M3-Weather: A Smart-M3 World-Weather Application for Mobile Users

Anna Samoryadova, Ivan Galov, Pavel Borovinskiy,
Kirill Kulakov, Dmitry Korzun
Department of Computer Science,
Petrozavodsk State University (PetrSU)
Petrozavodsk, Republic of Karelia, Russia
E-mails: m3-weather@cs.karelia.ru, dkorzun@cs.karelia.ru

Abstract

This paper describes current state M3-Weather, a smart space application that provides to mobile users up-to-date information on current city and weather. The application is based on the Smart-M3 platform and consists of several Smart-M3 knowledge processors (KPs). KP-GUI and KP-GPScoords are running on end-user mobile device. KP-GUI is a desktop widget; KP-GPScoords detects current location of the user. KP-City and KP-Weather are daemons, which run on dedicated servers if needed. KP-City detects the nearest city; KP-Weather is responsible for up-to-date weather data for this city. The client part is oriented to Maemo/MeeGo devices such as Nokia N900 mobile computers. The project was initiated in February 2010 at Petrozavodsk State University.

I. INTRODUCTION

World weather servers are widely-spread today; let us mention Weather.com [1] as an example. From the client point of view an important problem is receiving up-to-date weather information regularly. In this case, the well-known browser-based approach is not convenient, especially when the users are mobile. Such users prefer smarter scenarios; the weather data are displayed on user's mobile device in accordance with the current geographical location of the user. In particular, the popularity of mobile world-weather clients increases, e.g., OMWeather applet [2] for Nokia 770/n800/n810/n900 devices.

Smart spaces [3] is a paradigm for constructing distributed smart applications, where many mobile users can cooperate with many services according with non-trivial business logic. Smart-M3 [4], [5] is an open-source development platform for implementing smart space applications. In this paper we describe M3-Weather, a world-weather tourist application implemented in Smart-M3.

Similarly to OMWeather our application provides weather information for user's current locality. In contrast, M3-Weather has another concept and architecture since it is based on smart spaces. It should be considered as the first proof-of-the-concept world-weather application in smart spaces. Moreover, M3-Weather can be easier integrated into other smart space applications because of the smart space features.

M3-Weather client is a desktop widget on Nokia N900 device. GPS module detects the current coordinates. Web service Geonames.org [6] provides names of the nearest cities. Web service Weather.com [1] provides weather forecast and other weather-related data.

As a Smart-M3 application, M3-Weather consists of five parts (Smart-M3 knowledge processors, KPs). KP-GUI and KP-GPScoords are running on end-user mobile device. KP-GUI implements user interface as a desktop widget and KP-GPScoords detects the current location. KP-City and KP-Weather can be run as daemons on separate computers, e.g., on

dedicated servers. KP-City detects the nearest city. KP-Weather is responsible for up-to-date weather data for that city. The last KP is KP-Coords; it allows the user to input coordinates manually.

The project was initiated in 2010 at Petrozavodsk State University (PetrSU), PetrSU-Nokia-NSN laboratory of wireless and mobile technologies [7]. M3-Weather belongs to the family of FRUCT research projects [8].

The rest of the paper has the following structure. Section II describes M3-Weather use case from the end-user point of view. Section III introduces our Smart-M3 architecture of M3-Weather. Section IV presents the ontology that we use to describe the structure of M3-Weather smart space. Section V compares M3-Weather to the existing work. Section VI summarizes our contribution and future research.

II. USE OF M3-WEATHER

M3-Weather is a smart space tourist application. One of its important advantages is its proactivity from the user point of view. The detection process

$$\text{Coordinates} \rightarrow \text{Nearest city} \rightarrow \text{Weather data} \tag{1}$$

is seamless to the user. For example, Alice is traveling from Petrozavodsk to Lappeenranta. Her mobile device runs the M3-Weather widget. It shows the weather in all the places that Alice has passed. Clearly, such "real-time" presentation of up-to-date weather information is very convenient to mobile users.

Fig. 1.   M3-Weather main usecase

Consider the basic M3-Weather use case by the example of Alice's trip, see Figure 1. When Alice is in Petrozavodsk, the widget shows weather (from Weather.com) for Petrozavodsk. When Alice appears close to Lappeenranta, the widget shows weather for Lappeenranta. The same happens in all intermediate points of the trip.

Furthermore, Alice can input coordinates manually, and the widget shows the nearest city and its weather. Figure 2 shows how our M3-Weather widget is displayed on Nokia N900 mobile computer. Note that M3-Weather uses an economical detection. When the change in coordinates is small, the application does not try to update the city name. This strategy reduces the network usage.

Figure 3 shows the scenario when several clients use M3-Weather. Smart-M3 semantic information broker (SIB) maintains the smart spaces of each user. In our example, Alice's and

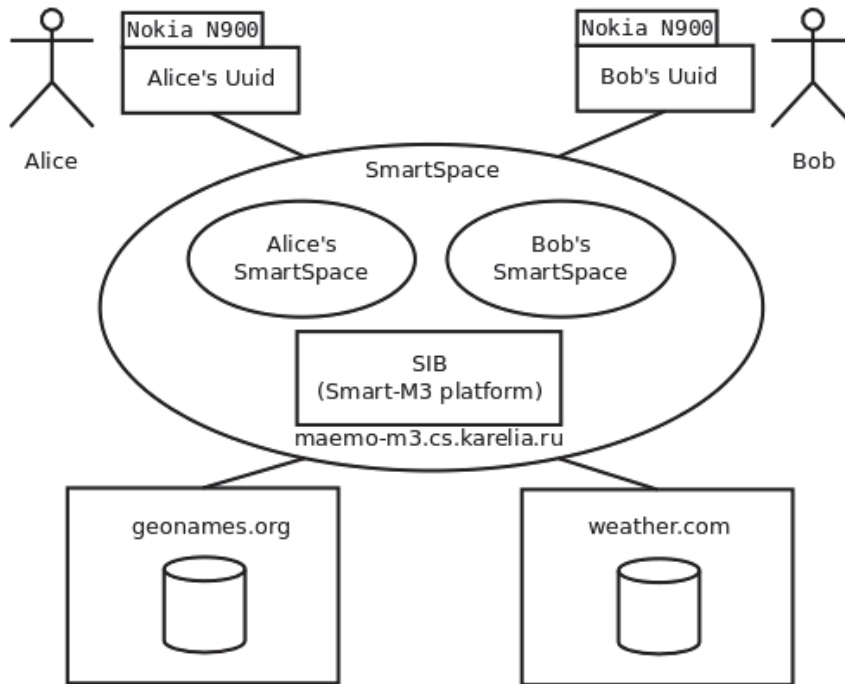Fig. 2.   Examples of M3-Weather widget on Nokia N900

Fig. 3.   M3-Weather using by several users

Bob's N900 devices are clients and SIB runs on machine `maemo-m3.cs.karelia.ru`. Each M3-Weather client accesses the SIB and publish coordinates into the personal smart space.

Data from GeoNames.org and Weather.com are also published in the appropriate personal smart space via the SIB. A client can read the data and then use them at its side. This way provides a base for more use cases utilizing a combination of several personal smart spaces into a group smart space. For example, Alice can know in what city Bob is currently located.

III. ARCHITECTURE

The Smart-M3 architecture of M3-Weather as a distributed application is presented in Figure 4. It consists of five KPs: KP-GUI, KP-GPScoords, KP-City, KP-Weather, and KP-
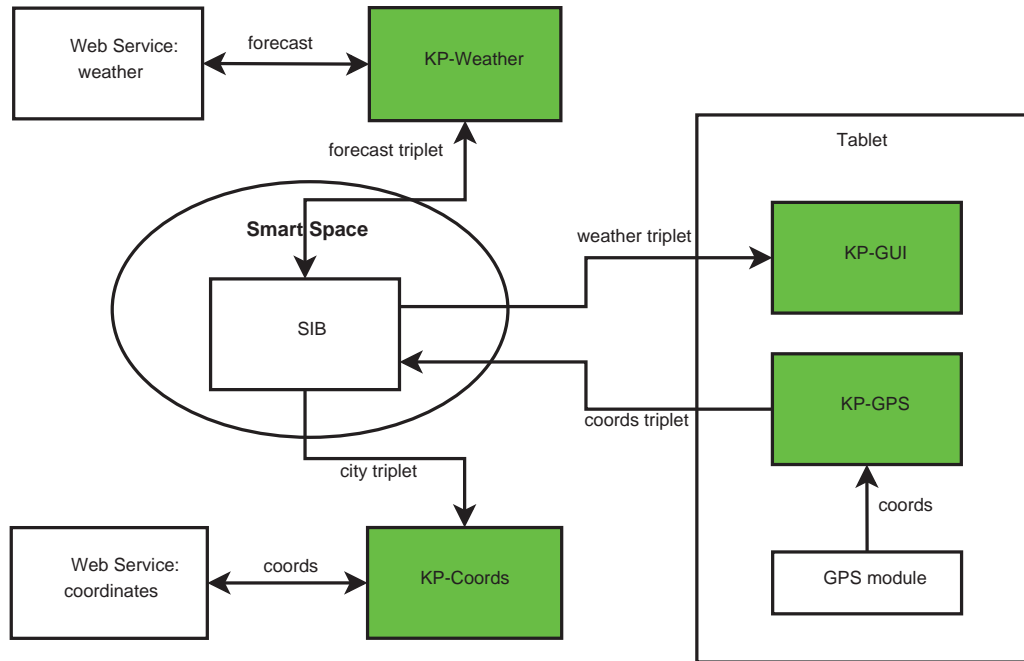

Fig. 4.   Smart-M3 architecture of M3-Weather. KPs are in green

coords. The latter KP does not participate in the basic use case, hence the KP is not presented in the high-level architecture. Let us consider how these KPs cooperatively realizes sequential mapping (1).

KP-GPScoords and KP-GUI run on end-user mobile device. KP-GUI is a desktop widget. It subscribes to weather and city data in the smart space. When new data appear KP-GUI visualizes them. KP-GPScoords regularly requests coordinates from device's GPS module. If they show significant change in the user location then KP-GPScoords update the coordinates in the smart space (via SIB). This proactive option can be switched off by the user in the widget (gps-buttom), and KP-GPScoords stops the update process (e.g., to save the resources).

KP-City and KP-Weather can work on dedicated servers. It reduces the load of the mobile device, and only KP-GUI and KP-GPScoords are required to run on user's mobile device.

KP-City subscribes for coordinates. When the data is updated (by KP-GPScoords or KP-coords), it send a request for the nearest city name to an appropriate web-server that maps coordinates to city names. Current implementation supports GeoNames.org [6]. Then KP-City publishes the result in the smart space.

KP-Weather subscribes for city names. When the data is updated (by KP-City), it send a request to an appropriate web-server that maps city names to weather data. Current implementation supports Weather.com [1]. Then KP-Weather publishes the result in the smart space.

When KP-GPScoord is stopped the user can activate KP-coords and input coordinates manually. Note that KP-coords is mostly intended for testing and demos. Its use in other use cases is deferred to our future development.

Communication with SIB is ontology-based: smart space content is structured by ontology. The ontology of M3-Weather smart space is described in Section IV.

Data exchanges with Smart-M3 SIB can use only low-level RDF triplets. We chose KPI_low [9] as the corresponding middleware library. It is modest to device capacity, e.g., suitable even for low-performance devices like sensors. KP-GUI utilizes KPI_low directly. All other KPs are constructed with SmartSlog tool [10], which allows programming in high-level ontology terms (OWL classes, properties, and individuals). Note that SmartSlog internally translates high-level ontology elements to low-level RDF triples and calls KPI_low to access SIB. Details of the M3-Weather code are given in Table I.

TABLE I
M3-WEATHER CODE

KP name	Language	Middleware	LOC	Downloads
KP-Coords	ANSI C	SmartSlog, KPI-low	262	16048
KP-GPSCoords	ANSI C	SmartSlog, KPI-low	280	16188
KP-City	ANSI C	SmartSlog, KPI-low	583	—
KP-Weather	ANSI C	SmartSlog, KPI-low	518	—
KP-GUI	Qt/C++	KPI-low	563	22891
Total	—	—	2206	55127

Download counters are by 25.9.2010. They are available only for KPs published as packages at the Maemo repository:

- KP-GPScoords: http://maemo.org/download-stats/index.php?unixname=m3w-gps&os=Maemo5&repo=extras
- KP-coords: http://maemo.org/download-stats/index.php?unixname=m3w-coords&os=Maemo5&repo=extras
- KP-GUI: http://maemo.org/download-stats/index.php?unixname=m3w-widget&os=Maemo5&repo=extras

KP-City and KP-Weather can run on platforms different from Maemo/MeeGo. The open-source code of all KPs can be found at http://gitorious.org/m3-weather, license GPLv2.

IV. M3-WEATHER ONTOLOGY

Smart space content is structured with ontology. The M3-Weather ontology is shown in Figure 5. There are three basic classes: Coords, City, and Weather.
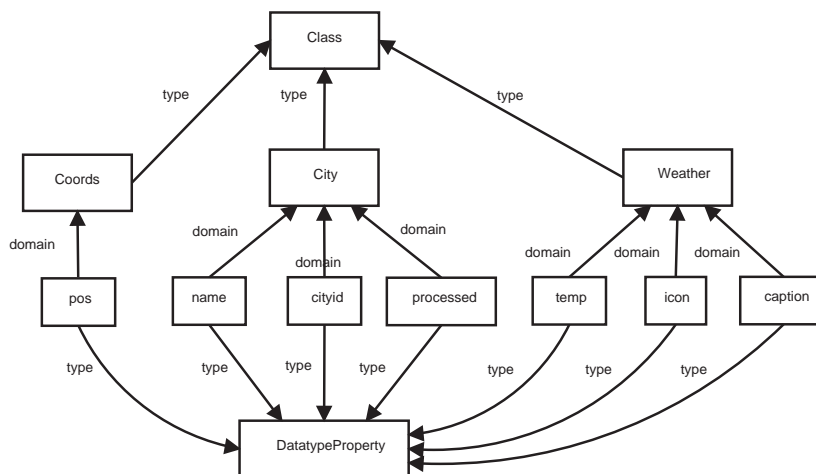

Fig. 5.   M3-Weather ontology

Class Coords describes current position of a mobile device. The class has one property pos that is latitude and longitude. One individual of the class represents one device.

Class City represents knowledge about a city. It has properties name, cityid, and processed. The former two are a conventional city name and a city ID on the weather service. The latter is an indicator of data actuality.

Class Weather has data properties temp, icon, and caption to store the temperature, the number of preinstalled pictures of weather, and short description of weather, respectively.

At initialization phase, KP-GPScoords creates its Coords-individual in the smart space. The individual has a unique ID in its name (related to the environment where the KP is running). Consequently, every device has its own individual Coords.

KP-City subscribes for all Coords-individuals. For each individual it creates a City-individual with the same unique ID in the name, requests a city name server, fills the properties, and publishes the individual in the smart space. While there are no up-to-date weather values the City-individual has property processed set to 'no'.

KP-Weather subscribes for all Coords-individuals with property processed = 'no'. For each individual it creates a Weather-individual with the same unique id in name. Then the KP requests a world-weather service, fill the properties, and publish the individual in the smart space. The property processed is set to 'yes'.

KP-GUI subscribes for appropriate City- and Weather-individuals (with the same ID) and displays the data whenever they have been updated in the smart space.

This ontology supports multiuser data. Each user device (running KP-GPScoords and KP-GUI) has a unique id, which identifies user's data. Therefore, each device has own (personal) smart space. The same SIB serves for maintaining multiple smart spaces from many users.

The M3-Weather ontology is simple and can be easily extended. For example, such extension can describe 5-day weather forecast. Several personal smart spaces can be combined to provide weather data of one user to another.

V. RELATED WORK

A known popular world-weather client for the Maemo/MeeGo platform is OMWeather [2]. It is a desktop widget with well developed user interface: it can be flexibly set up by the user as she/he would like. OMWeather can run as a full-screen application from the main menu of Nokia N900. OMweather uses Weather.com [1] and GISMETEO.RU [11] as world-weather web-services. The user can choose a city of her/his interest.

From the basic use case point of view, M3-Weather is very similar to OMWeather. The key difference in the architecture; M3-Weather is a distributed smart space application. It leads to the following benefits.

1. It is easier to integrate some parts of M3-Weather in other applications (where weather and city data are needed). Smart-M3 is a Multidomain, Multidevice, and Multivendor platform, thus the combination of homogenous part is seamless.

2. M3-Weather GUI is Qt-based in contrast to OMWeather GTK-based GUI. Most KPs is written in ANSI C. RDF middleware is SmartSlog and KPI_low without any strong restriction to mobile devices and their capacity. Hence, M3-Weather is strengthened again as a cross-platform application.

3. An OMWeather client uses its own static database for mapping coordinates to city names. It should be updated manually from GeoNames.org [6]. First, this solution consumes extra resources at users' devices. Second, the rare synchronization leads to less accurate

determination of cities. We discovered loss of the accuracy experimentally when traveling between cities of Republic of Karelia and compares the results of two clients.

VI. CONCLUSION

We described the current state of M3-Weather project. Packages of KP-GUI, KP-GPSCoords, and KP-Coords are oriented to use on mobile devices and published at the official Maemo repository, see Table I. The full code is publically available at Gitorious, a hosting for distributed open source projects.

Over 50 tests are implemented. There are 5 bugs posted at the project Bugzilla (see [7]). Several experiments were done using volunteers in real-life environments.

To the best of our knowledge, M3-Weather is the only world-weather client implemented according to the smart space paradigm. Although Smart-M3 is in the experimental phase, M3-Weather has certain advantages in comparison with OMWeather.

There are many ways how to develop the project further. User interface can be improved, e.g., similarly to OMWeather. The data set can be enhanced, e.g., with mid- and long-term forecast. A traveler can define several cities on her/his path and receive weather for them. If several users share their personal data during the trips, then various smart use cases can be supported.

ACKNOWLEDGMENT

REFERENCES

[1] "National and local weather forecast, hurricane, radar and report," Sep. 2010. [Online]. Available: http://www.weather.com/

[2] "Other maemo weather," OMWeather v0.28.4, Sep. 2010. [Online]. Available: https://garage.maemo.org/projects/omweather/

[3] I. Oliver, J. Honkola, and J. Ziegler, "Dynamic, localised space based semantic webs," in *Proc. IADIS Int'l Conf. WWW/Internet 2008.* IADIS Press, Oct. 2008, pp. 426–431.

[4] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *The 1st Int'l Workshop on Semantic Interoperability for Smart Spaces (SISS 2010) in conjunction with IEEE ISCC 2010*, Jun. 2010.

[5] "Download Smart-M3 software for free at SourceForge.net," Release 0.9.4beta, Sep. 2010. [Online]. Available: http://sourceforge.net/projects/smart-m3/

[6] "The GeoNames geographical database," Sep. 2010. [Online]. Available: http://www.geonames.org/

[7] "Open platforms for mobile devices," Petrozavodsk State University, Department of Computer Science, Sep. 2010. [Online]. Available: https://oss.fruct.org/

[8] "Open innovations framework program fruct," Sep. 2010. [Online]. Available: http://fruct.org/

[9] "Download KPI_low software for free at SourceForge.net," Jun. 2010. [Online]. Available: http://sourceforge.net/projects/kpilow/

[10] "Download SmartSlog software for free at SourceForge.net," Release 0.30, Sep. 2010. [Online]. Available: http://sourceforge.net/projects/smartslog/

[11] "Weather forecast for russia," in Russian, Sep. 2010. [Online]. Available: http://www.gismeteo.ru/