SimuContext: Simply Simulate Context[1]

Tom Broens, Aart van Halteren
Centre for Telematics and Information Technology,
University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands,
{t.h.f.broens, a.t.vanhalteren}@utwente.nl

Abstract

Testing and demonstrating context-aware applications is challenging. Gathering and using 'life' contextual information for these purposes, often requires significant extra development effort. Furthermore, repeating tests and demonstrations of context-aware applications that use 'life' context information, in a controlled way, is difficult or often impossible to achieve. This paper presents a context simulation framework, called SimuContext, which abstracts from the complexity of interfacing with physical context sources and facilitates testing and demonstrating context-aware applications in a controlled way.

1. Introduction

Context-Awareness (CA) is a promising paradigm to personalize applications. Traditionally, non-CA applications only use explicit user-inputs to provide output. CA applications use additional implicit (sensed) context inputs collected from the user's environment, to tailor the output to the user's need [1]. Context (e.g. location, speed) is sensed from the user's physical environment or derived from the user's computing environment. In CA applications, context sources represent software entities that produce context information relevant for the CA application (see Figure 1).

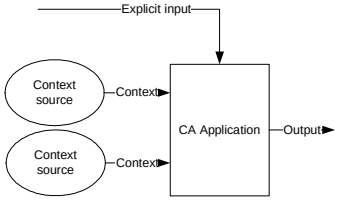

Figure 1. Context-aware application

The design and development of CA applications does not only concern traditional application logic, but also includes logic that deals with gathering and processing of context information coming from context sources. Therefore, application developers also face the challenges posed by the use of context sources.

Currently, context-aware technologies are still evolving and are not yet widely deployed. Therefore, to develop context-aware applications, a significant extra amount of development effort may go into the implementation of context sources. The general structure of such a context source (see Figure 2) is that it acquires raw data from a data source (e.g. sensor), possibly performs some processing and then transforms it into a data structure suitable for the context-aware application.

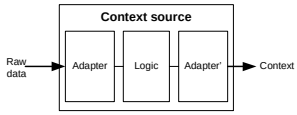

Figure 2. Generic structure of a context source.

Implementing these data transformations is quite cumbersome and time consuming. Furthermore, due to the device-specific nature of context sources, reusability of context sources and their transformations for other applications is limited. For example, to implement a context source of a Bluetooth GPS, the developer has to build a context source that can communicate with Bluetooth and the NMEA data format and then transform this to the data format that can be used by the CA application.

Ideally, CA application developers want to abstract from the internals of context sources and treat them as a black box. Their effort should go in the development of the CA application logic which includes interfacing with desired context sources. We recognize that for operational CA services, physical context sources are mandatory. However, we envision that in the future third parties will develop context sources and therefore creating new context sources seems redundant, takes scarce development time and distracts from the real challenge of developing a CA application.

[1] We would like to thank Marten van Sinderen for his contributions to this paper. This work is part of the Freeband AWARENESS project. Freeband is sponsored by the Dutch government under contract BSIK 03025. (http://awareness.freeband.nl)

Furthermore, testing and demonstrating CA applications in a controllable and reproducible way with physical context sources may prove extremely difficult. By nature, context information is highly dynamic (i.e. dynamic in value and quality) [2, 3]. Therefore, retrieving the same context in similar situations is hard. For example, when using a GPS, standing in the same spot can result in different context values over time due to changing accuracy. Also practically, it is hard to use life context information during tests or demonstrations. For example, GPS does not work inside buildings, which means that tests and demonstrations have to take place outside.

In this paper, we propose a context simulation framework (SimuContext). Simulation in general provides many benefits for software development like, cost reduction, improve reliability and shorten development time [4]. Our framework facilitates application developers in testing and demonstrating CA applications by only specifying the behavior of context sources and using simulated context sources as inputs in their CA application. As a result, CA application developers can concentrate on the logic of their application and do not have to worry about the complexities of dealing with physical context sources. The SimuContext framework has the potential to offer CA application developers the following:

- Rapid testing and demonstration of CA applications, in a controllable and reproducible manner, through configuration of existing SimuContext capabilities or by extending parts of the framework.
- Creation of a more comprehensive and realistic validation environment by deploying a multitude of SimuContext sources with specified properties.
- Easy replacement of SimuContext sources by physical context sources due to a realistic context model and generic SimuContext interfaces.
- Test specific characteristics of context sources for CA applications like subscribe/notify mechanism, connection loss, reasoning, quality changes etc.

Bylund [5] distinguishes two types of context simulation tools: (i) a simulation suite that simulates context values and (ii) a semi-realistic simulation environment. We have chosen the first approach to offer a generic simulation facility which can provide a configurable set of context source types and instances, tailored to a specific CA application scenario. Summarizing, the SimuContext framework substitutes the implicit context inputs of CA applications with context inputs from SimuContext sources (see Figure 3).

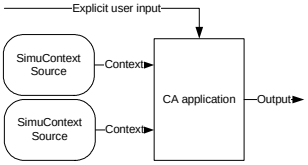

Figure 3. CA application with SimuContext sources.

The remainder of the paper is structured as follows. Section 2 discusses the requirements analysis which includes the definition of a context model. Section 3 presents the design of the SimuContext framework. Section 4 discusses the proof-of-concept implementation of the framework. Section 5 presents a discussion. Finally, section 6 gives some conclusions and future work.

2. Requirement analysis

This section discusses the non-functional (section 2.1) and functional requirements (section 2.2) for simulating context sources.

2.1. Non functional requirements

This section describes common quality aspects we require from our simulation framework:

- *Generality:* the framework should be generic enough to simulate a diversity of context sources (e.g. computing context, user context, physical context [6]). Furthermore, the framework should not pose severe constraints on the target application, and therefore it should be reusable for multiple applications.
- *Extensibility:* it should be easy to extend the framework to support specific context sources for specific context-aware applications.

2.2. Functional requirements

To create an accurate and realistic simulation framework for context sources, we have to analyze the nature of context. We start this analysis by researching what information elements are encapsulated by context.

From the context model (See Figure 4), we see that context consists of information on several levels of abstraction. First, context has meta-information on its quality, which is called quality of context (QoC). Common parameters that we adopt as an example, are [7]: Precision, Correctness, Trustworthiness, Resolution and Up-to-dateness. Precision describes how well the context information mirrors the reality. Correctness indicates the probability that context information is correct. Trustworthiness indicates the probability that context information is correct with respect to the context provider. Up-to-dateness depicts the age of context information. For example, the

physical location can be produced by a GPS. Such a context source has a precision of approximately 5 meters. Correctness of GPS is high outside with clear view on the sky but low in areas with high buildings. Trustworthiness and up-to-dateness is high because the context is machine-produced and updated regularly. Furthermore, context is privacy sensitive information [8]. Security information (e.g. who is authorized to retrieve this context) is therefore part of the meta-information. However, in this version of the framework we focus on the core functionality of context simulation. Therefore, we consider security a future extension of the framework.

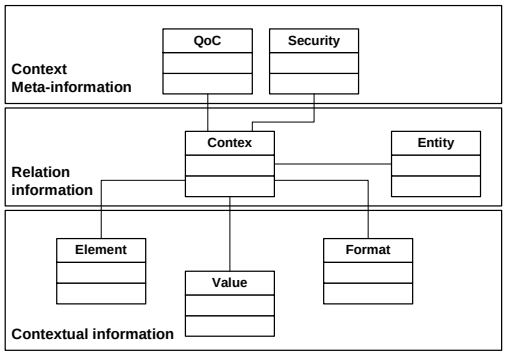

Figure 4. Context model

Second, context encapsulates information related to what it describes (relation information). Dey [9], defines context as any information that can be used to characterize the situation of an entity. This indicates that context is always related to an entity. An entity can be a human, physical, or computational object. For example, context can be related to Person X, Table Y or Application Z.

Finally, context encapsulates the real contextual information. This consists of an element, describing the context identifier (e.g. Location). Then it has a value, for example 52.123/6.23123. Finally, it has a format that describes the value, for example Lat/Long. Taking these aspects together, this leads to the following requirement:

- The SimuContext framework should support the elements that are encapsulated by the defined context model. This includes QoC, the entity to which it is related and the elements that describe the contextual information.

Additionally, context has some other characteristics [2, 3]: (i) Context is temporal, (ii) context is spatial, (iii) context is imperfect and (iv) context sources are often distributed. These characteristics trigger the following requirements for our framework to realistically simulate context sources:

- Due to its temporal and spatial nature, context is subject to continual changes. Therefore, the simulated context source should be able to have changing values specified in an application specific, pluggable value model.
- Furthermore, to facilitate the user to simulate changing context, our framework should support two types of invocation mechanisms: (i) Request – Response and (ii) Subscribe-Notify. The latter mechanism enables users to specify a condition when they want to be notified of a context change.
- To provide the Subscribe-Notify mechanism our framework should support an event model that specifies when context change events should be generated. This event model should be easy to plug into our framework.
- Users should be able to specify a notification condition, based on context. For example, speed > 50 km/hr.
- Due to the imperfect nature of context, it inherently has quality properties. Our framework should be able to express this in a QoC model. The realization of the QoC model depends on the target application therefore the QoC model should be inplugable and extendible.
- The quality values of context are related to the provided context. As this is subject to change the quality values are also subject to change. Our framework should support changing QoC values in a value model.

This requirement analysis provides an initial description of what capabilities our framework supports. In the future work section of this paper (see section 6), we identify additional requirements for future enhancement of our framework.

3. Design of the SimuContext framework

This section discusses the design of the SimuContext framework. We start with a high-level overview followed by more details on provided interfaces and the functional architecture of a SimuContext source. In Figure 5, we present a black-box overview of the SimuContext framework.

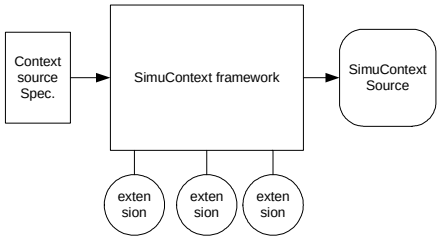

Figure 5. Overview of the SimuContext system

In the configuration phase, the user or application developer specifies a context source it wants to simulate. Additionally, they can implement application-specific extensions to the framework, which can be plugged-in, and then be used in the context source specification. For example, a developer can create a log file reader extension which is used to generate the context values. In the operational phase, the framework parses the context source specification and instantiates the corresponding SimuContext source. The simulated context source behaves as a physical context source and can be linked to the context-aware application.

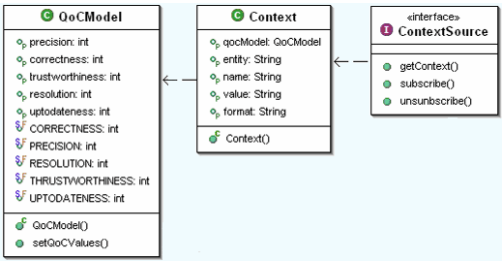

Figure 6. Interfaces

A ContextSource offers the two required invocation mechanism: request-response (getContext()) and subscribe-notify (subscribe(), unsubscribe() and notify() method of a client callback object). Both mechanisms deliver Context which consists of the elements described in the meta-model.

Figure 7 indicates the functional decomposition of the framework. It consists of different models that build up a SimuContext source and the information flow that exists between these models.

The ServiceModel implements the two required invocation methods: request-response and subscribe notify. The user can request context from here or can subscribe to context changes. The ValueModel implements the value generator that produces the values of the context samples and the values of the QoC parameters. The EventModel implements the event generator that produces the events when the sub-scribed user to this context source should be notified. The QoCModel implements the quality parameters of the delivered context sample. Both the ValueModel and the EventModel can be easily extended with application specific models like "Random-ValueModel" and "RandomEventModel" which generate random values or events at random times, respectively.

To configure SimuContext sources, we defined an administrator interface (SimuContextAdmin) (see Figure 8). It identifies methods to specialize the valuemodel and eventmodel to a user specified model (specialiseValueModel(), specialiseEventModel()).

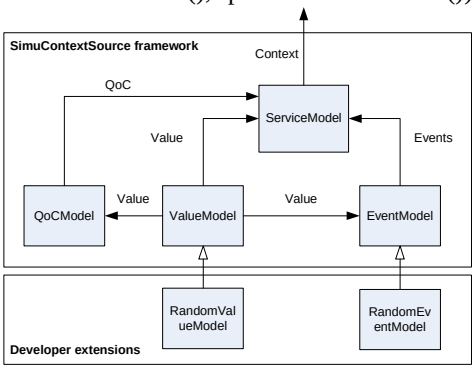

Figure 7. Functional decomposition

Furthermore, the eventmodel can be started and stopped. The configureSimuContextSource method enables the user to configure the SimuContext source using a configuration file.

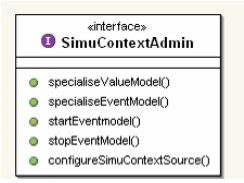

Figure 8. SimuContextAdmin interface

4. Implementation

We implemented the framework in Java. The framework is packaged as a standard Java library that can be included as a class library when developing CA applications.

For easy configuration and testing of SimuContextSources, we developed an administrator application with a graphical user interface (see Figure 9).

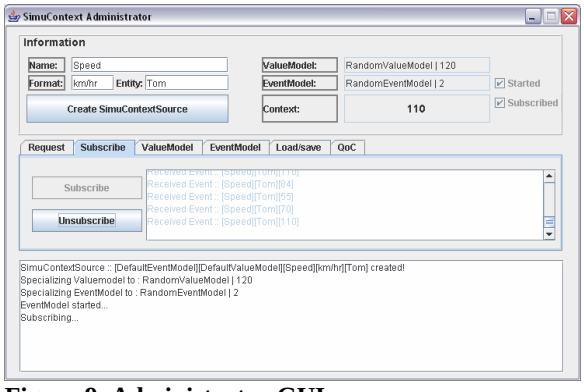

Figure 9. Administrator GUI

With the administrator application, developers can specify the characteristics of the context source they

want to simulate. They can test if the specified SimuContext source behaves like they need for their application. Finally, they can save the specification in a configuration file that can be read by the SimuContext framework linked to their CA application.

We implemented some basic eventmodels and valuemodels that application developers can use out-of-the-box: Random and GUI Event and Value models, Periodic Event model and Counter Value model.

In addition to these ready made models, the framework allows for application specific models. New valuemodels must implement the ValueModel interface. The getValue() method implements the value generator. Furthermore, the framework should be able to set variables and read the description of the variables. New eventmodels must implement the EventModel interface and inherit the AbstractEventModel. This includes implementing the event generator with the generateEvent() method. Again, the framework should be able to set the needed variables and read the description of the variables. For descriptions of the interfaces that need to be implemented, see Figure 10.

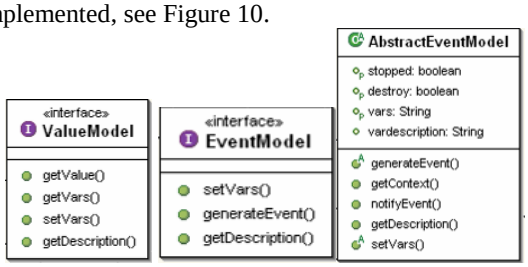

Figure 10. Interfaces

5. Discussion

By applying SimuContext sources, developers can abstract from the complexity of developing 'life' context sources for testing and development purposes. This enables them to focus on the application logic of their CA application. They can spend more time on developing their application.

Deploying SimuContext sources is simple and only requires definition of configuration files by hand or using the administrator GUI. Additionally, some custom models can be added by extending the provided interfaces and implementing their corresponding methods. The framework is integrated in the CA application as a class library and with some simple method calls the SimuContext source is instantiated and ready for use.

After developing the CA application and its corresponding SimuContext sources developers can use them to test, validate and demonstrate their CA application. Testing and validation becomes easy and realistic because the parameters of a SimuContext source can be modified in a controlled way. By modifying the parameters the (boundary) behavior of the CA application can be tested and validated. For example, when a 'log file' would be applied as valuemodel reproducible tests can be performed. Furthermore, when the SimuContext sources are configured more realistic (i.e. dynamic eventmodels and valuemodels that simulate a 'life' context source) also the application can be tested and validated in a semi-realistic setting. Additionally, by reconfiguring SimuContext sources the CA application can be tested for a multitude of context sources.

Once the CA application has been sufficiently tested and validated, it is relatively easy to replace the simulated context sources with real context sources, provided that these context sources implement the common ContextSource interface.

6. Related work

Several initiatives aim to facilitate application developers in coping with physical context sources, often called context managers, like the Context Toolkit [10], JCAF [11] and PACE [12].

There exist several semi-realistic simulation environments of which we will give some examples in this section. However, to our knowledge, context simulation suites are not existent.

Bylund [5] discusses a tool that interactively simulates context information in real-time. Their tool, called QuakeSim, uses the popular game engine of Quake III Arena to simulate a 3D environment. In this environment virtual persons can move and interact with other persons or the environment itself. The game engine provides the context information of these virtual persons which can be used as simulated information for CA applications.

UbiWise [13] uses similar technology as QuakeSim. It simulates a 3D environment to simulate ubiquitous environments which include prototyping of new devices and protocols. The simulator focuses mainly on computation and communication devices.

3DSim [14] provides a tool for rapid prototyping Ambient Intelligent applications. It uses a 3D based virtual environment to represent ambient devices. Context events are passed to the system with a TCP-based eventing interface.

Morla [15] discusses a simulation environment for location-based systems. They focus on component interaction, networking and location changes. Their environment supports the distribution of context events generated by distributed simulators using Web Services.

In contrast to the previous initiatives, SimuContext is a context simulation suite that enables the user to specify the behavior of context sources in stead of simulating an environment where the context is inferred from. In simulation environments, context changes are produced by interaction of a user with the environment (e.g. movement). SimuContext can be less attractive for live demonstrations (i.e. not a 3D GUI), however the simulated context is better controllable and reproducible. Additionally, testing and validation in an automated manner is more convenient. Furthermore, SimuContext is a context-centric approach while some of the related approaches focus on pervasive device and network aspects and use/provide context as a side-effect. SimuContext offers a generic light-weight approach that focuses on context simulation which is based on a robust context model.

7. Conclusions and future work

Context, delivered by context sources, becomes very important in pervasive environments. However, developing context sources can be cumbersome and time consuming. Furthermore, testing and demonstrating context-aware applications with physical context sources in a controllable and reproducible manner is difficult. This paper presents the SimuContext framework, a context source simulation framework, which facilitates developers in creating CA applications without losing scarce time on interfacing with physical context sources for testing and demonstration purposes. The framework enables them to easily specify context sources that will be simulated. These simulated context sources emulate the behaviour of life context sources. For operational context-aware application physical context sources are needed. However, for testing and demonstration purposes the SimuContext framework is a valuable tool.

Our SimuContext framework contributes to further advance research on context-aware applications, providing an easy to use context simulation suite. We envision the following future extensions of the SimuContext framework:

- *Subscription mechanism:* enhance the subscription mechanism with rules such that the user can specify a condition on which an event should be generated. Furthermore, this condition should incorporate QoC parameters such that only a notification is generated when the specified QoC requirements are met.
- *QoC model:* extend the framework such that other QoC models can be plugged-in like cost models.
- *Context operational issues:* context sources exhibit also some operational challenges such as sudden connection loss or the appearance of similar context sources (maybe with better quality). Future extensions of the framework should incorporate these challenges.
- *Security*: context is privacy sensitive information. Therefore, security should be taken into account as part of the context meta-model.

References

1. Lieberman, H. and T. Selker, *Out of context.* IBM System Journal, 2000. **39**(3): p. 617-632.
2. Broens, T., *Context-aware, Ontology based, Semantic Service Discovery*. 2004, University of Twente: Enschede.
3. Bunningen, A.v., L. Feng, and P. Apers. *Context for Ubiquitous Data Management.* in *International Workshop on Ubiquitous Data Management (UDM'05).* 2005. Tokyo.
4. Christie, A., *Simulation: An Enabling Technology in Software Engineering.* CROSSTALK Journal of Defense Software Engineering, 1999.
5. Bylund, M. and F. Espinoza, *Testing and Demonstrating Context-Aware Services with Quake III Arena.* Communications of the ACM, 2002. **45**(1): p. 46-48.
6. Schilit, B., N. Adams, and R. Want, *Context-Aware Computing Applications,* in *IEEE Workshop on Mobile Computing Systems and Applications*. 1994: Santa Cruz, CA, USA.
7. Bucholz, T., A. Kupper, and M. Schiffers, *Quality of Context: What It Is And Why We Need It,* in *Workshop of the HP OpenView University Association 2003 (HPOVUA 2003)*. 2003: Geneva.
8. Robinson, P., H. Vogt, and W. Wagealla, *Some research challenges in Pervasive Computing*, in *Workshop on Security and Privacy at the Pervasive 2004 Conference*. 2004: Munchen, Germany.
9. Dey, A., *Providing Architectural Support for Context-Aware applications*. 2000, PhD thesis, Georgia Institute of Technology.
10. Dey, A., *The Context Toolkit: Aiding the Development of Context-Aware Applications*, in *Workshop on Software Engineering for Wearable and Pervasive Computing*. 2000: Limerick, Ireland.
11. Bardram, J., *The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications*, in *Pervasive Computing*. 2005: Munchen, Germany.
12. Henricksen, K., et al., *Middleware for Distributed Context-Aware Systems*, in *DOA 2005*. 2005, Springer Verlag: Agia Napa, Cyprus.
13. Barton, J. and V. V., *UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment*, in *HP Labratory Technical Report HPL-2002-303*. 2002.
14. Shirehjini, A. and F. Klar, *3DSim: Rapid Prototyping Ambient Intelligence,* in *sOc - EUSAI*. 2005: Grenoble, France.
15. Morla, R. and N. Davies, *Modeling and Simulation of Context-Aware Mobile Systems.* IEEE Pervasive computing, 2004: p. 48-56.