CAESSA: Visual Authoring of Context-Aware Experience Sampling Studies

Mirko Fetter, Tom Gross
Human-Computer Interaction Group
University of Bamberg
96045 Bamberg
<firstname.lastname>(at)uni-bamberg.de

Maximilian Schirmer
Faculty of Media
Bauhaus-University Weimar
Bauhausstr. 11, 99423 Weimar
<firstname.lastname>(at)medien.uni-weimar.de

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
In this paper we present a toolkit that enables HCI practitioners to visually author and setup Context-Aware Experience Sampling studies—CAESSA (Context-Aware Experience Sampling Study Authoring).

Author Keywords
Experience Sampling Method; Context Awareness; Visual Editor.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces — Graphical User Interfaces, User-Centred Design; H.5.3 [Information Interfaces and Presentation]: Group and Organisation Interfaces — Computer-Supported Cooperative Work.

General Terms
Human Factors.

Introduction
A reoccurring challenge for designers in the fields of Ubiquitous Computing and Computer-Supported Cooperative Work is the need to gather user requirements or evaluate their systems in the field rather than in laboratory settings.

The Experience Sampling Method (ESM), derived from social psychology [2], has been successfully applied and adapted to research in these fields to meet this challenge. Through repetition, the ESM enables researchers to capture rich and in-depth data that describes a persons inner states or feelings in the moment they occur. It therefore has been used to study different questions around mobile phone usage and for evaluating ubiquitous computing applications [1], to get insights into the interruptibility of managers [5], as well as for acquiring labels for building predictive user models [7].

Intille et al. [6] improved the approach by introducing the concept of Context-Aware Experience Sampling (CAES) where sensor events trigger the presentation of specific questions. This way the sampling can be concentrated on situations or activities of interest, thus reducing the effort for the user. Additionally, the sensor data can be collected and analysed to give further insights.

Still the authoring of such studies is a considerable effort involving programming and scripting skills to setup the study. In this paper, we present CAESSA, a toolkit that enables HCI researchers and practitioners to setup complex CAES studies by means of an easy to use visual editor. It allows researchers the easy setup of complex CAES configurations involving multiple sensors.

Related Work

Several tools have been developed to support CAES, with the CAES-Tookit by Intille et al. [6] being the first. The CAES-Tookit is developed for PDAs and supports a variety of question types and allows for answers via the text input, microphone and camera. Beyond an implemented GPS sensor, a heart-rate monitor was planned. New studies can be loaded onto the device via a comma-delimited text file. The MyExperience (Me) Toolkit [1] for Windows Mobile devices was inspired by Intille et al.'s work and advances the authors' own work on the iESP tool. The freely available software supports 50 built-in sensors and allows for different sampling strategies, question types, and answering modalities. Studies are authored via an XML document that contains the questions as well as scripts that trigger the different questions.

While these tools offer excellent support for the participants of such CAES studies, the authoring of these studies can be quite challenging for the researcher. For example, the Me Toolkit requires the author to write Java-like code into the XML file to generate a sensor-based trigger (e.g., eleven lines are needed to trigger a question when the heart-rate monitor exceeds a specific threshold). More complex setups (e.g., with several sensors) seem to require substantial programming skills.

CAESSA therefore focuses on the easy, visual creation and authoring of complex CAES study setups, that also allows for an explorative approach.

CAESSA Concept

CAESSA presents opportunities for reducing the effort for participants and authors, collecting labels for machine learning, and conducting usability testing for ubiquitous and desktop applications. It allows researchers to conduct studies with context-triggered ESM that only samples during user activities of interest. This results in less burden for the participants, compared to regular ESM studies. It can also be used to collect la-

Sensors	Data
Active Access Point	1×1
Active Chats	1×1
Active Network Interfaces	n×1
Ambient Light	1×1
Application Focus	1×1
Applications	n×1
Battery	1×n
Bluetooth Devices	n×1
Calendar	1×n
Connected FireWire Devices	n×1
Connected USB Devices	n×1
CPU	1×n
Email	1×n
Ethernet Connected	1×1
Face Detection	1×1
Focus Title	1×1
Headphones Connected	1×1
Input Idle	1×1
IP Address	1×n
Motion	n×1
Mounted Volumes	1×n
Mouse Connected	1×1
Power Connected	1×1
Screensaver Active	1×1
Second Monitor	1×1
Skype	1×n
Time	1×n
Voice Activity	1×1
Volume Settings	1×n
Wi-Fi	n×m

Figure 1. Available sensors.

bels for machine learning and can help to find transition points between states for activity recognition. For usability testing, CAESSA allows to derive insights for the requirement analysis and evaluation of ubiquitous computing systems. The extensibility of CAESSA allows integrating own sensors that capture application usage on desktop applications and so helps to conduct usability testing for regular desktop applications.

Carrying out CAES studies can involve a large number of sensors that are interconnected in complex setups. With current tools, ESM researchers require extensive scripting skills in order to generate the necessary logic that processes the incoming sensor data in order to trigger a sampling.

In our approach, the process of discovering and using sensors, and integrating them into a configuration representing the study's logic is greatly eased by means of visual authoring. Authors of studies select sensors visually, inspect their current state and sensor values, and design the study logic that triggers samplings with a graphical user interface. This reduces the complexity of traditional approaches that involve extensive scripting or programming. All required components (i.e., sensors, logic, and samplings) can either be local on the device or distributed.

CAESSA Realisation

CAESSA is composed of three parts: A daemon for collecting sensor data and its corresponding GUI; an editor for authoring the flow of the event stream from the individual sensors via inference mechanisms to trigger question actuators; and a question actuator daemon for presenting the questions to the user in form of a dialog. All parts are connected by the underlying event-based sensor platform Sens-ation [4].

The CAESSA Sensor Daemon collects data from laptop and desktop computers. It runs as a background process, and currently supports 30 hard- and software sensors (cf. Figure 1) that are loaded via a plug-in mechanism and so can easily be extended with further sensors. The daemon handles pull sensors that are constantly polled with a defined sampling rate to deliver values (e.g., Bluetooth Devices and CPU sensor), and push sensors that publish their values on change (e.g., Mouse Connected and Voice Activity sensor). Via a GUI (cf. Figure 2), the researchers have a fine-grained control over the collection process by specifying parameters like the sampling rate of individual sensors or activating or deactivating individual sensors.

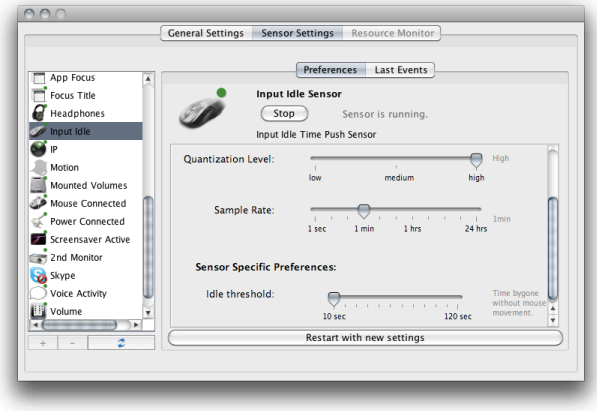

Figure 2. Sensor Settings tab of the CAESSA Sensor Daemon's GUI, with a list of the available sensors (left) and the settings pane for the selected Input Idle Sensor (right).

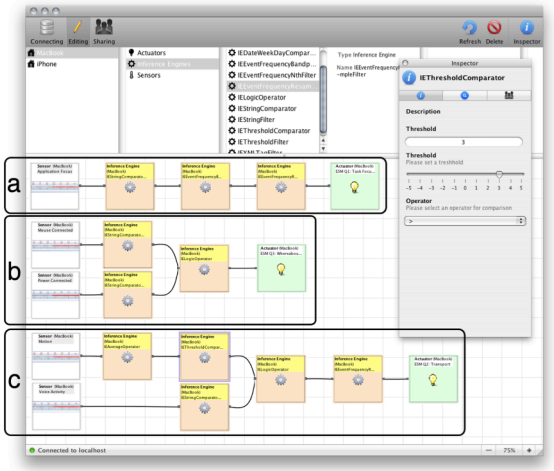

Figure 3. CAESSA Visual editor with scenario configurations.

Processors	Data
IEAverageOperator	V
IEDateIsWeekendComparator	M
IEDateWeekDayComparator	M
IELogicOperator	V
IEStringComparator	V
IEThresholdComparator	V
Filters	
IEEventFrequencyBandpassFilter	M
IEEventFrequencyNthFilter	M
IEEventFrequencyResampleFilter	M
IEStringFilter	V
IEThresholdFilter	V
IEXMLTagFilter	V

Figure 4. List of available value-based (V) and metadata-based (M) inference engines.

Each sensor reading is packed in an XML-based event description and made available to the further processing steps. Depending on the sensor, each event can contain a single value (1x1), multiple values of different types (1xn), list of values of the same type (nx1), or a table of values (nxm). The daemon saves the incoming sensor events to a sensor log and further directs the events to the next component, the CAESSA Visual Editor, via Sens-ation.

The CAESSA Visual Editor is based on the CollaborationBus Aqua editor for ubiquitous environments [9] and relies on the paradigm of visual programming [8]. It allows end-users to author rich configurations of CAES components with the help of an easy-to-use graphical user interface (cf. Figure 3). The editor represents all components (i.e., sensors, inference engines, and actuators) with abstract graphical user interface elements and assists users in the study setup process. Users instantiate components with drag-and-drop and create connections between them by drawing lines. An inspector window allows users to explore components, and presents configuration options for the 12 available inference engines (IE). Each inference engine encapsulates specific program logic and can be grouped into either Processors or Filters. Processors provide a computed result, based on a broad range of algorithms (e.g., for computing the moving average of sensor values over time). Filters block events that do not match the underlying filter criteria (e.g., filtering out sensor events that are below a critical threshold). We further distinguish value-based (V) and metadata-based (M) inference engines. Value-based inference engines use sensor values as input data, while metadata-based inference engines rely on additional metadata (e.g., occurrence time) of sensor events. Figure 4 shows an overview of the inference engines currently available in CAESSA, alongside with the classification.

The CAESSA Question Daemon is the third component of CAESSA. Triggered by an incoming event, the daemon presents the according question to the user with a simple dialog (cf. Figure 5). The daemon supports several kinds of question types: multiple choice (one/multiple answers), free text, numerical text, rating scale, and yes/no questions. Based on the question type, the actuator will render an adequate dialog, presenting the question and the answer modalities (radio buttons, text field etc.) to the users.

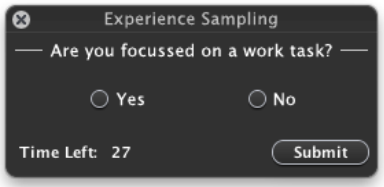

Figure 5. Screenshot of the CAESSA Question Daemon with a Yes/No question dialog example.

The dialog will be presented as the front-most window on the current screen. The users' input is published on a corresponding sensor and this way saved together with the sensor log. For each question, a timeout can be configured, after which unanswered dialogs will dis-

appear from the screen. Currently, the questions are created via a XML document. In the future, we plan to integrate the generation of new questions into the CAESSA Visual Editor.

Use Scenario

In order to illustrate the capabilities of CAESSA, we present a CAES scenario of a researcher investigating mobile laptop work behaviour of train commuters that can be easily authored with CAESSA. Figure 3 shows a study setup for this scenario, with three configurations (processing flow from sensors to actuator) of increasing complexity.

In the first configuration (cf. Figure 3a), our researcher is interested in finding out whether commuters that often switch between the browser and other applications are focussed on a work task. In order to achieve this, an application focus sensor is installed on a subject's laptop. The sensor is connected to a StringComparator to ensure that only focus events from a certain application (i.e., the web browser Firefox) are processed. An EventFrequencyBandpassFilter that is configured to pass on only sensor events that occur a given number of times within a given period of time is the next step in the logic to detect frequent switches. In order to reduce the number of times the question is shown, an EventFrequencyResampleFilter is configured. The Filter samples the event occurrence frequency down to one event each 30 minutes triggering the question actuator. The actuator presents a question to the subjects, asking if they are currently focused on a task. This example illustrates a simple sequential workflow with sensor events from a single sensor that are processed by several inference engines and trigger an ESM question actuator.

The second configuration (cf. Figure 3b) aims at detecting location changes of a subject. The researcher assumes that disconnecting all connected devices from a mobile computer is a strong indicator for an upcoming location change (i.e., the subject is leaving and going somewhere with the computer). Two sensors (Mouse Connected, Power Connected) are linked to StringComparator inference engines that check if the events from these sensors contain the word "disconnected". If the comparators detect this keyword, the connected LogicOperator inference engine triggers the question actuator that presents a question asking the subjects where they are leaving to. This example represents a more complex parallel workflow with several sensors and corresponding inference engines.

The third configuration (cf. Figure 3c) aims at determining how well a subject is able to work concentrated while on a usually crowded commuter rail. The question is triggered based on accelerometer data from a motion sensor and input from a voice activity sensor. The Motion sensor is an accelerometer in a subject's mobile computer that detects movement on three axes; it is connected to an AverageOperator inference engine that computes the moving average of the movement data over time. This inference engine is connected to a ThresholdComparator that determines if the average movement has reached a certain threshold, indicating that the subject's mobile computer is moving. Another sensor, Voice Activity, measures sound levels in the frequency band of the human voice. It is connected to a StringComparator inference engine that analyses the voice activity sensor's state. The processing chains of both sensors are combined by a LogicOperator inference engine that configured as a logical AND operation. Finally, an EventFrequencyResampleFilter samples the

occurrence frequency of the LogicOperator down to 1 event per 30 minutes and triggers the ESM question actuator that presents a question with a rating scale from "highly concentrated" to "highly distracted". This example illustrates a branched workflow.

Conclusion and Future Work

We presented CAESSA, a toolkit that supports researchers with the setup of CAES studies by providing an infrastructure and an editor that allows for visual authoring of such studies. How the approach compares to current setups that require intensive scripting and programming has to be shown with an evaluation with the finished system.

In the future, we plan to extend the system to also support mobile devices. This includes the collection of sensor data on mobile phones as well as the presentation of the questions to the user on the go, for example via Instant Messaging [3]. Furthermore, we plan to develop new IEs that extend the possibilities of the current system. For example, an IE is planned that allows taking previously answered questions as an input in order to allow contingency or follow-up questions. Additionally, placeholders will allow asking questions that reflect on specific sensor values (e.g., "Is the nearby Network %value accessible for you?").

An improved visualisation of the event data will simplify the live monitoring of studies and accordingly support researchers to adapt ongoing studies to better address specifics of different participants. Also, in order to simplify the generation of questions, a GUI will be developed that eliminates the current temporary solution to specify the questions via an XML document.

Acknowledgment

We thank all members of the Cooperative Media Lab.

References

[1] Consolvo, S., Harrison, B., Smith, I., Chen, M.Y., Everitt, K., Froehlich, J. and Landay, J.A. Conducting In Situ Evaluations for and With Ubiquitous Computing Technologies. Int. Journal of HCI 22, 1-2 (April 2007). pp. 103-118.

[2] Csikszentmihalyi, M. and Larson, R. Validity and Reliability of the Experience-Sampling Method. Journal of Nervous and Mental Disease 175, 9 (Sept. 1987). pp. 526-536.

[3] Fetter, M. and Gross, T. PRIMIExperience: Experience Sampling via Instant Messaging. In CSCW 2011. pp. (accepted).

[4] Gross, T., Egla, T. and Marquardt, N. Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures. Int. Journal of Internet Protocol Technology (IJIPT) 1, 3 (2006). pp. 159-167.

[5] Hudson, J.M., Christensen, J., Kellogg, W.A. and Erickson, T. "I'd Be Overwhelmed, But It's Just One More Thing To Do": Availability and Interruption in Research Management. In CHI 2002. pp. 97 - 104.

[6] Intille, S.S., Rondoni, J., Kukla, C., Ancona, I. and Bao, L. A Context-Aware Experience Sampling Tool. In CHI 2003. pp. 972-973.

[7] Kapoor, A. and Horvitz, E. Experience Sampling for Building Predictive User Models: A Comparative Study. In CHI 2008. pp. 657-666.

[8] Myers, B.A. Visual Programming, Programming by Example, and Program Visualisation: A Taxonomy. In CHI 1986. pp. 59-66.

[9] Schirmer, M. and Gross, T. CollaborationBus Aqua: Easy Cooperative Editing of Ubiquitous Environments. In CT 2010. pp. 77-84.