

What's that hot thing in my pocket? SocioXensor, a smartphone data collector

G.H. (Henri) ter Hofte

Telematica Instituut, Enschede, the Netherlands

Henri.terHofte@telin.nl

Abstract. Mobile phones tend to travel along with people wherever they are and whatever they are doing. Contemporary smartphones can be fully programmed and have many types of sensors. We exploit this in the Xensor System, an extensible research software platform that can collect data about human behavior in the field. Running on a smartphone carried by a human subject, The Xensor System can sample subjective user experiences and can capture objective data about (social) context and application usage, at any time and in any location where subjects go. As such, the Xensor System is an addition to existing data collection methods such as surveys, interviews and lab experiments.

In this paper, we first present the design and implementation of the Xensor System in general and SocioXensor, RouteXensor and SeniorXensor in particular. Then, we describe the experiences we gained with SocioXensor in a study among 10 subjects that lasted 5 weeks. Finally, we compare Xensor in general and SocioXensor in particular to related in-situ data collection tools for studying human behavior.

Introduction

Mobile phone adoption in many countries is increasing, even approaching market saturation in some western nations. Many people habitually carry their mobile phones to virtually all places and activities. Technological progress increasingly gives mobile phones “smart” capabilities at an affordable price, most notably various sensors (e.g., for location, movement and vibration) and the ability to run third-party applications. These developments provide an unprecedented opportunity for social scientists to start using mobiles phones as a means to collect objective data about human behavior and the (social) context in which this takes place, together with subjective data collection techniques such as experience sampling (Csikszentmihalyi et al., 1987), under naturalistic conditions, at any time, in any location.

SocioXensor is an extensible research software platform that turns a person's mobile phone into a data capturing device for social scientists. Rather than bringing the people to the lab, SocioXensor helps to bring the lab to the people. SocioXensor does not intend to replace, but add to existing data collection techniques (see Figure 1). SocioXensor supports in-situ data collection methods logging and experience sampling, which can be used avoid or minimize problems of retrospective recall present in e.g. surveys and interviews. The experience sampling support provided by SocioXensor can be more obtrusive than logging, but is typically less obtrusive than more rich direct observation methods such as ethnography or lab experiments. At the same time, researcher involvement during data collection is minimal, which supports collecting data longer, at more places and/or for more subjects.

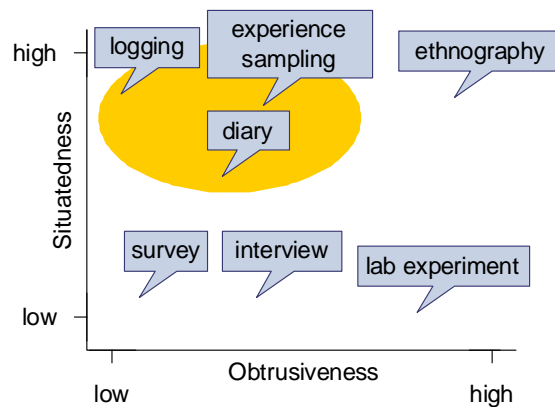


Figure 1. SocioXensor compared to other methods (Mulder et al., 2005)

Design goals

The SocioXensor concept (Mulder et al, 2005; Ter Hofte et al., 2006) was initially developed in the FRUX project, which considered experience sampling as a promising way to study user experience in general and user experience of mobile social software in particular. In an exploratory study into experience sampling (Ter Hofte et al., 2004), we used the CAES research tool (Intille et al, 2003), which satisfied our needs in many ways, but we also observed a few crucial limitations, which ultimately inspired us to design SocioXensor. Most notably, CAES took over the entire device and could not run alongside other applications, which makes it virtually impossible to do studies using a person's own mobile device such as a PDA or smartphone. Realizing that in-situ data collection using mobile (smart)phones carried by subjects can be applied to more phenomena than just social phenomena, we set out to design a more general in-situ data collection platform, the Xensor System. We structured the design goals for the Xensor System along four phases of a typical study: configuration, deployment, data collection and analysis, as summarized in the sections below (details can be found in (Ter Hofte et al., 2005)).

Configuration

Non-technical researchers should be able to configure the Xensor System to collect different types of data in a particular study, either by editing a simple text file, or via a graphical study configuration interface. In particular, the Xensor System should support:

- *Selecting various data collection techniques*
 - *Behavior and Context Logging:* The Xensor System should support automatic logging of raw, objective data about human behavior and the context in which it takes place, such as location, proximity to others, and communication with others. In particular, the Xensor System should be able to use mobile device technologies such as Bluetooth (e.g. to detect nearby devices), GSM radio, Wi-Fi radio, and GPS (e.g. to detect location), microphone, camera, call logs, contact and calendar data.
 - *Self-report:* More subjective information, such as opinions and feelings, are very hard to collect with automated logging. Therefore, the Xensor System should support self-report methods. In the diary method, a subject is asked to remember to initiate a self-report regularly and/or whenever a particular phenomenon occurs (e.g. whenever the subject feels lonely, or just before going to bed). In the experience sampling method, the prompt for a self-report is delivered by the system, e.g. according a researcher-defined (random) schedule, or based on the occurrence of system-detectable events. Self-reports may vary between a short audio recording to quick surveys defined by a

researcher to which the subject can reply via e.g. keys, screen, audio & photo and video.

- *Usage logging*: Evaluation of mobile or ubiquitous applications often needs to take place in-situ, i.e. under naturalistic conditions (Consolvo et al., 2007). In such evaluations, the usage (pattern) of the application under study often needs to be logged. Usage logging refers to recording the usage of a particular application under study, which may vary between recording key presses, screen touches and screen recordings to recording particular events of interest in an application only.
- *Extensible set of data collection modules*: We expect research needs to vary from study to study and device technology to progress. Hence, the Xensor System should be set up in a modular and extensible fashion. When the modules available no longer are sufficient, a programmer should be able to create and add a new one.
- *Configurable Xensor data collection modules*: For each data collection module, for each study the researcher should be able to configure:
 - *Active variables*: A single Xensor module may be able to collect various types of data. For example, A Xensor module for GPS may be able to collect data about location, speed, direction and altitude. In a particular study, a researcher may wish not to record all these types of data that, but only a subset, e.g., to protect privacy or save storage space. The Xensor System should support researchers to select subsets of variables.
 - *Sampling interval*: Xensor modules may provide the researcher with a configurable interval that defines how often a measurement should be taken. For example: an interval of 30 seconds for a GPS Context Sensor. Some modules may not provide a configurable sampling interval, or a minimum interval may apply. A sampling interval of zero is used to indicate: record as fast as possible, but only changes.
 - *Reporting interval*: Xensor modules may provide the researcher with a configurable interval that defines how often measurements that have been taken are reported to persistent storage (on the mobile device). This helps a researcher make a tradeoff between performance and robustness: a shorter reporting interval, increases robustness (data will not go lost when a device crashes), at the possible cost of less performance.
 - *Xensor module-specific settings*: In addition to configurable variables, sampling and recording intervals, Xensor modules may provide researchers with more configuration options. For example, the basic experience sampling module included with the Xensor System, the following elements should be configurable: questionnaire (textual questions, textual answer options for each question, which question follows when an answer is selected), minimum and maximum for a uniform random distribution of inter-sample time, notification mechanism (sound volume and/or vibration, or follow the phone profile), notification duration, notification timeout, notification retry time, number of retries and question timeout.
- *Study start and stop date and time*: A researcher should be able to configure a date and time at which a particular study starts and stops automatically. If no start date and time is set, the study will start as soon as deployed on a subject's mobile device. If no stop date and time is set, the study will stop as soon as the researcher stops data collection on the subject's mobile device.
- *Daily start and stop date and time*: A researcher should be able to configure a time at which a particular study starts and stops automatically for each day of the week.
- *Data upload policies*: A researcher may configure when data is uploaded to a central server that stores all data for all subjects in the study. Upload policies that should be supported are: at fixed times, at regular intervals, after a certain amount of data is collected, when remaining battery level is below a certain threshold, when remaining local storage is below a certain threshold, or whenever a certain network is available (e.g. Wi-Fi network, fixed connection to the internet via the device cradle, etc.)

Deployment

In the deployment phase, the study subjects are recruited, informed consent is obtained from the subjects and care is taken that subjects have suitable devices, enough local storage for collected data (e.g., an SD card) and the appropriate Xensor System software, and study configuration data. Moreover, the subjects are trained and informed about incentives. The Xensor System should support various deployment scenarios:

- *SD/cradle/wireless deployment*: Deployment by inserting a storage card or connecting the device to a cable is only suitable when the researcher can meet subjects face-to-face for deployment. Deployment via download initiated by accepting e-mail or SMS invitations, via wireless connections, such as Wi-Fi, or GPRS/UMTS, can also be used when the researcher does not meet subjects face to face for deployment.
- *Anonymous / non-anonymous subjects*: In some studies, identities of the subjects are known (e.g., e-mail address and name), in others anonymous (only subject ID is known).
- *Closed / open studies*: In some studies, all subjects are known before data collection starts, in others subjects can be added after data collection has started.

Data collection

During data collection, the role of the researcher is relatively limited, compared to other data collection methods such as interviews and observations. The researcher may need to maintain a helpdesk, in case the subjects experience problems with using the device, collect the devices at the end of the study and provide the incentives that may have been promised in the deployment phase. The role of the Xensor System in this phase, on the other hand, is quite extensive, and gives rise to the following design goals:

- *Non-interference when used on a subject's own mobile device*: Like any measurement instrument, interference of the Xensor System with the phenomenon it measures should be minimal, which implies it should be able to run on light, small, easily portable/wearable mobile devices such as small (smart)phones that researchers either can hand out, or that people already use and routinely carry with them everywhere. Its battery impact should not change the user's charging habit too much. In addition, it should work alongside other applications with minimal interference with these applications (e.g. only use screen estate when an Experience Sampler requires it)
- *Robustness*: The Xensor System should be able to work autonomously for a full day, which implies it should work without a network connection or under intermittent network connection conditions, data should be stored persistently, the data collection process should resume after a device crash or whenever the device is switched off and later on again. For those (rare) cases that Xensor is not operational nevertheless, it should provide a (minimal) indication that allows the user to check whether the study is still active and working without significant anomalies. This allows the user to recognize significant anomalies and contact the helpdesk.
- *Security*: Xensor should maintain the confidentiality and integrity of the data collected, i.e., make sure only authorized users can get access to the data on the mobile device, in transport or in the Xensor Repository.
- *Privacy*: In addition to proper informed consent, research ethics and Institutional Review Boards will often require giving subjects control over data collection:
 - *Suspend/resume data collection by subject*: The Xensor System should provide subjects with the means to suspend all data collection for a specified duration in the future, and resume the data collection later.
 - *Erase recent data by subject*: The Xensor System should also provide subjects with the means to erase all data collected for a specified duration in the past.

Analysis

In the analysis phase, the researcher retrieves all the data collected and analyses this data. This phase is beyond the support scope of Xensor System; it does not provide advanced analysis capabilities, but it will need to facilitate export of the data collected into formats understood by analysis tools. Successful analysis of the data however, requires additional design goals for data collection by the Xensor System:

- *Time stamping*: All data collected is time-stamped according to a sufficiently accurate local device clock. Clock drift on mobile devices and changes by subjects may require regular synchronization between local clocks and a central clock and/or recording the time skew between them.
- *Traceability*: The data can be traced back to a unique subject (which may be anonymous, or can be traced back to a unique person, depending on the study).
- *Completeness*: The data collected should be complete, including information about anomalies, such as non-responses.

Xensor System architecture and implementation

Based on the design goals described in the previous section, we defined an architecture for a Xensor System, as illustrated in Figure 2.

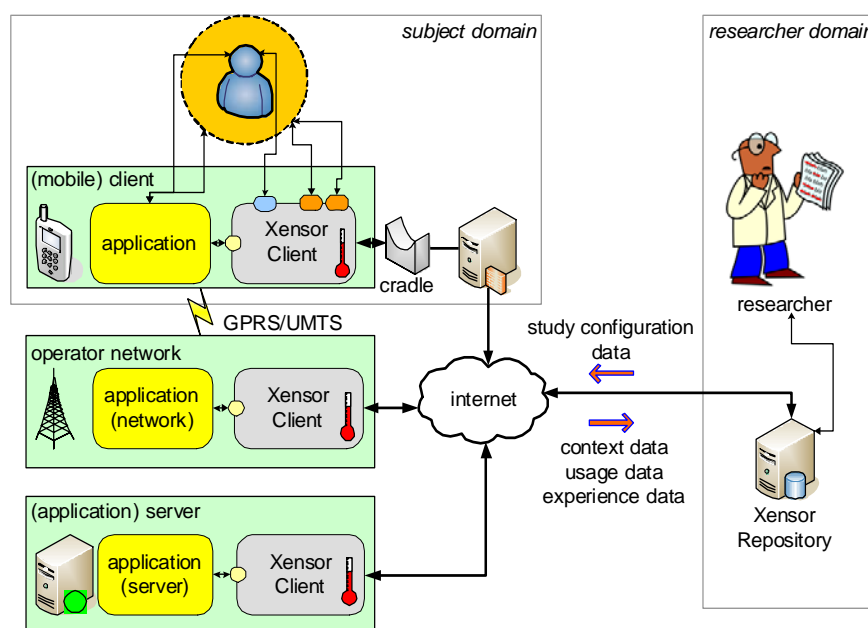


Figure 2. Xensor System high-level architecture (Mulder et al., 2005).

Xensor Repository

A Xensor Repository is located on a central computer such as a server and provides support for study configuration, for deploying the configuration to Xensor Clients, for persistent and secure storage of data collected by Xensor Clients and for providing (basic) export functionality to analysis tools. Our current implementation of the Xensor Repository is based on the SQL Server 2005 database server. At the time of writing this paper, design and implementation of the web-based configuration interface was still in progress; changing configuration data is only supported by a database management interface, which requires more technical skill than we desire.

The Xensor relational database can support multiple *Studies*, each having several *StudySubjects*, with an anonymous study-specific name. If so desired, non-anonymous data can be stored about *Persons*. Each *Person* can be a *StudySubject* in many *Studies*. Each *Study* has one or more *ActiveXensorModules*, which is a subset of all *XensorModules*. Each *ActiveXensorModule* in a *Study* can have one or more *ActiveXensorModuleVariables*, which is a subset of all *XensorModuleVariables* of that *XensorModule*. The *XensorData* data collected is stored as a time stamped string value along with the name of the *XensorModule*, *XensorModuleVariable*, *Study* and *StudySubject* the value belongs to.

Xensor Clients

A *Xensor Client* allow (third-party) *Xensor Modules* to collect data and interact with a *Xensor Engine*, which takes care of local storage and uploading data to a *Xensor Repository* via appropriate media at appropriate moments (see Figure 3). In theory, a *Xensor Client* can not only be located on a mobile device, but also in the network or on an application server.

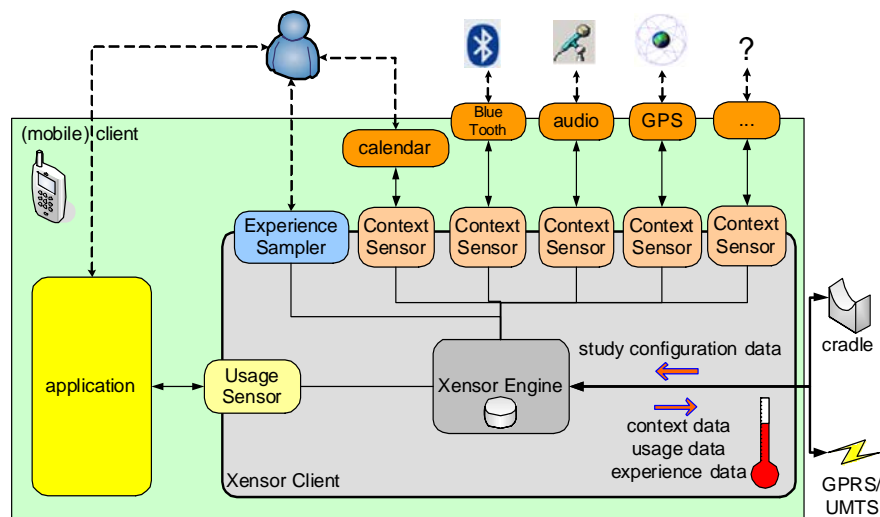


Figure 3. Xensor Client detailed architecture (Mulder et al., 2005).

Our current Xensor Client implementations all run on small, portable yet powerful mobile devices with the Windows Mobile 5.0 operating system. We used the .NET Compact Framework 2.0 as programming platform. To store the data collected by Xensor Modules until it is uploaded at opportune moments to a Xensor Repository, we initially used SQL Server 2005 Mobile, with exactly the same database schema as the Xensor Repository. Deployment can be done over a wired or wireless secure network connection by synchronizing only the study- and subject specific configuration data to the Xensor Client, or by inserting a storage card that includes a pre-populated database with this information. Although replication back and forth worked fine with the SQL Mobile database stored on device memory, we ran into stability problems with running SQL Mobile on storage cards, which could not be resolved before the first study with SocioXensor, in which we switched to storing configuration data and data collected in an encrypted text file on the storage card of the mobile device. Our current clients do support suspend and resume, but not erasing recently collected data.

In Table I, we list various Xensor Modules that have been built as part of various implementations of the Xensor System.

Table I. Xensor Modules that were implemented

Xensor Module	Data logged	SocioXensor	SeniorXensor ¹	RouteXensor ²
Context Sensors	Human behaviour and context			
BatteryXensor	Percentage of battery left	X	X	X
NalXensor	ID's of connected GSM cells (continuously), names and addresses of Wi-Fi access points in range (one out of every ten minutes and when device is on), and names and addresses of visible Bluetooth nodes (once every 5 minutes). Based on the CoSphere Network Abstraction Layer (http://cosphere.telin.nl).	X		
BluetoothXensor	Names and addresses of visible Bluetooth nodes		X	
PhoneXensor	Time, duration, and phone number of phone calls		X	
GpsXensor	Location, speed, direction and quality of GPS signals			X
AliveHeart3DXcelXensor	ECG heart signal & 3D-accelerometry from the Alive Heart Monitor ³			X
Xamplers & Diarycorders	Self-report data			
BasicXampler	Text-based questions and closed answers with a uniform random inter-sample distribution and notification conform current profile.	X		
AudioDiarycorder	Brief audio recordings when subject presses key		X	
SkateXampler	Text-based and audio-based questions and answers.			X
Usage Sensors	Application usage			
SocketLogger	Time-stamped strings containing variable-value pairs written by an application to a particular local socket		X	

Implementing a Xensor Module can be very simple, using the *XensorModule* base class which provides scheduling, time stamping and reporting facilities. In many cases, programmers only need to override the *sampleNow()* method with code that collects data from sensors and makes it available for reporting. With proper hardware libraries, the amount of code that needs to be written can be very small, as illustrated in the C# source code for BluetoothXensor below, based on In The Hand's 32feet.NET library (<http://32feet.net/>):

```
using InTheHand.Net.Sockets;
public class BluetoothXensor: XensorModule
{
    BluetoothClient btc = new BluetoothClient();
    public override void sampleNow()
    {
        BluetoothDeviceInfo[] bti = btc.DiscoverDevices();
        for (int i = 0; i < bti.Length; i++)
        {
            XensorDataArray data = new XensorDataArray(xensorModuleName, newDataCollectorId());
            data.Add("address", bti[i].DeviceAddress.ToString());
            data.Add("name", bti[i].DeviceName);
            _dataCollector.reportData(data.ToArray());
        }
    }
}
```

On the other hand, implementing a Xensor Modules may also prove to be hard. A good example is the NalXensor, which needed to collect Wi-Fi data for 1 minute every 10 minutes, without bothering the user. When a Windows Mobile 5.0 PocketPC device is turned "off" by the user or by a timer, the device goes to sleep. In this situation it proved virtually impossible to start collecting Wi-Fi data, without turning on the screen, which could possibly confuse the subject. We managed to do so, nevertheless. At least, so we thought.

¹ SeniorXensor is built for an in-situ study into the mobile device and application usage of patients with light dementia.

² RouteXensor is built for a study into inline skating experience and automatic route (segment) quality detection.

³ <http://www.alivetec.com/products.htm>

SocioXensor: first experiences

SocioXensor is the first Xensor System implemented. It was built for and used in a study into:

- the extent to which particular context information (logged with BatteryXensor and NalXensor and sampled with BasicXampler) is predictive for a person's availability for a phone call from a particular social relation and to what extent does that person want to share that context info with those relations (sampled with BasicXampler).
- network contexts people encounter in daily life (logged with NalXensor), which could be used to explore and validate algorithms that deal more efficiently with network resources.

In this paper, we only report about our experiences with SocioXensor as an in-situ research tool, not about the results of the studies themselves, which can be found in (Ter Hofte, 2007) and <http://cosphere.telin.nl>, respectively.

- *Most subjects used their existing, primary mobile phone:* We recruited 12 subjects with an e-mail to knowledge workers that worked for the same company. Nine subjects had been using the device for at least several months as their primary mobile phone; one started using it just a few weeks before the study, one used it as a PDA and used a separate mobile phone, and one other subject used it as a new PDA device, but kept using the old mobile phone, despite us transferring all contacts from the old phone to the new phone.
- *Informed consent was not trivial:* We carefully explained and described what data would be collected and obtained signed informed consent forms. Some data that we collected concerned not only the subject, but also others around that subject (e.g. Bluetooth scans). We do not know whether and how we should have and could have obtained informed consent from such secondary-order subjects. Some of the data we collected, such as GSM Cell-ID traces can be traced back to a particular person with some (publicly available) knowledge about GSM Cell-IDs. So we decided to replace GSM tower numbers in publications. Part of the informed consent procedure we set up required us to challenge a particular trusted independent researcher (working on context-aware authentication) to try and identify the person(s) from the data we were about to publish.
- *Early dropouts:* Excessive crashes of the device caused one subject to terminate the participation in the study after a few days. Others did not seem to experience this problem. The BasicXampler was only supposed to be active during the first week of the study, whereas other data would be logged throughout all weeks. Towards the end of the study, data analysis revealed a subtle bug in the way BasicXampler logged data, which nullified the added value of that data with respect to earlier studies. Hence, we asked the subjects to extend their participation with another week, this time including experience sampling again. One subject chose not to, referring to the device crashing quite often.
- *Device crashes:* Of 785 experience samples that were scheduled, 35 were logged as missing, possibly due to the device being completely off or crashed at the time the sample should have been taken. Battery samples, scheduled once per minute, reveal that during 6% of time, no sampling could have taken place. We also logged 55 device restarts, less than one per subject per day. The post-study survey and interviews revealed that most of these were caused by spurious crashes of the device, not by deliberately turning off the device. A particularly bad example of this was a subject that occasionally noticed the device becoming very hot, depleting remaining battery very quickly and then crash. We suspect that device power management in combination with Wi-Fi drivers is the culprit.
- *Low explicit non-response, high implicit non-response:* Of all samples 3% (25/750) were explicitly reacted to with the "Not now" option (12), or were explicitly suppressed with the suspend/resume feature (13). Out of the 10 subjects that participated during the last week, 7 never used the suspend feature. Of all samples, 23% (173/750) were not reacted to within 2 minutes at all (166), or were reacted to, but were not completely answered (7).

- *Variable response rate:* 74% (552/50) of samples were completely answered. The response rate per subject varied between 56% and 97%. We suspect that different habits carrying device and device usage frequency to be the cause, but did not do a formal evaluation.
- *Low time effort per subject, though it may have felt differently:* Each experience sample consisted of 6-7 questions, depending on the answers given. On average, all subjects spent 30s per sample, varying between 12s-237s (see Figure 4). The averages per subject varied between 22s-55s. Subjects spent 28 minutes on average answering all experience samples during the last week (with averages per subject varying between 14-47 minutes).

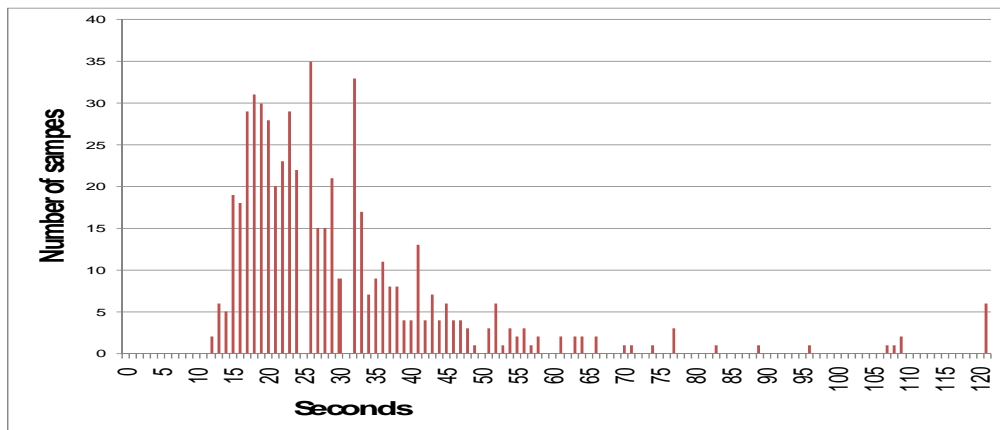


Figure 4. Histogram of time spent answering one sample

Related work

One of the earliest inspirations for our work on SocioXensor was the Sociometer (Choudhury et al., 2002), a custom-made neck/shoulder-worn device that provided new ways of studying dynamics in social networks, by recording conversations between people with a microphone and measuring proximity between people with infrared transmitters and receivers. More recently, research tools have started to emerge that support in-situ evaluation using the hardware (sensors) and software functionalities of contemporary mobile devices, such as PDAs and smartphones. The Context Aware Experience Sampling (CAES) research tool (Intille et al, 2003) was one of the first that combined experience sampling with context logging, but it monopolizes the device, thus preventing “normal” usage of the device during a study. ContextLogger, a research tool that was part of the ContextPhone platform (Raento et al., 2005), does not monopolize the device and can log both context data and application usage data. However, ContextLogger did not provide any support for experience sampling. Similar to SocioXensor, MyExperience (Froehlich et al, 2007) is one of the first data collection tools for in-situ research that combine experience sampling, context logging and application usage logging. Whereas SocioXensor’s architecture is focused around Xensor modules, MyExperience’s architecture is based on sensors, triggers, and actuators, which can be composed very flexibly, but researchers have to learn an XML scripting language to do so.

Conclusion

Using smartphones as a data collection platform provides many cool opportunities for social scientists to study phenomena at places, times, durations and scales that have not been feasible before. While designing, implementing and using the SocioXensor data collection platform for smartphones, we needed to work hard to solve a surprising amount of technical hot issues, some even literally.

When the Xensor System stays cool, it will be released open source and free of charge.

Acknowledgments

We thank Raymond Otte, Arjan Peddemors, Peter Ebben and Ingrid Mulder and for their help and support. Also, we thank all subjects for their participation in the study and the e-Social Science 2007 reviewers for their comments and suggestions. The projects FRUX and AWARENESS in the Dutch Research Programme Freeband Communication supported the work on SocioXensor research under contract BSIK 03025. The EU project COGKNOW supported the work on SeniorXensor and the work on RouteXensor was supported by a grant under the Microsoft Research e-Science initiative.

References

- Consolvo, S., Harrison, B., Smith, I., Chen, M.Y., Everitt, K., Froehlich, J., & Landay, J.A. (2007): 'Conducting In Situ Evaluations for and With Ubiquitous Computing Technologies'. *IJHCI*, 2007, vol. 22, no. 1-2, pp. 103-118.
- Choudhury, T., & Pentland, A. (2002): The Sociometer: 'A wearable Device for Understanding Human Networks', in CSCW '02 Workshop: Ad hoc Communications and Collaboration in Ubiquitous Computing Environments, <http://www.media.mit.edu/~tanzeem/TR-554.pdf>.
- Csikszentmihalyi, M. & Larson, R. (1987): 'Validity and reliability of the experience-sampling method'. *Journal of Nervous and Mental Disease*, Vol. 175, pp. 526-536.
- Froehlich, J., Chen, M., Consolvo, S., Harrison, B., Landay, J. (2007) 'MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones', in *Proceedings of MobiSys 2007*, San Juan, Puerto Rico, pp. 57-70.
- Hofte, G.H. ter & Ebben, P.W.G. (2004): *Experience Sampling: Method, Tool, Pilot and Experiences* (TI/RS/2004/072), Enschede: Telematica Instituut, <https://doc.telin.nl/dscgi/ds.py/ViewProps/File-45237>.
- Hofte, G.H. ter & Ebben, P.W.G. (2005): *Xensor: Description, requirements and architecture* (TI/RS/2005/063), Enschede: Telematica Instituut, <https://doc.telin.nl/dscgi/ds.py/ViewProps/File-52015>.
- Hofte, G.H. ter & Otte, R.A.A., Peddemors, A., Mulder, I. (2006): 'What's Your Lab Doing in My Pocket? Supporting Mobile Field Studies with SocioXensor', in Conference Supplement of CSCW 2006, ACM: New York, pp. 109-110.
- Hofte, G.H. ter (in press): 'Xensible interruptions from your mobile phone', in *Proceedings of MobileHCI 2007*, September 9-12, 2007, Singapore.
- Intille, S.S., Rondoni, J., Kukla, C., Ancona, I., Bao, L. (2003): 'A Context-Aware Experience Sampling Tool', in *Proc. of CHI 2003*, ACM, New York, pp. 972-973.
- Mulder, I., ter Hofte, G.H., Kort, J. (2005): 'SocioXensor: Measuring user behaviour and user eXperience in ConteXt with mobile devices', in L.P.J.J. Noldus, F. Grieco, L.W.S. Loijens and P.H. Zimmerman (eds.): *Proceedings of Measuring Behavior 2005, the 5th International Conference on Methods and Techniques in Behavioral Research*, August 30 – September 2, 2005, Wageningen, the Netherlands, pp. 355-358.
- Raento, M., Oulasvirta, A., Petit, R., Toivonen, H. (2005): 'ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications', *IEEE Pervasive Computing*, 4, 51-59.