

A Generic Large Scale Simulator for Ubiquitous Computing

Miquel Martin*, Petteri Nurmi†

* NEC Europe Ltd

Kurfuersten Anlage 36, 69115 Heidelberg, Germany

miquel.martin@netlab.nec.de

† Helsinki Institute for Information Technology HIIT

P.O. Box 68, University of Helsinki, FI-00014, Finland

petteri.nurmi@cs.helsinki.fi

Abstract—The complexity associated to gathering and processing contextual data makes testing mobile context-aware applications and services difficult. Furthermore, the lack of standard data sets and simulation tools makes the evaluation of machine learning algorithms in context-aware settings an even harder task. To ease the situation, we introduce a generic simulator that has been designed with the above mentioned purposes in mind. The simulator has also proven to be a good demonstration tool for mobile services and applications that are aimed at groups. The simulator is highly customizable and it can output context information of individual entities both through an interactive GUI and as data streams consisting of comma separated values. To support a wide range of tasks and scenarios, we have separated the three main information sources: behavior of agents, the scenario being simulated and the used context variable. The simulator has been implemented using Java, and the data streams have been made available through a web service interface.

I. INTRODUCTION

At the moment, the proper testing of mobile context-aware applications and services is a difficult task. One of the main reasons is the heterogeneous nature of data sources and sensors, which makes context gathering and processing difficult. In addition, the lack of readily available context infrastructure makes collecting large data sets unfeasible. Finally, raw sensors measurements seldom are meaningful to the user and instead we have to derive higher level abstractions from the data. For example, we could use GPS measurements to infer places that are meaningful to the user. However, the inference of these abstractions is both challenging and hard to evaluate. Furthermore, although prices of sensing devices have rapidly decreased in recent years, it is still expensive to equip multiple users, or their mobile devices, with the necessary sensors.

A closely related issue is how to evaluate and to compare machine learning algorithms in context-aware settings. Namely, although a number of datasets are freely available, e.g., [1]–[3], only very few are suited for this purpose. One of the main flaws in the existing data sets is that the data is gathered from very few users and with a limited set of sensors. In addition, the duration over which the data has been gathered is usually short.

The problems mentioned in the previous two paragraphs are by no means unique to context-aware settings, but common to all complex systems. To this end, researchers in these fields

have created simulation tools that attempt to give a suitable abstraction for the problem domain. In this paper we follow this line of thought and present a generic simulation tool for ubiquitous computing. The idea to build a simulator originated in the need to test the functionality of group services and applications before performing user evaluations. Furthermore, we wanted to generate data that was as realistic as possible in order to test machine learning methods that support group-awareness. We have followed ideas from the field of simulation and used a modular structure that separates behavior of agents, the scenario being simulated and the used context variables. As a result, our simulator is applicable to a wider range of tasks than originally intended. For example, we have used the simulator for application testing and we are currently exploring the possibility to create new, more realistic, mobility models for ad hoc networks.

The idea to use simulation in context-aware computing is still relatively new. One of the current research directions is to simulate low level sensors. For example, [4] simulates sensor measurements in a smart space. Another direction is to combine virtual reality simulation with the simulation of physical sensors. Examples of this approach include the QuakeSim [5], [6] and the UbiWISE simulators [7]. The third main direction is to use multi-agent simulation. As an example, the Generic Location Event Simulator [8] creates context-dependent events, which can be used, e.g., to test the functionality of a middleware. The simulator we present in this paper differs from earlier work on two aspects. First of all, we have adopted well established techniques from the field of simulation. Secondly, our simulator has been designed to be generic and applicable in a wide range of scenarios.

II. SIMULATION

In order to build a simulator that is as flexible as possible, we have separated the main information sources from each other. First of all, the behavior of individual agents is modeled in a separate agent model. Secondly, the environment, including possible random events, is handled in a world model. Finally, we have a context model which is used for defining how context data is simulated. In this section we detail each of these components in more detail and, as a starting point

for discussion, Fig. 1 shows a screenshot of the implemented simulator. A close-up view of the simulator as well as a view of the information panel that is used to visualize context information is given in Fig. 2. The agents in the scenario are workers in a city environment whose daily routines follow patterns such as going to work, to a restaurant, or to the movies.

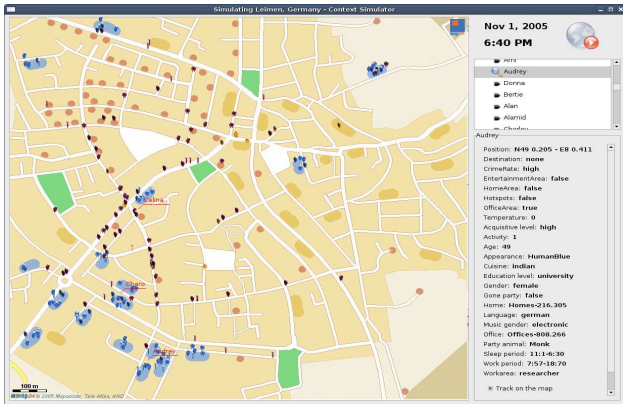


Fig. 1: A snapshot of the Context Simulator

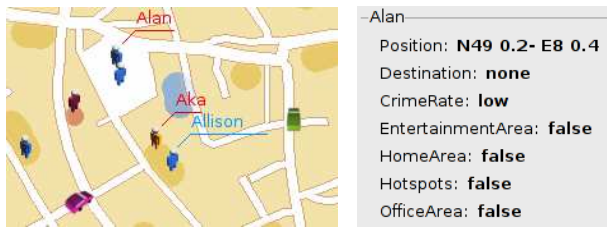


Fig. 2: Blow up of Fig. 1

A. Agent model

As the first information source we address the agent model, which is the decision logic of an agent. The agent model decides what an agent should do given its current context and the status of the entities such as places and people around it. The context of an agent is given by the context model, whereas information about the status of surrounding entities is provided by the world model.

The agent model can, first of all, change the properties of an agent. For example, it can switch the agent's activity from sleeping to working or change the mood from sad to happy. The agent model can also set a destination for an agent. In our setting, movement and path finding routines are handled automatically by the simulator, but the decision on where to go falls to the agent model.

In order to change the behavior of an agent, the simulation designer must assign a different agent model to the agent. To simplify creating complex simulations with large number of agents, and to avoid the need to create agents always from scratch, we allow using templates. An agent template is a

stereotyped agent model, which specifies the agent model on a high level. The templates are modeled as state machines, where the state change is triggered by context switches and optionally by random factors.

B. World Model

The next information source we consider is the world model, which consists of three parts: the environment that we want to simulate, the places of interest for the simulation and a global event model that handles global events such as public holidays. As an example we consider an urban scenario where the environment is the city being simulated. The places, on the other hand, could be, e.g., offices, homes and entertainment places. Finally, the global events include citywide festivals and public holidays.

The world model is specified by a set of image files, which are needed even if the GUI is not used. The base set of image files consists of a background map and of an alpha map. The alpha map defines the areas where actors can not step on, usually, walls. At the beginning of the simulation, the simulator uses the alpha map to pre-calculate a gradient map that is used for path finding. The gradient maps are stored on disk and need to be recalculated only if the simulated environment changes. Most of the places, such as offices and homes, are present throughout the simulation. However, some places, such as meeting places and pickup locations, exist only for a temporary time and disappear when they are no longer needed.

Places have properties, which are specified as key-value pairs and which can be modified at each iteration. For example, a nightclub can decide to throw a special party or a restaurant can decide to have special offers. In addition, agents can create new places or remove old places. Currently this is done deterministically so that whenever certain preconditions in the agent model are met, a new place is created or an old one is removed. For example, when the agent model indicates that a meeting must be arranged, a meeting place is created and, once the meeting ends, the meeting place is removed from the simulation.

To simplify the process of creating a world model, also places can be created using templates. For a place, the template specifies, first of all, the properties that a place has at the beginning of the simulation. Secondly, the template specifies the event model that is responsible for generating new events. Finally, the template specifies how and when the place changes its properties. Each one of these three components can also be left unspecified. As examples of templates we consider a nightclub and an office desk. For a nightclub the initial properties define the music style and the event model defines how and when the place generates special events. On the other hand, for an office desk the only thing that is needed is the location of the desk. The places that have a template can be instantiated as many times as needed by specifying an image map that indicates where an instance of the place should be. The simulator then automatically creates the places in the

locations specified in the image map and sets the properties of the place to their default values.

C. Context model

As the final information source we consider the context model, which manages the context variables that are used in the simulation. For each context variable, the context model specifies the possible values of the variable, the model that is used to simulate the values and potentially also how the values are distributed over the environment. As an example, we consider a context model that contains two variables: temperature and WiFi hotspot coverage. For temperature, the context model could specify, e.g., the mean temperature on specific areas as well as the standard deviation of the values. On the other hand, for WiFi hotspots the context model could include the locations of base stations, signal strengths of the base stations and a signal decay model that determines the signal strength at a given distance from the base station. Alternatively, the value of a context variable can be read from external sensors.

In order to provide an easy mechanism to initialize and update context variables we support annotated bitmaps. To generate a new context variable, the simulation designer needs to provide an overlay image map and a calibration file to the simulator. The simulator uses this information to calculate a mapping that associates pixel values with specific values of the context variable at each location in the simulated environment. Once the mapping has been calculated it is stored in a file. As an example we consider simulating temperature in a city. For this purpose, the simulation designer could generate a fractal cloud image. The intensity values of the image need then to be corrected to a uniform value on areas that denote indoor spaces. Next, the simulator calculates the mapping and when the simulator is run for the next time the output include the temperature value at the position of the agents. Fig. 3 shows overlays of temperature and WiFi hotspot coverage superimposed on the simulation map. The simulation user can also modify the overlay values at run time and (s)he can also replace the image map.

III. CONCLUSION

In this paper, we have presented a simulator that serves as a tool for testing the functionality of mobile applications and services. In addition, our simulator has been designed as a means to generate data for the evaluation and the comparison of different machine learning methods in mobile context-aware settings. Our main focus has been on tasks related to the usage of group context information. The simulator is not meant to replace user studies or the collection of real data, but rather to serve as a way to perform sanity checks, preliminary evaluations and scalability tests before validation with real users. Each scenario that is being simulated is specified by three information sources: an agent model, a world model and a context model. By having a clear separation between the different information sources, our simulator has become applicable to a wide range of scenarios. In addition, we have

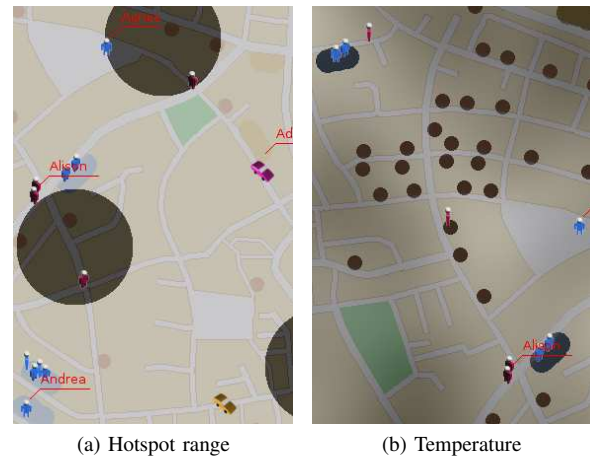


Fig. 3: Overlay examples shown as a transparency on the area map.

used a template-based technique, which simplifies the work of the simulation designer. In this paper we have focused on presenting how these information sources are handled in our simulator. In the future, our goal is to extend the functionalities of the simulator, to explore additional application areas and to perform rigorous user studies with different user groups.

ACKNOWLEDGMENT

The authors would like to thank Patrik Floréen, Jukka Suomela and Alban Hessler for their support and ideas. This work has been performed in the context of the IST project IST-2004-511607 MobiLife, partly funded by the European Union. The authors would like to acknowledge the contributions of their colleagues, although the views expressed are those of the authors and do not necessarily represent the project.

REFERENCES

- [1] (2006, May) CRAWDAD: A community for archiving wireless data at Dartmouth. [Online]. Available: <http://crawdad.cs.dartmouth.edu/meta.php?name=intel/placelab>
- [2] M. Raento. (2006, May) Mobile Communication and Context Dataset. [Online]. Available: <http://www.cs.helsinki.fi/group/context/data/dataset.pdf>
- [3] (2006, May) Context Database. [Online]. Available: <http://www.soft.uni-linz.ac.at/Research/Context.Database/index.php>
- [4] M. C. Huebscher and J. A. McCann, "Simulation model for self-adaptive applications in pervasive computing," in *Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA)*. IEEE Computer Society, 2004, pp. 694 – 698.
- [5] M. Bylund and F. Espinoza, "Using Quake III Arena to simulate sensors and actuators when evaluating and testing mobile services," in *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, 2001, pp. 241 – 243.
- [6] —, "Testing and demonstrating services with Quake III Arena," *Communications of the ACM (CACM)*, vol. 45, no. 1, pp. 46 – 48, Jan. 2002.
- [7] J. J. Barton and V. Vijayaraghavan, "UBIWISE, A simulator for ubiquitous computing systems design," Hewlett-Packard Laboratories Palo Alto, HPL-2003-93, 2003.
- [8] K. Sanmugalingam and G. Coulouris, "A generic location event simulator," in *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp)*, ser. Lecture Notes in Computer Science (LNCS), G. Borriello and L. Holmquist, Eds., vol. 2498. Berlin Heidelberg: Springer - Verlag, 2002, pp. 308 – 315.