Understanding Context with ContextViewer – Tool for Visualization and Initial Preprocessing of Mobile Sensors Data

Szymon Bobek(✉), Sebastian Dziadzio, Paweł Jaciów, Mateusz Ślażyński, and Grzegorz J. Nalepa

AGH University of Science and Technology,
Al. A. Mickiewicza 30, 30-059 Krakow, Poland
{szymon.bobek,gjn}@agh.edu.pl

Abstract. Mobile context-aware systems are becoming more and more popular due to the rapid evolution of personal mobile devices. The variety of sensors that are available on such devices allow building intelligent applications that adapt automatically to user preferences and needs. Together with a growth of such self-adaptable systems, number of tools for collecting, visualising and modelling context were developed. However, there is still a need for tools and methods that will support building mobile context-aware systems at the very early stage of development. Such solutions should provide mechanisms for collecting, visualising and initial preprocessing of data to allow better understanding of processes, patterns and dynamics of mobile contextual data. In this paper we propose ContextViewer– a toolkit that aims at providing such mechanisms. It is a part of a methodology for building context-aware systems that besides ContextViewer includes also modelling methods and runtime environment for executing models.

1 Introduction

According to Dey [1] context can be defined as any information that can be used to characterise a situation of an entity. Such a general definition allows to treat different types of information as context, but also makes it difficult to model and process by context-aware systems (CAS). Over the last decade several frameworks supporting this task were proposed [2]. They provide mechanism for building context-aware applications which include collecting, visualising, modelling and processing contextual data. However, most of the available solutions were crafted for the needs of stationary environments. This assumes that the contextual information is dominated by the user activities within the system, like websites visited, applications ran[1]. Although such approach seems sufficient

This work was funded by the National Science Centre, Poland as a part of the KnowMe project (registration number 2014/13/N/ST6/01786).

[1] Exceptions here are systems that use eye tracking for user profile discovery, or EEG signal analysis for brain-computer interactions. This types of systems however, are beyond the scope of this paper.

H. Christiansen et al. (Eds.): CONTEXT 2015, LNAI 9405, pp. 77–90, 2015.
DOI: 10.1007/978-3-319-25591-0_6

for most of the desktop context-aware systems, it cannot be considered valid in mobile systems which use dynamic and time-dependant data.

Such systems are equipped with a variety of sensors like accelerometers, gyroscopes, mobile network providers, GPS, etc., which deliver huge amounts of information in a streaming manner. This information changes very fast in the short time perspective which is mainly caused by the nature of the streaming data itself, but also in the long time perspective, which reflects changes in the user habits and routines (phenomena known in the literature as *concept drift* [3]). It is very difficult to efficiently model system that process such data without prior knowledge about patterns within the data and correlations between context providers.

Machine learning methods provide mechanisms that allow for automated discovery of knowledge from large amounts of data. However, applying machine learning algorithm to context-aware system is not always straightforward. Choosing appropriate algorithm, its parameters, and input attributes (*feature set*) is non trivial task, which requires a lot of experiments and preliminary data analysis. Therefore, the comprehensive approach is needed that will provide tools and methods supporting the designer of mobile CAS in understanding the context at the very early stage of development.

As a response to this need, we proposed ContextViewer – a web application that allows for visualising and initial preprocessing of contextual information from mobile devices. It is integrated with AWARE[2] – one of the most popular framework for logging, sharing and reusing mobile contextual data. It allows for basic semantization of raw sensor data and provides bindings with machine learning software like WEKA[3] and ProM[4] to allow direct and immediate analysis of context. ContextViewer is a part of a larger toolkit which consists of ContextSimulator [5] and HeaRTDroid[6]. These tools provide respectively an lightweight simulation framework, and rule-based runtime for prototyping, testing and deployment of mobile context-aware applications.

The rest of the paper is organised as follows. Comparison of available frameworks for context management and motivation for our work was presented in Sect. 2. In Sect. 3 a detailed description of ContextViewer tool was provided. The remaining parts of the toolkit for building mobile context-aware systems that ContextViewer can be combined with, were briefly discussed in Sect. 4. In Sect. 5 an use-case scenario was presented. Summary and future work plans were enclosed in Sect. 6.

[2] See http://awareframework.com.
[3] See https://weka.waikato.ac.nz/.
[4] See http://www.processmining.org/prom/start.
[5] See http://glados.kis.agh.edu.pl/doku.php?id=pub:software:contextsimulator:start for details.
[6] See https://bitbucket.org/sbobek/heartdroid for details.

2 Overview of Context-Management Frameworks and Motivation

Context management frameworks can be divided into four groups:

1. frameworks that offer collecting, storing and distributing context within the system,
2. frameworks that provide visualisation of selected contextual information,
3. frameworks for modelling and processing context,
4. hybrid approaches that combine features of former three.

Systems that allow for collecting and storing contextual information are one of the most popular tools for building context-aware applications. They provide basic programistic interface for accessing context in a real time, as well as mechanisms for storing historical data. Examples of such frameworks are SDCF [4], AWARE, JCAF [5], SCOUT [6], and ContextDriod [7]. These frameworks do not provide any mechanisms for visualisation nor preprocessing of data. They serve as a middleware between the physical sensors or native API and the context-aware application. They usually store the data locally to preserve the privacy issues, but some of them like AWARE provides mechanisms for synchronization with external databases. This allows to transfer complex data analysis from mobile devices to the more powerful machines, integrate the knowledge from many sources and exchange this knowledge between the mobile devices in the distributed manner.

The other type of context-management frameworks are systems which focus on the visualisation of selected contextual data. In most cases these systems are commercial applications like: Ubidots[7] or Valarm[8]. However there are also open source solutions like Nimbits[9] and Freeboard[10]. The visualisation frameworks give more insight into the raw contextual data by plotting them on diagrams, or augmenting them on the map. They do not provide any tools for automated analysis of data, nor any runtime. Exception here is the Ubidots and Nimbits framework, which allows for including simple rule-based triggers that are fired when a context defined by some preconditions is met. These frameworks are mostly designed as cloud-based solutions which communicate with many mobile devices and provide web interface for visualising data.

Frameworks that support modelling context-aware systems and provide runtime environments for executing contextual models form the third class of context management systems. One of the most popular runtime for stationary context-aware application is ContextToolkit [8]. It supports rule-based modelling language and execution runtime. Other examples of rule-based frameworks crafted especially for the needs of mobile environments are Context Engine [9],

[7] See http://ubidots.com/.
[8] See https://play.google.com/store/apps/details?id=net.valarm.android.pro.
[9] See http://nimbits.com/.
[10] See https://freeboard.io/.

SWAN [10] and HeaRTDroid [11]. The last one is used by us as a part of the runtime environment.

Different approach for modelling and executing context models was presented by Brezillon et.al. [12]. They proposed a structure called Contextual Graph. It is a directed acyclic graph that represents the actions to undertake according to the context. The action nodes represent actions to undertake to achieve a goal while the event nodes become as explained above, contextual nodes describing the possible contextual issues of a given event.

The last group of context-management frameworks are hybrid solutions like CoBrA [13] for building smart meeting rooms, GAIA [14] for active spaces or SOCAM [15] and mobileGaia [16] – middlewares architecture for building pervasive systems. They combine features of frameworks for collecting storing and distributing context with features of frameworks for modelling and processing context. They do not provide tools for visualisation and data analysis. What is more, they are usually built for very narrow, specific solutions, and can not be considered as frameworks for more general purpose than initially assumed. The exception is MUSIC [17] framework – an open platform for development of self-adaptive mobile applications, which includes a methodology, tools and middleware.

A comprehensive comparison of all the frameworks and tools discussed in this section was presented in Table 1. It is worth noting that the solution proposed by us (bolded rows in the table) covers all the features that we distinguished important for the context management toolkit.

Although there exist numerous of such solutions, they all approach the issue of building context-aware systems in a standard manner, which begins with a modelling phase. However, mobile context-aware system are not yet well established field and there is still a lot of research ongoing in the area of context modelling, automated extraction of knowledge from multiple sources, uncertainty handling, etc. Therefore, in our work we aim at providing tools that will support the initial phase of the design of mobile context-aware systems which precedes the modelling phase. During this initial phase, the designer (or researcher) should be able to understand the contextual data that later will be modelled. This involves visualisation of this data, but also semantization of raw sensor streams and integration with most common machine learning software like WEKA of ProM for deep analysis. To achieve this task we designed and implemented ContextViewer and integrated it with AWARE – one of the most popular framework for building mobile context-aware systems. ContextViewer is part of software bundle for building context-aware systems that includes ContextSimulator and HeaRTDroidsystems for simulation, testing and deploying context-aware solutions on mobile platforms. In the next section the architecture and capabilities of ContextViewer are described.

3 Understanding Context with ContextViewer

ContextViewer is a tool for making Aware framework data easy to browse and process. It consists of four main modules implementing corresponding functionalities:

Table 1. Comparison of available frameworks for context management

Framework	Features									
	Context acquisition	Storage	Context distribution	Context visualisation	Context semantization	Context analysis	Modelling	Simulation	Runtime	Designed for mobile
AWARE	**yes**	**yes**	**yes**	**no**	no	**no**	**no**	**no**	**no**	**yes**
SDCF	yes	yes	no	no	no	no	no	no	no	yes
JCAF	yes	yes	no	no	no	no	no	no	no	yes
SCOUT	yes	yes	yes	no	no	no	no	no	yes	yes
ContextDroid	no	no	no	no	no	no	yes	no	yes	yes
ContextToolkit	no	no	no	no	no	no	yes	no	yes	no
Gimbal	no	no	no	partially	no	no	yes	no	yes	yes
Contextual Graphs	no	no	no	model only	no	no	yes	yes	yes	no
SWAN	no	runtime history	no	no	no	no	yes	no	yes	yes
Context Engine	no	no	no	no	no	no	yes	no	yes	yes
HeaRTDroid	**no**	**runtime history**	**no**	**no**	via rules	**no**	**yes**	**no**	**yes**	**yes**
Ubidots	not direct	yes	no	yes	rules	basic	basic rules	no	basic	yes
Valarm	not direct	yes	no	yes	no	yes	no	no	no	yes
Nimbits	not direct	yes	no	yes	rules	basic	basic rules	no	basic	yes
Freeboard	not direct	yes	no	yes	rules	basic	no	no	no	yes
ContextSimulator	**no**	**no**	**no**	**no**	no	**no**	**no**	**yes**	**no**	**yes**
ContextViewer	**no**	**no**	**no**	**yes**	yes	**yes**	**no**	**no**	**no**	**yes**
CoBrA	yes	no	yes	model only	ontology	no	yes	no	yes	no
GAIA	yes	no	yes	model only	ontology	no	yes	no	yes	no
MobileGaia	yes	no	yes	model only	ontology	no	yes	no	yes	yes
SOCAM	yes	no	yes	model only	ontology	no	yes	no	yes	yes
MUSIC	yes	yes	yes	model only	ontology	no	yes	yes	yes	yes

1. *Contextual data visualization module* – that aims at transforming raw Aware data to an intuitive graphical form. With it, instead of browsing database tables, user can inspect the data visually using only few intuitive controls.
2. *Basic semantisation module* – that is responsible for basic semantization of contextual data by translating raw sensor information into human readible concepts.
3. *Machine learning module*– which enables pattern discovery in contextual data. It includes WEKA data mining tool integration component and process mining component based on ProM tool.

The following section describes in details the architecture and implementation of these modules.

3.1 Architecture of ContextViewer

ContextViewer is a web application developed using Google Web Toolkit (GWT). GWT is an open source project that makes creating JavaScript front-end applications in Java possible.

Front-end of ContextViewer, referred later as a client-side application, is received over a network by users where it runs as JavaScript in their web browsers. At some point, the client-side needs to interact with a backend server. For example when loading or processing data. The so-called server-side of ContextViewer is based on Java Servlets. Client communicates with the server via HTTP, sending requests and receiving updates. The communication is done using remote procedure calls (RPC). The concept of ContextViewer's architecture is presented in Fig. 1. There is also an external contextual data provider, which is the Aware framework.

The client-side code implements everything that is responsible for displaying context data in an end user's web browser. Simple mechanisms for browsing the data are available for the user. Handling his or her input allows for changing display dynamically to match new conditions. The main goal of this part of ContextViewer is to transform sensors' data from tables with numbers or codes to its intuitive graphical representation. For example, numerical latitude and longitude are represented as a marker on a map using Google Maps API. For each mobile sensor type used there is a class for storing its context data. It is possible to easily add components for additional sensor types. There are methods for invoking interaction with the server, sending requests and receiving and handling results or answers. It is worth noting that to change appearance of ContextViewer's client one mostly needs only to edit included Cascade Style Sheet (CSS) file or to replace graphics to be displayed.

The server-side code implements servlets for handling a database, exporting data to ARFF and running a process mining tool. The main goal of this part of ContextViewer is to process the context data. Mobile sensors context data is stored in a MySQL database. Servlet loads data from a user-specified interval and sends it back to the client. Attribute-Relation File Format (ARFF) is a file type for use with Weka machine learning software. It is created by

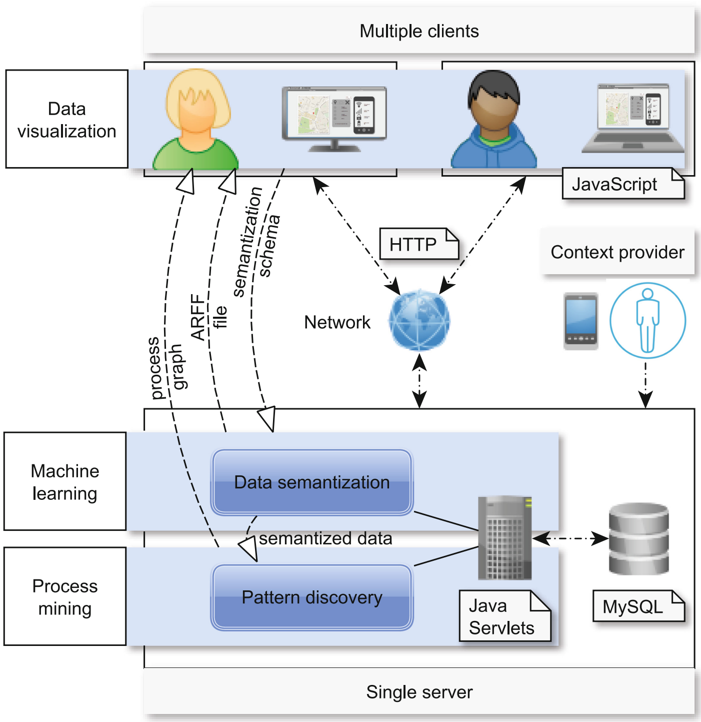


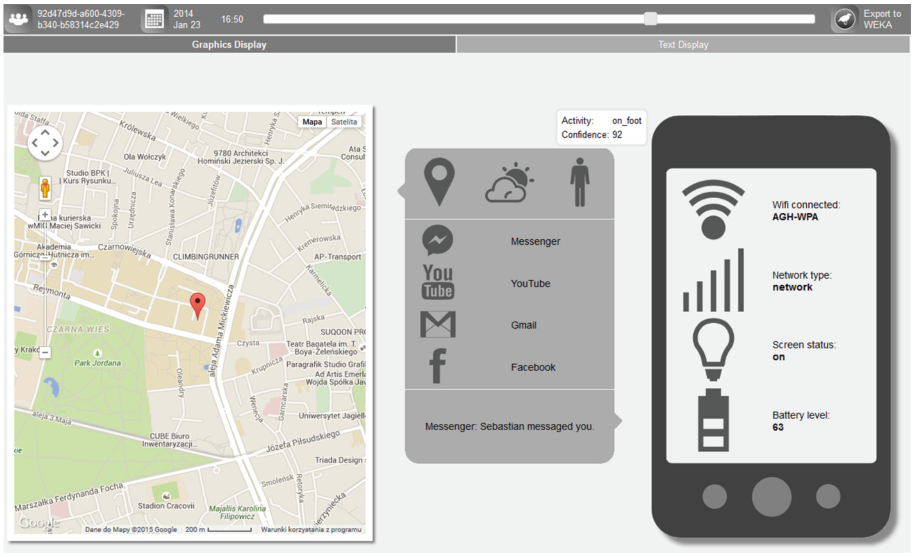
Fig. 1. The concept of ContextViewer's architecture

invoking a Python script on the server-side. Servlet allows the user to download generated file for further data mining in Weka at his or her workstation. Process mining is performed with ProM Tools. Servlet sends result to the client where it is displayed.

There are some requirements for ContextViewer to run. On the client-side, the end user's browser must have JavaScript enabled in order for the application to work and display correctly. To run the server-side code, ContextViewer must be deployed to a web engine. Currently, it is working with Apache Tomcat. Besides that, the server must have Java JRE 7, Python 3 and ProM 6 installed.

3.2 Visualization of Contextual Data

Visual inspection is an effective means to understanding context data. It can give useful insights concerning the data format, structure, and meaning. Since ContextViewer was initially designed as a visualization tool, it provides the user with several tools for data visualization. The user interface offers two interactive display modes: graphical and textual. The former is aimed to give a quick overview of collected data in an accessible form, while the latter offers a more detailed outline.



92d47d9d-a600-4309-b340-b58314c2e429
2014 Jan 23
16:50

Activity: on_foot
Confidence: 92

Messenger: Sebastian messaged you.

Wifi connected: AGH-WPA
Network type: network
Screen status: on
Battery level: 63

Fig. 2. ContextViewer user interface in graphical display mode

The graphical mode (Fig. 2) was designed to be maximally clean and intuitive. It is divided into three isolated sections, reflecting tiers of context data. The rightmost section presents low level information about the device: screen state, battery level, and network information. The middle section provides higher-level information about the device (running applications, notifications) and its surroundings (weather, activity recognition). The last section is a map marking the device geographical location. Users can easily interact with the application - change the device and date, use a slider to quickly run through the data, point to any icon to get more information, or play with a fully functional GoogleMaps plugin (including StreetView).

While in the text mode, ContextViewer lets users specify a shorter time interval with a slider and then simply displays the relevant data in tables. There are eight subsections: location, activity, weather, screen, applications, battery, wifi, and network. Although the scope of presented data is similar, the text mode is far more verbose. Take battery data for example: in graphical mode, only the battery level and adaptor status (plugged/unplugged) is presented, while the text mode includes precise measurements of voltage and temperature, as well as information about battery health, status, charges and discharges, all accurate to milliseconds. Location data are additionally plotted on a map, but now in form of splines reflecting device movement within selected time period, as opposed to a single point in graphical mode.

3.3 Basic Semantization of Contextual Data

Although data visualization is certainly a useful feature, the true power of ContextViewer lies in its semantization capabilities. Users can control and tweak the process by modifying the configuration file. It is a simple JSON file defining the mapping between unprocessed data from AWARE database and higher-level abstractions. Take GPS data for example. Raw values of latitude and longitude are not especially meaningful in modelling device environment or its owner's behaviour. The configuration file allows to define and label interesting areas such as workplace, home, city centre, as shown in Listing 1.1. The data is then translated accordingly.

Listing 1.1. An excerpt of the configuration file

```
{
  "label": "Work",
  "vertices": [
    { "latitude":50.066459, "longitude":19.926258 },
    { "latitude":50.061400, "longitude":19.924912 },
    { "latitude":50.067130, "longitude":19.897875 },
    { "latitude":50.071567, "longitude":19.900858 }
  ]
}
```

Time can be another example - Unix timestamp doesn't carry much information about context, but with ContextViewer we can get extract information about time of the day, day of the week, season etc. The config file is fully customizable, well-documented and easily extendable. The output is a CSV, ARFF, or XES file, which can be easily loaded into WEKA, R, Python or ProM to perform data mining or process extraction.

3.4 Process Mining and Pattern Discovery in Contextual Data

One of ContextViewer's modules enables process mining in contextual data. Firstly, data from an end user-specified interval is loaded. Next, on the server-side of ContextViewer, an external tool's algorithm detects some kind of business process based on given data log. Finally, the graph visualization that resulted from the analysis is returned to the user via RPC and is displayed in the client-side application. Figure 3 shows sample output graph in ContextViewer.

The external process mining tool invoked by ContextViewer's server-side servlet is ProM Tools [18]. ProM is a framework supports a huge variety of process mining methods in the form of plug-ins. It is free of charge, implemented in Java and cross-platform.

Currently used version of the tool is ProM 6. A servlet invokes its methods in a batch mode using a prepared script. ProM loads the data from a XES file, generates a graph visualization of discovered process and saves it into a PNG file. Before that, contextual data is converted from MySQL tables to XES (XML-based standard for event logs, [19]) file accepted by ProM.

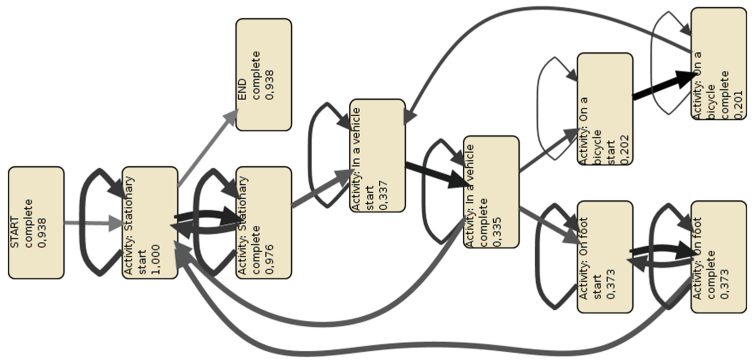

START complete 0,938
Activity: Stationary start 1,000
END complete 0,938
Activity: Stationary complete 0,976
Activity: In a vehicle start 0,337
Activity: In a vehicle complete 0,335
Activity: On a bicycle start 0,202
Activity: On a bicycle complete 0,201
Activity: On foot start 0,373
Activity: On foot complete 0,373

Fig. 3. Sample process discovered with ProM, visualising user transportation routines

In the following section the runtime environment was presented which allows to test and evaluate models discovered with WEKA and ProM.

4 Context Simulation and Processing

Results provided by the ContextViewer are used to perform data analysis in WEKA or ProM, but can not be directly executed on the device. In order to allow this, system designer has to implement a related functions on the mobile-platform's side. This task could be too laborious, particularly in the phase of prototyping. Therefore, to make the whole process as seamless as possible, tools for simulating and processing data according to the declarative model provided by system designer has to be provided. This section describes ContextSimulator and HeaRTDroidtools, created specifically to fulfil these requirement.

4.1 ContextSimulator

The simplest way to simulate device's sensors is to reuse the historical data that was already acquired and used earlier to perform the semantization on the server's side. ContextSimulator is a tool implemented in Java, that fetches the required parts of the ContextViewer's database and performs the simulation based on them. It allows system designer to repeat the series of sensors' readings in the controlled manner, first by choosing the speed of the simulation and secondly by dynamic allocation of the sensors, e.g. some of the can be turned off in order to check the robustness of the system. ContextSimulator was designed to extends the capabilities of the AWARE framework and therefore it is easily integrable with the existing ContextViewer's projects.

4.2 HEARTDROID

In order to semantically process data on the device, the toolkit was enriched with the HeaRTDroidinference engine — a rule-based environment designed to meet requirements of the modern mobile platforms The most distinctive features of the engine are usage of the XTT2 knowledge representation method, which renders the knowledge as a graph of the interconnected decision tables and straightforward integration with the Android platform. The XTT2 representation is formally founded in the ALSF(FD) logic [20]. HeaRTDroidwas successfully deployed in mobile context-aware application supporting the on-line threat monitoring system for urban environments [21].

The input of the HeaRTDroidconsists of the XTT2 textual representation written in HMR— a declarative modelling language based on the Prolog syntax. The model itself has to include all the types and attributes used in the conditional and decisive parts of the rules, which are specified in the latter section of the same file as parts of the formally defined schemes; all these elements together specify clean and unambiguous semantics of the computations performed by HeaRT-Droid. The basic example of the HMR rule, which has semantics described by the natural language sentence: “if `day` attribute has value `sat` or `sun`, then set the value of the `today` to `weekend` value”, is presented below:

```
xrule Today/1: [day in [sat,sun]] ==> [today set weekend].
```

More advanced modelling features of the HMR language allow handling of uncertain data (for example to represent sensors’ unreliability) by usage of the certainty factors associated with the elements of the model [22] and aggregating historical data to perform statistical tests on them in the conditional part of the rules. There exist also two types of statements connecting model with the application’s logic: (1) callbacks, which are used to set value of the attribute as a result of the code written in Java, (2) actions, which execute specified Java code, after the associated rule’s decisional part is performed.

The following section describes briefly how the entire bundle of ContextViewer, ContextSimulator and HeaRTDroidcan be used for the understanding of context and basic prototyping task.

5 Use Case Scenario

The use case scenario presented in this section was prepared using data from one month, collected with AWARE framework with LG Nexus 5 mobile phone. For the sake of simplicity and space limitation we will describe only one simple model extracted from this data with ContextViewer. The objective of the model was to minimize energy consumption of the mobile network provider. There are four different levels of connectivity quality available on mobile devices (in descending order with respect to quality): LTE, 4G, 3G and Edge. This connectivity levels corresponds to the energy consumption levels, with LTE being most energy consuming and Edge less energy consuming.

The initial analysis of data in ContextViewer panel revealed that there are three main location that the user is present during a day: Home, Work or in-between, called Commuting. It was also noted that the day of the user is relatively regular, therefore it is enough to partition it into: Morning, Day,

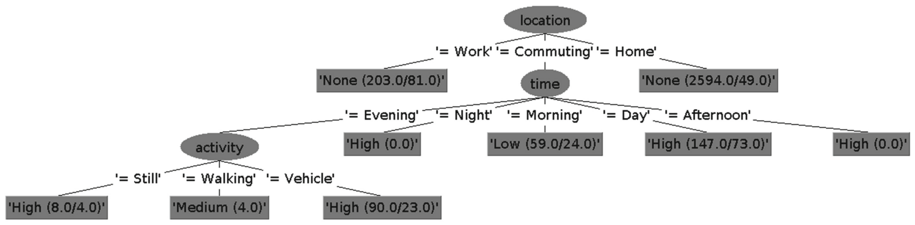
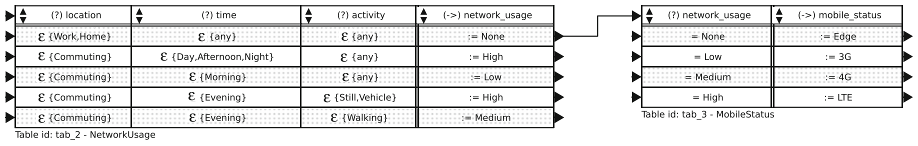


Fig. 4. Decision tree for prediction of mobile network usage with respect to user location, time and activity.



(?) location	(?) time	(?) activity	(->) network_usage
∈ {Work,Home}	∈ {any}	∈ {any}	:= None
∈ {Commuting}	∈ {Day,Afternoon,Night}	∈ {any}	:= High
∈ {Commuting}	∈ {Morning}	∈ {any}	:= Low
∈ {Commuting}	∈ {Evening}	∈ {Still,Vehicle}	:= High
∈ {Commuting}	∈ {Evening}	∈ {Walking}	:= Medium

Table id: tab_2 - NetworkUsage

(?) network_usage	(->) mobile_status
= None	:= Edge
= Low	:= 3G
= Medium	:= 4G
= High	:= LTE

Table id: tab_3 - MobileStatus

Fig. 5. XTT2 model for energy saving with respect to predicted mobile network usage. The model is based on the decision tree presented in Fig. 4

Afternoon, Evening and Night periods. More analysis of the network usage data allow to distinguish the thresholds for the usage for the: None, Low, Medium and High values. Also the activity of the user was defined based on the speed provided by the GPS sensor. All of this information was serialised into the JSON format used by the systematization module, in a form presented in the Listing 1.1 and exported to ARFF format. The decision tree generated with a WEKA is shown in Fig. 4.

The decision tree can be later transformed into the XTT2 tables as presented in Fig. 5 and evaluated with HeaRTDroidand ContextSimulator tools.

6 Summary and Future Work

In this paper we presented a toolkit for visualising and understanding context, which supports also rapid prototyping tasks. It is a bundle of tools integrated with AWARE – a framework designed for researched and developers for collecting, logging, and reusing context related information from mobile devices[11]. It provides also an integration with WEKA and ProM machine learning software. We focused on the phase that precedes the design and prototyping stages in a software development cycle. Our approach aim at providing tools that will allow the designer to understand the domain which alter will be modelled with appropriate methods. Although this approach could seem not scalable when we assume that the manual analysis has to be performed for every user separately, it should not be considered as a modelling phase in the development cycle. The analysis

[11] It is worth noting, that our work was has been acknowledged ad referenced on the AWARE framework website.

performed with ContextViewer and Weka or ProM reveals information on the higher level of abstraction like: which attributes are good/bad for particular use-cases, what discretization ranges are good/bad for semantization purposes, what places (how many) users usually visit, etc. This phase should give the designer better insight into the data and allow to experiment and rapidly prototype fragments of software that later will be deployed in a real-life application, possibly with an usage of automated methods.

For the future work we plan to develop tools that will automatically generate XTT2 rule models used by HeaRTDroidfrom the ContextViewer systematization configuration file. This will allow to reuse the information already provided by the designer/research into the ContextViewer and speed up the prototyping process.

References

1. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
2. Nalepa, G.J., Bobek, S.: Rule-based solution for context-aware reasoning on mobile devices. Comput. Sci. Inf. Syst. **11**(1), 171–193 (2014)
3. Gama, J.: Knowledge Discovery from Data Streams, 1st edn. Chapman & Hall/CRC, Boca Raton (2010)
4. Atzmueller, M., Hilgenberg, K.: Towards capturing social interactions with sdcf: an extensible framework for mobile sensing and ubiquitous data collection. In: Proceedings of 4th International Workshop on Modeling Social Media. ACM Press (2013)
5. Bardram, J.E.: The java context awareness framework (JCAF) - a service infrastructure and programming framework for context-aware applications. In: Schmidt, A., Want, R., Gellersen, H.-W. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 98–115. Springer, Heidelberg (2005)
6. Woensel, W.V., Casteleyn, S., Troyer, O.D.: A Framework for Decentralized, Context-Aware Mobile Applications Using Semantic Web Technology (2009)
7. van Wissen, B., Palmer, N., Kemp, R., Kielmann, T., Bal, H.: ContextDroid: an expression-based context framework for android. In: Proceedings of PhoneSense 2010, November 2010
8. Dey, A.K.: Providing architectural support for building context-aware applications. Ph.D. thesis, Atlanta, GA, USA (2000). AAI9994400
9. Kramer, D., Kocurova, A., Oussena, S., Clark, T., Komisarczuk, P.: An extensible, self contained, layered approach to context acquisition. In: Proceedings of the Third International Workshop on Middleware for Pervasive Mobile and Embedded Computing. M-MPAC 2011, pp. 6:1–6:7. ACM, New York (2011)
10. Palmer, N., Kemp, R., Kielmann, T., Bal, H.: Swan-song: a flexible context expression language for smartphones. In: Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones. PhoneSense 2012, pp. 12:1–12:5. ACM, New York (2012)
11. Bobek, S., Ślażyński, M., Nalepa, G.J.: Capturing dynamics of mobile context-aware systems with rules and statistical analysis of historical data. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2015. LNCS, vol. 9120, pp. 578–590. Springer, Heidelberg (2015)

12. Brezillon, P., Pasquier, L., Pomerol, J.C.: Reasoning with contextual graphs. Eur. J. Oper. Res. **136**(2), 290–298 (2002)
13. Chen, H., Finin, T.W., Joshi, A.: Semantic web in the context broker architecture. In: PerCom, pp. 277–286. IEEE Computer Society (2004)
14. Ranganathan, A., McGrath, R.E., Campbell, R.H., Mickunas, M.D.: Use of ontologies in a pervasive computing environment. Knowl. Eng. Rev. **18**(3), 209–220 (2003)
15. Gu, T., Pung, H.K., Zhang, D.Q., Wang, X.H.: A middleware for building context-aware mobile services. In: Proceedings of IEEE Vehicular Technology Conference (VTC) (2004)
16. Chetan, S., Al-Muhtadi, J., Campbell, R., Mickunas, M.D.: Mobile gaia: a middleware for ad-hoc pervasive computing. In: Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE, pp. 223–228. IEEE, January 2005
17. Floch, J., Fra, C., Fricke, R., Geihs, K., Wagner, M., Lorenzo, J., Soladana, E., Mehlhase, S., Paspallis, N., Rahnama, H., Ruiz, P., Scholz, U.: Playing music - building context-aware and self-adaptive mobile applications. Softw. Pract. Exp. **43**(3), 359–388 (2013)
18. van der Aalst, W.M., van Dongen, B.F., Günther, C.W., Mans, R., De Medeiros, A.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H., Weijters, A.: Prom 4.0: comprehensive support for real process analysis. In: Yakovlev, A., Kleijn, J. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)
19. Günther, C.W., Verbeek, H.: Xes-standard definition (2014)
20. Ligęza, A., Nalepa, G.J.: A study of methodological issues in design and development of rule-based systems: proposal of a new approach. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**(2), 117–137 (2011)
21. Bobek, S., Nalepa, G.J., Ligęza, A., Adrian, W.T., Kaczor, K.: Mobile context-based framework for threat monitoring in urban environment with social threat monitor. Multimedia Tools and Appl. (2014)
22. Bobek, S., Nalepa, G.J.: Incomplete and uncertain data handling in context-aware rule-based systems with modified certainty factors algebra. In: Bikakis, A., Fodor, P., Roman, D. (eds.) RuleML 2014. LNCS, vol. 8620, pp. 157–167. Springer, Heidelberg (2014)