Rule Engine based on RuleML. In Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International (pp. 35-44). IEEE.

[48] BRMS definition from Hartmann Software Group IT Training & Consulting web page. (n.d.), Retrieved August 23, 2014, from http://www.hartmannsoftware.com/pub/Enterprise-Rule-Applications/brms

[49] Aida, K. (2000, January). Effect of job size characteristics on job scheduling performance. In Job Scheduling Strategies for Parallel Processing (pp. 1-17). Springer Berlin Heidelberg.

[50] Podnar, I., Hauswirth, M., & Jazayeri, M. (2002). Mobile push: Delivering content to mobile users. In Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on (pp. 563-568). IEEE.

[51] Wang, M., Kawamura, T., Sei, Y., Nakagawa, H., Tahara, Y., & Ohsuga, A. (2014). Context-Aware Music Recommendation with Serendipity Using Semantic Relations. In Semantic Technology (pp. 17-32). Springer International Publishing.

[52] Last.fm web page. (n.d.), Retrieved August 23, 2014, from http://www.last.fm

[53] Twitter web page. (n.d.), Retrieved August 23, 2014, from http://www.twitter.com

[54] Yahoo! Local web page. (n.d.), Retrieved August 23, 2014, from https://local.yahoo.com/

[55] LyricWiki web page. (n.d.), Retrieved August 23, 2014, from http://lyrics.wikia.com/Lyrics_Wiki

[56] Google Calendar web page. (n.d.), Retrieved August 23, 2014, from https://www.google.com/calendar/render?tab=mc%EF%BB%BF

TEZ
FOTOKOPİ
İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	☐
Sosyal Bilimler Enstitüsü	☐
Uygulamalı Matematik Enstitüsü	☐
Enformatik Enstitüsü	X
Deniz Bilimleri Enstitüsü	☐

YAZARIN

Soyadı : GÜRGAH
Adı : Melih
Bölümü : Bilişim Sistemleri

TEZİN ADI (İngilizce) : A CONTEXT-AWARE MOBILE EVENT NOTIFICATION SYSTEM USING THE PUBLISH-SUBSCRIBE MODEL WITH A BUSINESS RULE ENGINE AND LINKED DATA

TEZİN TÜRÜ : Yüksek Lisans [✓ X] Doktora ☐

1. Tezimin tamamı dünya çapında erişime açılsın ve kaynak gösterilmek şartıyla tezimin bir kısmı veya tamamının fotokopisi alınsın. ☐

2. Tezimin tamamı yalnızca Orta Doğu Teknik Üniversitesi kullancılarının erişimine açılsın. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) ☐

3. Tezim bir (1) yıl süreyle erişime kapalı olsun. (Bu seçenekle tezinizin fotokopisi ya da elektronik kopyası Kütüphane aracılığı ile ODTÜ dışına dağıtılmayacaktır.) X

Yazarın imzası .... Tarih : 10/10/2014