M. Koch, A. Butz & J. Schlichter (Hrsg.): Mensch und Computer 2014 Tagungsband,
München: Oldenbourg Wissenschaftsverlag, 2014, S. 195-204.

# CONTEX(T)PLORER: A Mobile Inspector for Ubiquitous Environments

Maximilian Schirmer[1], Sven Bertel[1], Ronja Lars Wilkening[2]

Usability Research Group, Bauhaus-Universität Weimar[1]
plazz entertainment AG[2]

**Abstract**

The complex invisible interactions between components of ubiquitous environments present a challenge for end-users and thereby raise concerns of privacy and nontransparency. In this paper, we will introduce CONTEX(T)PLORER, a mobile inspector system that aims at lowering the acceptance barrier of ubiquitous computing environments. The inspector actively involves end-users in an in-situ investigation process: users select, explore, and inspect components in a ubiquitous environment while interacting with them. Individual components are used as entry points to the process. We will present a brief overview of the system's concept and implementation, and will discuss related work.

## 1 Introduction and Motivation

Due to the continuous technological and conceptual advances in the fields of mobile devices, cloud computing, and context awareness, the visions of ambient intelligence and ubiquitous computing are turning into an everyday reality. For example, smartphones can act as ambient sensor nodes (Schirmer & Höpfner 2011) thanks to their increased processing power and capabilities. Furthermore, active research in the field of context awareness provides interesting concepts for automatically adapting user environments based on detected contexts (Brown, et al. 1997, Rittenbruch 2002, Schilit, et al. 1994, Schmohl & Baumgarten 2008).

In this paper, we will repeatedly refer to the following ambient intelligence *scenario*: A group of co-workers whose workplaces are scattered throughout a large office building like to spend their coffee breaks together. Inviting everyone or making appointments is not practical for this group and has proven ineffective in the past, for instance as suitable times for breaks are likely to vary depending on employees' activities and schedules. Instead, smart ubiquitous technology installed in the coffee break room detects the number of people present and their current activities. When pre-defined people count and activity levels are reached, all interested “coffee breakers” are automatically notified. Also, the fully-automated coffee machine is triggered to prepare additional coffee for all participants that are expected to join. All ubiquitous environment components are tagged with QR codes for identification. One day, Jane, a new employee, joins the group and is baffled by the electronic components installed in the coffee kitchen. She is concerned by the components’ potential to monitor activities of employees and has strong privacy concerns.

This scenario illustrates that, due to the experimental and advanced nature of the smart, interconnected components, and even with only a small number of components, the data flow in a ubiquitous environment can quickly become complex and overwhelming for the end-user. Such complexity can easily make it difficult to understand and oversee the results of their actions within the environment, to the effect that, for end-users, the embedded system and its functioning become nontransparent. Privacy concerns may arise, for example, due to embedded systems tracking the whereabouts of users or disclosing this information. Concerns resulting from such nontransparency (Weiser, et al. 1999) constitute a high acceptance barrier for ambient intelligence and ubiquitous computing. Lowering this barrier marks the first necessary step towards more widespread acceptance of such concepts and technologies.

In this paper, we will present the concept and a first prototype for CONTEX(T)PLORER, a mobile system that allows users to learn about the current state of a ubiquitous environment through component-based in-situ investigation. We favour this in-situ approach because it lowers the level of knowledge required to understanding which components are relevant when users interact with a ubiquitous environment. The system’s investigation process aims at increasing the transparency in ubiquitous environments for their users by revealing the environment’s components and interactions between them. It is built to support an interaction process with three steps: (1) *selection* of components of interest to a user, (2) *exploration* of the selected components’ composition with respect to the structure of the environment, and (3) detailed *inspection* and manipulation of specific components.

The paper is structured as follows: Section 2 will present the main conceptual aspects of CONTEX(T)PLORER. Section 3 will introduce the prototype implementation and discusses aspects of user interaction with the system. Section 4 will give an overview of related work. Section 5 will summarise the paper and provide an outlook on future work.

## 2 Interaction Concept

CONTEX(T)PLORER is a mobile system that allows in-situ configuration and exploration of ubiquitous environment components at the same time at which users are interacting with them. It focuses on short-term, target-oriented configuration tasks that are conducted with on-site access to the components. Our main motivation for designing this system was to alleviate privacy concerns by allowing users in a ubiquitous environment to transparently learn about available components and their interconnections. We believe that in order to incorporate privacy aspects into ubiquitous computing, a seamless interplay of social norms, legal protection, and technology is required. An early concept in ubiquitous computing, feedback and control (Bellotti & Sellen 1993), is often stated as the happy medium towards privacy-aware ubiquitous computing. For us, feedback in this context means to inform users about information flows, to inform them which information is currently being disclosed, and to whom. Control on the other hand is supposed to put users into the position to define which information they want to disclose, and to whom.

In the scope of this paper, we will use the following types of ubiquitous environment component types: (1) sensors, (2) processors, and (3) actuators. Sensors gather data about users and their environments; processors evaluate sensor data and generate output based on aggregation or filtering; and actuators initiate reactions in the environment. Typically, these components are managed by sensor platforms or event-notification infrastructures (e.g., (Carzaniga, et al. 2001, Fitzpatrick, et al. 1999, Funk, et al. 2011, Gross, et al. 2006)).

From a technological point of view, the scenario introduced in the previous section may likely be realised based on movement and noise sensors in the coffee break room, and based on calendar sensors on the participants' computers. Processors in the form of threshold comparators (for movement and/or noise) and a calendar event parser aggregate and analyse incoming sensor data in order to determine activity in the room, as well as degrees of availability of individual participants. Based on the processors' results, email notifications and a control unit for the coffee machine are then triggered as actuators.

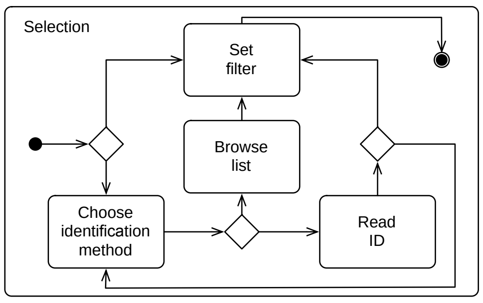
The new employee Jane uses CONTEX(T)PLORER to identify the ominous components in the coffee kitchen, and to learn about their interconnections as well as about the data flow between them. Because the system runs directly on her smartphone, she can experiment and interact with the components in-situ, and directly see the effects of her actions in the graphical user interface of CONTEX(T)PLORER.

## 2.1 Building Blocks

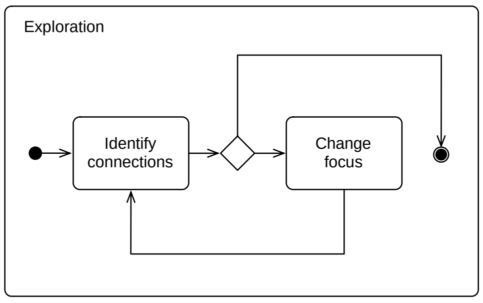
CONTEX(T)PLORER consists of three main conceptual building blocks. Based on our informal observations of typical interactions with sensor platforms and ubiquitous environments, we structured them into the three steps *selection*, *exploration*, and *inspection*. The steps occur in the sequence that we describe here, and each step's results influence the following steps. We understand this sequence as a form of applying a drill-down navigation paradigm from traditional graphical user interfaces to an interaction process that starts in the real world and ends with fine-grained insight and control of the inner workings of smart components in a ubiquitous environment.

*Selection* is a natural starting point for the interaction with environment components. It enables users to detect all available components in the environment and to define their personal components of interest by communicating their interest to the inspector system. The selection process is realised by scanning and interpreting component identifiers, enabling users to actively define their entry points to the subsequent exploration process. In our *scenario*, Jane selects one of the sensor components in the coffee kitchen by scanning their QR code. That way, she does not require any knowledge about the component, other than where its location is marked in the environment.

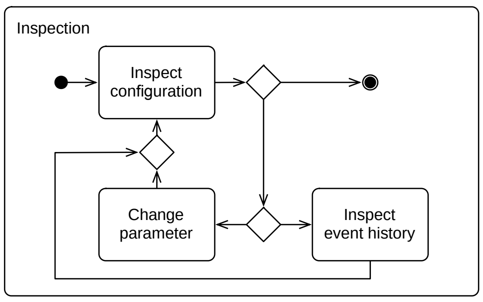
The process of selecting components is shown in Figure 1(a) and employs one of a wide range of possible component identification methods. Often this is realised as browsing a list of available components, entering an identification number, or using a scanning method. Setting filters to confine the search space simplifies the selection process. Such filters can restrict the set of components by type, location, function, or by other relevant parameters.



(a) Selection



(b) Exploration



(c) Inspection

*Figure 1: Activity diagrams of the interaction steps.*

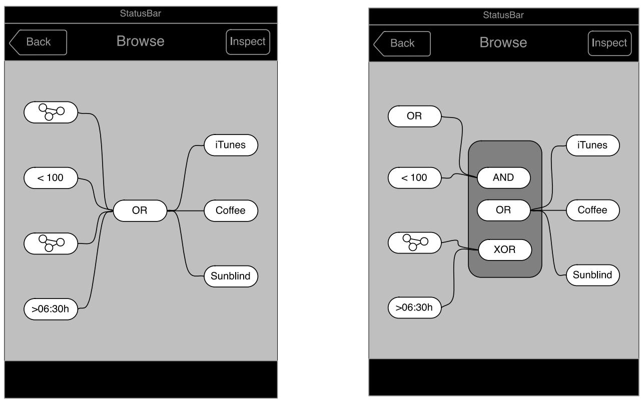
*Exploration* helps users to learn about the environment's structure and the transitive interconnections between components by visualising all relevant components. Components can also be clustered in order to abstract from their complexity. In the exploration process, users are able to analyse the composition of these logical clusters of components. During the process, users incrementally shift the visualisation's focus until they reach a desired component for closer inspection (cf. Figure 1(b)). Reflecting back on our *scenario*, Jane has selected a noise sensor in the coffee kitchen in the first step. She is now able to see all outgoing connections from this sensor. By setting the focus to one of the connected threshold processors, she ultimately explores the complete data flow from her sensor up to an actuator at the end of the processing chain. In essence, exploration employs selection to give users the opportunity to shift focus between several components of interest.

*Inspection* allows users to view the current component values as well as the history of events associated with this component. Furthermore, configuration parameters are checked and configured in order to change the environment's behaviour (cf. Figure 1(c)).

In our *scenario*, Jane has reached a desktop notification actuator that sends messages to the computers of co-workers who subscribed to be notified when the conditions seem appropriate for the initiation of a coffee break. Of course, the actuator does not expose the identities of the subscribers. From the history log of events, she learns that the default setting of the actuators sends out repeated notifications until users either accept or decline the invitation. She finds this setting too annoying and configures her actuator to show only a small notification window that disappears after a few seconds.

## 2.2 Graph Visualisation and Animation

CONTEX(T)PLORER provides a rich graph visualisation of components and their interconnections. In the visualisation, we distinguish two types of graph nodes: cluster nodes and regular nodes. Cluster nodes encapsulate a number of other nodes and thereby provide abstraction or grouping of environment components (cf. Figure 2).



(a) Regular nodes (b) Cluster node

*Figure 2: Node types in the graph visualisation.*

At any time, the visualisation provides a single logical viewing frame, consisting of a focus node in the centre, and neighbouring nodes positioned to the left and right of it. If the focus node is a cluster node, all other nodes that have connections to nodes outside of the cluster are displayed inside of the focus node. This helps to visualise the data flow between this cluster and other clusters. Users then change focus by pressing on any one of the displayed nodes, the inspector then displays this node’s logical viewing frame. As we have previously introduced, changing focus is an essential part in our interaction process. That is why we have used animations to provide users with a clear understanding of the transitions that occur from one focus to another. Foremost, this means that users should be able to see which nodes are identical in two consecutive logical viewing frames. Figure 3 outlines the animation steps during a change of focus. Shaded nodes are visible in the start view, but not in the finish view. Dotted nodes are spawned and white nodes remain visible throughout the sequence.

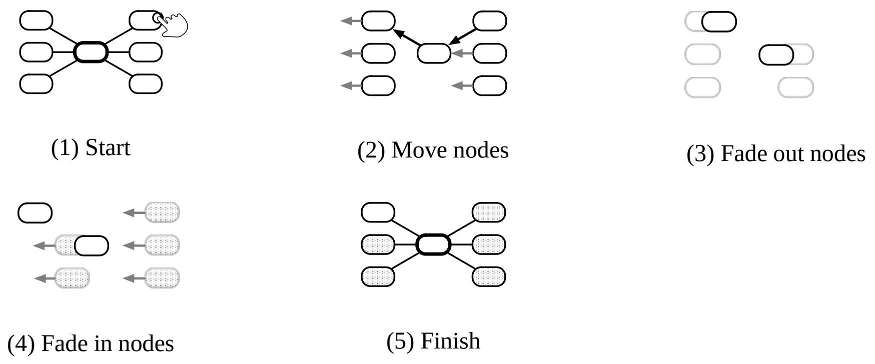


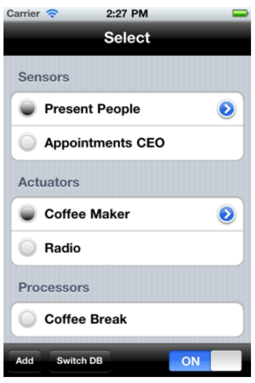
*Figure 3: Sequence of animation steps during focus change.*

During the animation, an already visible node is moved to the centre. This node may be positioned to the left/right of, or inside (cluster) the old focus node. Figure 3(1) points out that a user selects a neighbouring node to the right of the current focus node. The position of the selected node determines the movement direction of the animation (cf. Figure 3(2)). The previous focus node assumes its position at the left or right border, while all nodes that are not part of the new logical viewing frame move out horizontally in the animation direction. At the same time, they fade out visually (cf. Figure 3(3)). Figure 3(4) shows how new nodes fade in while they move along the animation direction to their position in the logical viewing frame. Finally, the result of the animation is shown in Figure 3(5) and provides the final arrangement with the new focus node and all neighbouring nodes.

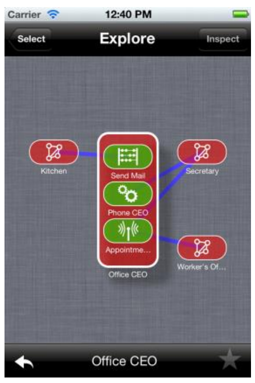
## 3 Prototype and Walkthrough

The CONTEX(T)PLORER prototype is realised as a mobile application on the Apple iPhone platform. We have deployed and successfully tested the application on iPhone 3GS, iPhone 4 and iPhone 5 devices. CONTEX(T)PLORER possesses a modular architecture with modules for *selection*, *exploration*, and *inspection*, and an adapter module responsible for the connection between the graphical user interface and the data model. The sequence of these modules follows a task model derived from our informal observations of typical interactions with sensor platforms and ubiquitous environments. Currently, we can either use an SQLite database that simulates structural information (i.e., components with parameters and values, interconnections, and clusters) about environments, or our lightweight smartphone sensor data distribution platform. The platform collects sensor data from smartphones and provides a socket-based interface for accessing the gathered sensor data. Due to the modular nature of the adapter pattern that we employed in the implementation of CONTEX(T)PLORER's data model, we can connect to existing sensor platforms with manageable effort. It is also possible to combine collections of components from different platforms as a single data model.

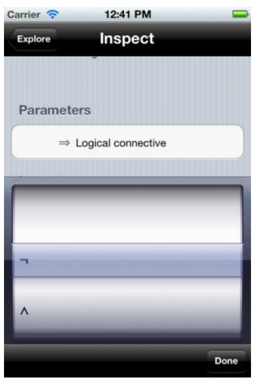
As shown in Figure 4(a), users initiate the *selection process* from a structured list of components. Typically, this list is already populated with previously discovered components. Users add nearby components by scanning their QR codes. Each environment component is associated with a QR code as visual identifier. Users can also add components by browsing the complete list of all available sensors, processors, and actuators (provided by the corresponding sensor platform). After users have assembled their individual list of components, they can mark particular components of interest. These components are then highlighted to indicate that they are included in the exploration process. Users can also choose to ignore such highlighting and include all components in the list by toggling a switch button. In our *scenario*, Jane wonders what kind of equipment all the strange components in the kitchen are. She scans the QR code of a noise sensor and begins to explore.



(a)



(b)



(c)

*Figure 4: Graphical user interface of CONTEX(T)PLORER.*

The *exploration process* as shown in Figure 4(b) provides a dynamic graph visualisation with animations. Animations help users to keep track of components and connections while navigating through the graph structure. Each node in the graph represents an environment component or an aggregated cluster of components. Nodes are labelled with identifiers and provide icons according to their type. When users tap a node in the graph, the view centres on this node and its connected neighbours. Additionally, the centred node is highlighted by a white border and shadow. If the centred node is a cluster, the enclosed components with connections to nodes outside of the cluster are revealed. At any time, users can navigate backwards along their exploration path with the help of a "back button". Furthermore, a star-shaped button allows users to mark components they have discovered in the exploration process to be included in their own component list. Following our previous *scenario*, the coffee kitchen is a cluster of components (movement/noise sensors, coffee machine actuator), with connections to other clusters for participants' offices (calendar sensors, email notification actuator), and to a cluster of processors (activity, availability).

Users start the *inspection process* (cf. Figure 4(c)) by navigating to a component node and tapping the "Inspect" button. The application presents an inspection view with meta information about the selected component (i.e., name, description, and icon), a list of parameters, and a list of inputs and outputs with their current values. Tapping on one of the list items reveals additional details. For parameters, a configuration interface with text fields and drop-down lists of possible configuration values is presented. For inputs and outputs, the application shows a list of received or sent values with timestamps. In our *scenario*, Jane inspects the notification actuator and sets her personal notification preferences. The user interface elements in the inspection view correspond to the complexity of the inspected components. For example, simple processor components may only require sliders or drop-down menus, while more complex ones may require the user to make changes to a script using a text field.

## 4 Related Work

CONTEX(T)PLORER is closely related to the field of end-user configuration of ubiquitous environments. Within this active field of research, several interactive editor applications exist that allow users to learn about the components in their environment, and to actively change the configuration of these components. The *Speakeasy Browser* (Newman, et al. 2002) is a text-based editor that allows the configuration of various environment components by forming connections between available modules, which are visualised in a tree structure. The editor follows the paradigm of a file browser and is realised as a website, so that it can also be accessed by mobile devices. In contrast to CONTEX(T)PLORER, the editor does not provide any graphical abstractions. The *Jigsaw Editor* (Humble, et al. 2003) puts great emphasis on an easy-to-use, abstract graphical user interface. Available environment components are visualised as jigsaw pieces. However, this simplified user interface also limits its capabilities. In contrast to CONTEX(T)PLORER, the editor does not allow users to filter available components, and it cannot be used in-situ on a mobile device. The *CollaborationBus: Aqua Editor* (Schirmer & Gross 2011) is a graphical editor for ubiquitous environments that uses a graph-based visualisation of the interconnections between components. It finds similarities between environment configurations. In contrast to CONTEX(T)PLORER, the editor cannot be used in-situ on a mobile device, does not cluster components and does not provide component filtering. CONTEX(T)PLORER is also related to the concept of the inspector window in graphical user interfaces: a special palette window found in a wide range of desktop applications. Mostly used in productivity tools, the inspector provides information and manipulation of an entity's properties. The content of the inspector changes adaptively with the currently selected entity. Users typically select an entity in their productivity environment (e.g., a shape in a vector drawing application, a sound clip in an audio editor, or a piece of text in a text editor), the inspector window reacts to this selection and displays information and settings for this entity. Users then explore their options and learn about their selection, before they finally interact with the settings to manipulate their selected entity. This behaviour served as a model for the interaction process of CONTEX(T)PLORER, and we aimed to translate it to a mobile platform. Our goals behind introducing a dedicated *inspection* building block that helps rendering ubiquitous technical components and their interrelations less opaque for the user are comparable to goals behind rendering automatically generated user models and user profiles less opaque for the modelled/profiled user (e.g., (Ahn, et al. 2007, Carmichael, et al. 2005)). The goal of *scrutability* (Kay 2006) is comparable to our goal of *transparency*.

## 5 Discussion and Outlook

In this paper, we introduced CONTEX(T)PLORER, a mobile inspector for ubiquitous environments. The inspector is based on a concept for a three-fold investigation process that allows users to explore an environment *in-situ*, while interacting with its components. We presented our first implementation of the concept and described its user interaction. The implementation provides a modular architecture with an adapter interface for connecting sensor platforms. CONTEX(T)PLORER goes beyond other currently existing approaches in that it combines graphical, user-explorable abstractions of components' interconnections with display filtering options (e.g., based on an end-user's preferences) and a tiered investigation process. This functionality is realised on a mobile platform for in-situ use.

The current QR-code-based identification mechanism in our prototype system clearly collides with the intended invisibility of ubiquitous computing devices; it has been chosen purely for our proof of concept. We will explore additional identification methods, for example indoor positioning or beacon-based methods (using Wifi or Bluetooth LE), as these methods involve far less explicit interactions. We are currently implementing and improving a sensor platform for using smartphone sensor data in ubiquitous environments that will soon become the primary data source for CONTEX(T)PLORER. Furthermore, we plan to pay closer attention to the interaction steps described in this paper. Specifically, one can question whether an *exploration* step always needs to follow a *selection* step, or whether both may not also occur in reversed or repeatedly alternating orders. We will conduct further user studies to test the hypotheses derived from our informal observations of the structure of the inspection process and to learn about usability issues our design might have.

For future work, we believe that it would be interesting to explore the possibility of using the gathered context data of the sensor platform in order to improve the selection phase. In the most simple use case, this could help to adapt the search space in a different way for novice users and expert users. Other options could be highlighting the most frequently used components or dynamically adapting the level of detail in clusters (e.g., based on different profiles of interest, contexts, or on different roles of end-users). Last, it seems useful to examine what kind of inspections best serve to alleviate (or substantiate) end-users' privacy concerns. For example, when nontransparency of the ubiquitous system's structure is at the root of privacy concerns, does it suffice to present some (selected) uses of the collected data (such as was the case for Jane, in our scenario above)? Or, would alleviating privacy concerns usually necessitate communicating all uses of the collected data? If the latter is the case, the inspection process presented here may be easily extended to include information on the user's progress of inspecting all the relevant components and their interconnections, guiding the inspection process by marking up those components and interconnections which the user has not yet inspected.

## References

Ahn, J. W., et al. (2007). Open user profiles for adaptive news systems: help or harm? *Proc. of 16th international conference on World Wide Web*, ACM, 11-20.

Bellotti, V., Sellen, A. (1993). Design for privacy in ubiquitous computing environments. *Proc. of ECSCW'93*, ACM, 77-92.

Brown, P. J., et al. (1997) Context-aware Applications: from the Laboratory to the Marketplace. *IEEE Personal Communications*, 4, 58-64.

Carmichael, D. J., et al. (2005) Consistent modelling of users, devices and sensors in a ubiquitous computing environment. *User Modelling and User-Adapted Interaction*, 15, 197-234.

Carzaniga, A., et al. (2001) Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19, 332-383.

Fitzpatrick, G., et al. (1999). Augmenting the workaday world with Elvin. *Proc. of Sixth European Conference on Computer Supported Cooperative Work - ECSCW'99*, 431-450.

Funk, A., et al. (2011). Open Sensor Platforms: The Sensor Web Enablement Framework and Beyond. *Proc. of 6. Konferenz Mobile und ubiquitäre Informationssysteme - MMS'11*, 39-52.

Gross, T., et al. (2006) Sens-ation: A Service-Oriented Platform for Developing Sensor-Based Infrastructures. *IJIPT*, 1, 159-167.

Humble, J., et al. (2003). "Playing with the Bits": User-configuration of Ubiquitous Domestic Environments. *Proc. of UbiComp'03*, 256-263.

Kay, J. (2006). Scrutable adaptation: because we can and must. *Proc. of Adaptive hypermedia and adaptive web-based systems, 4th international conference*, Springer, 11-19.

Newman, M. W., et al. (2002). Designing for Serendipity: Supporting End-user Configuration of Ubiquitous Computing Environments. *Proc. of DIS'02*, 147-156.

Rittenbruch, M. (2002) Atmosphere: A Framework for Contextual Awareness. *Int. Journal of Human-Computer Interaction*, 14, 159-180.

Schilit, B. N., et al. (1994). Context-Aware Computing Applications. *Proc. of First Workshop on Mobile Computing Systems and Applications - WMCSA'94*, 85-90.

Schirmer, M., Gross, T. (2011) Lightweight Editing of Distributed Ubiquitous Environments - The CollaborationBus Aqua Editor. *CollTech 2010 Proc.*, 2.

Schirmer, M., Höpfner, H. (2011). SenST: Approaches for Reducing the Energy Consumption of Smartphone-Based Context Recognition. *Proc. of CONTEXT'11*, 250-263.

Schmohl, R., Baumgarten, U. (2008). Context-aware Computing: a Survey Preparing a Generalized Approach. *Proc. of IMECS 2008*, 744-750.

Weiser, M., et al. (1999) The Origins of Ubiquitous Computing Research at PARC in the late 1980s. *IBM Systems Journal*, 38, 693-696.