universität
wien

DISSERTATION

Titel der Dissertation

"A Holistic Multi-Purpose Life Logging Framework"

Verfasser

Mag.rer.soc.oec. Reza Rawassizadeh

angestrebter akademischer Grad

Doktor der Technischen Wissenschaften (Dr. techn.)

Wien, 2012

Studienkennzahl lt. Studienblatt:	A 786 880
Dissertationsgebiet lt. Studienblatt:	Informatik
Betreuer:	O.Univ.Prof. Dipl.-Ing. Dr.techn. A Min Tjoa

Acknowledgement

It is hard to mention all who helped me reaching this milestone of my life. On one hand I am grateful to God and my family, especially my parents and my spiritual leader who have all given me ability, motivation and support. Without their support during my Ph.D. I could not have survived.

On the other hand I am very thankful to Prof. A Min Tjoa who is my Ph.D. advisor. His intelligence and scientific vision, which are the result of his great experience and work in many different disciplines, was of very great help to me. For any new subject he came up with a good ideas, for instance the surprise detection idea. I should also be thankful to Prof. Wolfgang Klas who is my second advisor and I appreciate his precision.

Moreover I am thankful to Prof. Abigail Sellen and Dr. Katarzyna Wac who have read my thesis and gave me helpful feedback on my dissertation. Dr. Martin Tomitsch is another important person who helped in my research with his great personality and patience.

Furthermore I appreciate my colleagues Dr. Amin Anjomshoaa, Elaheh Momeni and Jana Machajdik for their feedback and assistance. Each of them helped me at different period of time and I have benefited from their knowledge.

I need to be thankful for the IFS staff for their brilliant support and the students who helped me in developing my prototypes including: Victor Andrei Gugonatu, Soheil Khosravipour, Michael Kuehberger and Christian Feichtinger. At the end I am thankful for all institutions who supported me in doing my research which are: Berlitz (especially Barry Kearnan), ÖFG and research services and the international relation department of the University of Vienna.

I dedicate this work to all the talented people around the world, who are suffering from lack of supports, which prevents them from studying and increasing their knowledge. I am happy that I have had this chance, but there are many more talented and hard working people around the world who would like to follow this direction and are not able to.

Abstract

The paradigm of life-logging promises a complimentary assistance to the human memory by proposing an electronic memory. Life-logs are tools or systems, which automatically record users' life events in digital format. In a technical sense, they are pervasive tools or systems which continuously sense and capture contextual information from the user's environment. A dataset will be created from the collected information and some records of this dataset are worth preserving in the long-term and enable others, in future generations, to access them. Additionally, some parts are worth sharing with society e.g. through social networks. Sharing this information with society benefits both users and society in many ways, such as augmenting users' social interaction, group behavior studies, etc. However, in terms of individual privacy, life-log information is very sensitive and during the design of such a system privacy and security should be taken into account.

Currently life-logs are designed for specific purposes such as memory augmentation, but they are not flexible enough to accept new sensors. This means that they have been configured to work only with a predefined set of sensors. Sensors are the core component of life-logs and increasing the number of sensors causes more data to be available for acquisition. Moreover a composition of multiple sensor data provides better qualitative and quantitative information about users' status and their environment (context). On the other hand, sensor openness benefits both users and communities by providing appropriate capabilities for multidisciplinary studies. For instance, users can configure sensors to monitor their health status for a specific period, after which they can change the system to use it for memory augmentation.

In this dissertation I propose a life-log framework which is open to extension and configuration of its sensors. *Openness* and *extendibility*, which makes the framework holistic and multi-purpose, is supported by a sensor classification and a flexible model for storing life-log information. The framework enables users to share their life-log information and supports required features for life logging. These features include *digital forgetting*, facilitating *information retrieval* (through annotation), *long-term digital preservation*, *security* and *privacy*.

Zusammenfassung

Die Paradigm des Life-Loggings verspricht durch den Vorschlag eines elektronisches Gedächtnisses dem menschlichem Gedächtnis eine komplementäre Assistenz. Life-Logs sind Werkzeuge oder Systeme, die automatisch Ereignisse des Lebens des Benutzers aufnehmen. Im technischem Sinne sind es Systeme, die den Alltag durchdringen und kontinuierlich konzeptuelle Informationen aus der Umgebung des Benutzers erfassen. Teile eines so gesammelten Datensatzes könnten aufbewahrt und für die nächsten Generationen zugänglich gemacht werden. Einige Teile sind es wert zusätzlich auch noch mit der Gesellschaft geteilt zu werden, z.B. in sozialen Netzwerken. Vom Teilen solcher Informationen profitiert sowohl der Benutzer als auch die Gesellschaft, beispielsweise durch die Verbesserung der sozialen Interaktion des Users, das ermöglichen neuer Gruppenverhaltensstudien usw. Anderseits, im Sinne der individuellen Privatsphäre, sind Life-log Informationen sehr sensibel und entsprechender Datenschutz sollte schon beim Design solcher Systeme in Betracht gezogen werden.

Momentan sind Life-Logs hauptsächlich für den spezifischen Gebrauch als Gedächtnisstützen vorgesehen. Sie sind konfiguriert um nur mit einem vordefinierten Sensorset zu arbeiten. Das bedeutet sie sind nicht flexibel genug um neue Sensoren zu akzeptieren. Sensoren sind Kernkomponenten von Life-Logs und mit steigender Sensoranzahl wächst auch die Menge der Daten die für die Erfassung verfügbar sind. Zusätzlich bietet die Anordnung von mehreren Sensordaten bessere qualitative und quantitative Informationen über den Status und die Umgebung (Kontext) des Benutzers. Offenheit für Sensoren wirkt sich also sowohl für den User als auch für die Gemeinschaft positiv aus, indem es Potential für multidisziplinäre Studien bietet. Zum Beispiel können Benutzer Sensoren konfigurieren um ihren Gesundheitszustand in einem gewissen Zeitraum zu überwachen und das System danach ändern um es wieder als Gedächtnisstütze zu verwenden.

In dieser Dissertation stelle ich ein Life-Log Framework vor, das offen für die Erweiterung und Konfiguration von Sensoren ist. Die Offenheit und Erweiterbarkeit des Frameworks wird durch eine Sensorklassifizierung und ein flexibles Model für die Speicherung der Life-Log Informationen unterstützt. Das Framework erm- öglicht es den Benützern ihre Life-logs mit anderen zu teilen und unterstützt die notwendi-

gen Merkmale vom Life-Logging. Diese beinhalten Informationssuche (durch Annotation), langfristige digitale Erhaltung, digitales Vergessen, Sicherheit und Datenschutz.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Hypothesis and Questions

We need to adapt our resources such as time, health and energy to our new lifestyles. New technologies such as life logging could resolve these challenges, because the paradigm of life logging promises a **complimentary digital assistance to human memory**. This can be used to augment users' memory and assist us in health monitoring, transportation optimization, energy usage monitoring and time management. However, available life log tools are designed for separate and unique purposes and are thus not flexible enough to be configured for different use cases. Moreover they do not take into account some technical (e.g. annotation) and conceptual (e.g. forgetting) requirements of digital memory. Here I propose a lightweight, holistic multipurpose life logging framework which is flexible enough to enable users to configure it and use it for different purposes such as those previously described. In addition this framework tries to simulate some real memory characteristics which include sharing and forgetting in digital memory.
This dissertation tries to answer following research questions:

- What are the technical challenges of a flexible life logging system that can be used for different purposes ?

- In order to create a digital memory that works in synergy with human memory, what features of human memory should the digital memory support ?

- Since life-logs host very sensitive personal information, how can they be made secure while respecting users' privacy ?

1.2 Contributions

This dissertation proposes a generic life-log framework which is open, extendable, lightweight and considers the requirements of a life-log. Additionally, it enables users to share their life-log information with society, respecting their privacy, and enabling them to perform "forgetting" in relation to their shared and private information. It has been assumed that the process of life logging includes collecting, aggregating and preserving life-log information, therefore I do not delve deeply into reflecting on life-log information. However, in order to evaluate the usages of implemented prototypes and tools, I propose some *visualizations* and two *knowledge mining* methods (surprise detection and life routine detection). Following are the major contributions of the dissertation:

Software Architecture

This contribution focuses on software architecture that can be used as a reference model for implementing a flexible and extendable life-log framework. Life logging here has been interpreted as the sensing and recording of users' contextual information. It should sense continuously and should be ubiquitous.
Sensors are core components of a life-log system, and they are used to sense contextual information. Since there are many sensors available that can be used for life logging, and in the future more new sensors could be invented for this purpose, the framework is designed to be sensor independent. In other words, the framework allows the addition of new sensors or the configuring of existing sensors. *Openness* to accept new sensors is one of the major novelties and advantages of the framework architecture, which makes the framework *flexible* and *extendable* in terms of sensor

configuration.

Another challenge of life-log systems can be observed within the area of information retrieval. In order to facilitate the information retrieval process I will use *annotation*. The annotation enriches raw sensors data objects by assigning semantics to them.

The life-log dataset is worth keeping during the users' life, and it can be inherited too. Therefore *digital preservation* of the life-log dataset is important. However, life-logs rely heavily on pervasive devices such as mobile phones. Pervasive devices are prone to loss and are unreliable for keeping data in the long-term. A lightweight digital preservation approach regarding pervasive devices will be described.

Furthermore, mobile life-log applications should run in the background of a pervasive device 24/7, therefore resource efficiency of the pervasive life-logs will be considered during the prototype implementation.

Sharing and Forgetting

A major requirement of a human memory is sharing experiences and knowledge. Some information from the life-log dataset (e-memory) are worth being shared with society. On one hand sharing life-log information with society, benefits both users and community in many ways. On the other hand, life-log information are highly sensitive in terms of users privacy.

To my knowledge, there are few research about *sharing* life-log information with respect to the life-log requirements. A major novelty of this framework is sharing life-log information within a community.

Any shared information in human society should have the chance to get forgotten. Forgetting is a major requirement of human memory. Life-logs can be interpreted as electronic memories for their owners and assist them in augmenting their biological memory. This framework supports users in performing *digital forgetting* (for both private and shared information).

Besides other theoretical requirements of sharing life-log with society will be examined in detail. These requirements include risk and benefit assessment of sharing life-log information and ethical suggestions for service provider and service consumer.

Privacy, Security and Ethics

Since life logging tries to replicate the biological memory in digital format, it should record all individuals' life events. This level of recording could be a serious treat to users' privacy. Thereby, privacy should be considered during the development of a life-log system. In order to consider users' privacy I describe: methods for securing the process of life logging, a lightweight pseudonymization approach which enable users to the control their information, ethics for sharing life-log information, and a light weight datamodel which enforces users to restrict sharing their life-log information by expiration date-time.

1.3 Background and History

Self monitoring and memory augmentation are two main advantages of life logging and both have a long-standing history in human life. Many religious and spiritual statements emphasize that humans should observing their behaviors and review history. Although self monitoring is not new and does not arise after the digital age, in the digital age it has been grown rapidly. Because of the digitization, cheap storage, easy retrieval and global reach, remembering become a norm [148] and humans remember their behavior easily by employing digital devices.

In 1913 Bevans [37, 122] wrote an article which describes how people spend their money and it leads to the question of how people spend their spare time. It can be called the first scientific effort toward self monitoring. After World War II these types of research grows rapidly. In 1945 Vaneevar Bush proposed an imaginary device called Memex, in his famous article "As we may think" [46]. Memex senses and records all individuals' life time data such as communications, books, microfilms, etc. He described a machine can act as a human memory prothesis and he anticipated that Memex will be used to optimize humans' life:

Presumably man's spirit should be elevated if he can better review his shady past and analyze more completely and objectively his present problems.

Bush's vision in this article has inspired a variety of other research endeavors from information retrieval to distributed hypertext systems. Memex has been interpreted

as the first personalized knowledge base [61] or in a more technical sense it is the first life-log system [33].
Skinner [207] described that individuals learn behavior by reinforcement and punishment. The process of getting reinforcement, and producing a new behavior by reinforcement, is called "Operant Conditioning". Operant conditioning forms an association between a behavior and the consequences of this behavior.
In 1962 Engelbart [76] introduced a "Conceptual framework for Augmenting human's intellect" which cites some Memex ideas, such as links between data objects.
In 2004 Bell and Gemmel [1] launched a workshop series on Capture, Archival and Retrieval of Personal Experience (CARPE). This annual workshop held for three years and after that life-log research grows slowly, but continuously.

1.4 Human Memory and Digital Memory

Memory is the human ability to remember [2] and humans by their very nature try to remember, preserve knowledge and hold their memories. Numerous devices and tools have been invented to assist humans in augmenting or preserving their memory. These devices vary from paper to SD memory cards of Mobile phones and PDAs.
In order to describe electronic memories and their role in augmenting our lives, first I describe the structure of human memory and the process of remembering. Then I describe how technology have been employed to augment human memory.
There are three types of memory *short-term memory*, *longterm memory* and *sensory memory* [27]. Sensory memory acts as a buffer for stimulus received through the human senses. Information in the sensory memory disappears in a second. Short-term memory retains a piece of information for less than a minute and is able to retrieve it during this short time frame. Information goes from the sensory memory to the short-term memory based on the attention paid to the information in the sensory memory. Long-term memory stores information over a long time. Information can remain there for a very long time, even for the rest of human life. Long-term memory itself is composed of two parts: *Explicit (declarative) memory* and *Implicit*

[1] http://replay.waybackmachine.org/20090517023401/http://www.sigmm.org/Members/jgemmell/CARPE
[2] Remember is similar to recall but it uses when there is an intention not to forget something.

(non-declarative) memory. Explicit memory contains information that we are aware of remembering them (conscious recall). In contrast to implicit memory, which is non-obligatory or facultative, here information will not be recalled by a conscious recall.

Recent technological advances enable us to easily sense and record where we are, who we met, what we did, what we say, our vital signs, calls, notes, etc. These information objects can be indexed, filed and cross-referenced by time and location via life logging. In particular, life-logs enable us to collect and record our life experiences and events. The result of life logging is a dataset, which contains diverse array of information about the user activities. This dataset is the digital memory (e-memory)[3] of the user. The e-memory augments users memory plus it can assist users in self monitoring, time management and other benefits, which will be described in the next sections.

Although the process life logging is interpreted as creating an e-memory it does not support all real memory functions. In addition to that, psychological properties of a human memory have been ignored by the e-memory designers [202]. Life logs could benefit more by supporting human memory functions and properties such as forgetting [195]. Human memory properties and functionalities are not yet identified completely. Technically, just some of those such as forgetting can be implemented. Moreover, we cannot claim that e-memories and life-logs are available and operational. This is due to the numerous issues such as data fragmentation, different data format for different devices, battery life of the sensing devices, maintenance of passwords and personal profiles, etc. Nevertheless it has been proved that life logging assists users in memory augmentation and reminiscence [201]. Another principle of human memory [38] is that contextual cues assist individuals in recall and reminiscing and life-logs collect contextual cues. Therefore they facilitate recall and reminiscence.

In addition to the reminiscence life logging can benefit users in self monitoring and thus self-insight. We gain experiences from the consequences of our behaviors, self monitoring enable us to review our past behaviors and thus not repeating a wrong behavior. Additionally, sharing life experiences and outcome of our behavior, can re-

[3] Sometimes the term electronic memory (e-memory) is being used instead of digital memory, but to my knowledge there is no difference in between.

move the cost of experiencing the experienced mistakes for others. Not forgetting an experience is a major challenges in our life, which leads us to make a wrong decision or perform a wrong behavior. Life-log systems provide a form of self quantification, which enables us quantifying our behaviors. Through self quantification we will gain better vision on our behavior outcomes, we can manage our time, monitor our health and aggregate information from different sources to gain knowledge about ourselves. For instance, consider a scenario in which users will get informed about how many hours did they spent watching TV. Time is not the only resource that an individuals can measure, physical and even psychological health are the other resources which can be monitored by using life-logs. Observing personal life resources frequently is a way which assists individuals to prevent wasting these resources. For example, a user can measure her heart-beat rates when she is in a specific location and have social interaction with a specific individuals there. These data objects can be aggregated and can be used to infer who or appearance in which locations makes the user nervous. In another scenario assume a user which she is not satisfied with her presence in the library, because she should study more. She can use a location sensor of a life-log tool to monitor her study hours by her presence in the library. Based on her presence in the library she can specify a goal and the life-log system reports about her achievement toward the goal.

These examples show that life-logs have high potential to be used as a self monitoring tool and tracking users' behaviors. Analog behavior monitoring approaches are resource intensive and cost time, but life-logs could remove this burden from researchers and users. Besides, issuing queries on life-log dataset could produce stimuli to motivate users evaluating their behavior.

People who are studying e-memories like Bell [33] and Mayer-Schoenberger [148] all agree that we are moving toward ultimate and immortal memory prothesis in the digital age. This is because of the amount of digital recording in our daily life, easy information retrieval methods, cheap storage and wide access to information.

1.5 Life-log Usages

In order to realize the importance of using life-logs and their effects on human life, I list usages of life-logs. Furthermore, detailed risk-benefit assessment of sharing life-logs with society will be described in the next chapters.

Bell [33] as a pioneer in life logging anticipated that e-memories, which is the result of life logging, will have a revolutionary impact on our lives:

e-Memories will change the way we work and learn. It will unleash our creativity and improve our health, it will change our intimate relationships with loved ones both living an dead. It will, I believe, change what it means to be human.

Sellen and Whittaker [202] identified life-log benefits and they introduced five Rs. The five Rs are activities which are recollecting, reminiscing, reflecting, retrieving and remembering. Recollecting enable users to re-live an specific life events and thinking back in detail (refer to episodic memory). Reminiscing assists users in remembering past experiences. Retrieving assists users in retrieving digital information that they have encountered over the long period of time. Reflecting supports more abstract representation of personal information. Remembering assists users in activities which concern remembering such as taking medication (refer to prospective memory).

In order to be more use-case specific, usage of life-logs has been categorized into personal and social fields. In this section, only usage in the personal domain will be described. These usages have been identified based on analyzing existing systems and theories. It is possible that in the future more usages will be discovered in this domain, because few experiments have been done with the real operational system. It is notable that recording human life time data can be interpreted as life logging. This means that research about personal monitoring or continues sensing can be interpreted as life-logs. Hence related works in these domains assist me to identify usages of life-logs. Those works will be cited here, but more explanation about them is available in the “Related Works” chapter.

The following lists potential advantages of using life-log in personal scope and only for end users. However the border between the personal and social usage will get blurred sometimes.

- *Memory Augmentation*: As has been described before life-logs is designed to record individuals life events. Our life experiences is being kept in the episodic memory [30]. Time and location of the occurred event are two important factors that assist users in event retrieval from episodic memory (when and where) [221]. All pervasive life-logs supports date-time and some supports location (e.g. GPS receiver). In simple terms all life events will be saved with a timestamp and location (if it is possible to sense the location). Life-logs are well known as being used as memory aids and it has been proved that life-logs augment the episodic memory [201] of users. Moreover life-logs can provide rich information about context and contextual information assists the recall process in memory [38]. There are several research projects which have employed life-logs for memory augmentation such as: [201, 21, 228]. Gemmell one of the “MylifeBits” [91] designers, stated followings about the their life-log project MyLifeBits:
  ”Having a surrogate memory create a feeling, uplifting, and secure feeling similar to having an assistant with perfect memory”.

- *User Modeling, Personalization and Life Patterns Analysis*: Individuals past actions are the best predicator of their future decisions [174]. Most user modeling approaches are based on users past activities e.g. Doppelgänger user modeling system [164]. Past users activities constitutes the foundation of user modeling and software personalization. Clrakson [56], an early entrepreneur in this domain, extracted users life patterns by using a video recorder apparatus, which was always carried by him. User tracking, which is based on collecting users’ activities, is the base of online advertising. Therefore life-logs highly benefit users in those systems such as recommendation systems [111, 222].

- *Desktop Metaphors*: User interface (UI) and information appearance in desktops can be customized by using desktop activities and contextual cues [90, 68]. Users activities in the desktop build contextual cues. Desktop customization facilitates information discovery such as optimizing search and browse for specific information objects. In simple terms, information access will be optimized by analyzing past activities of users, because the system can learn users’ preferences. LifeStream [78], iMemex [65], SemanticLife [19] are examples of these desktop metaphors.

- *Health Monitoring and Medical Studies* Recording individuals' biological information continuously and in the long-term, can be interpreted as life logging. For instance FitBit [3] or Q-Sensor [5] are tools which could be interpreted as life-logs for users' health.

  Sensing and recording biological information or individuals' vital signs, were available before the digital age. Holmes and Rahe gave a credit to Adolph Meyer for inventing the "life chart" [134, 122], a device for organizing medical data as dynamic biography showing changes of habitant, births, deaths, etc. The device which is used for medical data acquisition called Holter devices or Event Monitors [104].

  In the late 1970 Lazarus [122] and his associates discovered that identifying stress markers was obtained via observing content of the individuals life. Another research shows that automatically-created photographs boost Alzheimer patients short-term memory more effectively than other methods such as keeping meticulous diaries [36]. These facilities can be acquired by using life-logs. Additionally recording vital signs benefits users in gaining self-insight about their health status, enriching their health record and assisting them in interactively learn how to affect their biological parameters consciously (biofeedback exercises). It has been proved that biofeedback exercises improve users' health [131, 132]. "Every sign of life" [95] is one of the earliest scientific approaches which tries to monitor users' health by continuous recording of their vital signs such as heart beat, body temperature and galvanic skin responses.

- *Sustainability and Environmental Impact* Reducing greenhouse gas emissions and moving toward a low carbon economy is a major challenge of this century. Monitoring and recording users' indoor energy usages in a smart home assists users in analyzing their energy usage patterns and identifying bottlenecks of their energy utilization systems. Thus it persuades them to consider their energy use. Furthermore, mobile devices are well suited to sense and provide feedback about users' transportation mode and calculate their energy expenditure [89, 155]. This is due to the fact that recent advances in sensor network and computing capabilities of smart phones enable users to sense their environmental impact.

There are some others advantages that users will gain by life logging such as digital immortality [35], which are still far from reality. Therefore I did not list them here. How society or a community can benefit from using life-logs will be explained in the "Framework Features" chapter.

1.6 Thesis Organization

As has been described before the main contribution of this research is a multipurpose life-log framework that can handle identified requirements of a life-log system. In order to explain the framework, this dissertation has been divided into three section as follows:
The first section describes framework features include sharing life-log information with society and its requirement. This chapter will be started by performing a risk-benefit assessment on sharing life-logs with society. Next a conceptual model, which supports a fine-grained access model will be proposed. This model can be used for sharing life-log information while considering forgetting (as a major requirement of e-memory). This model describes the concept of data-entity as a unique data object in the spatio-temporal world. The collection of data entities builds the life-log dataset, which is the e-memory of the user. Afterward impact of openness will be examined in detail.
The first section describes the technical architecture of the framework and prototypes which are the implementation instances of architecture. Sensors are the core component of life-logs. Therefore, a sensor classification will be introduced, which covers any sensors in a process of life logging. The sensor classification assists life-log designers in designing an open and extendable life-log framework. The architecture includes the client(s) and the server. Besides, Life-log requirements such as long term digital preservation will be identified and they will be taken into account during the implementation of prototypes. The framework covers the information collection, annotation and aggregation stages, which means how raw data is collected, aggregated from sensors and how does it get augmented through annotation. The architecture

is component based and the technical specification of each component will be described in detail.
The third section identifies security and privacy related risks in this framework, then I describe counter methods to resolve them. A platform which handovers the control of user generated data to users and a method for evaluating a mobile application resource utilization, with their associated prototype implementation will be described. Moreover, ethics for sharing this information on both user level and service provider level will be proposed. Ethics will be suggested due to the fact that on one hand life-log information are highly privacy sensitive and on the other hand a real life-log tool with sharing capabilities has not been implemented yet.
At the end I finish the dissertation with a future work and conclusion. Analyzing the life-log dataset, information mining and knowledge extraction is out of the scope of this dissertation. However to evaluate the implemented prototypes, some reflection and knowledge extraction methods will be explained.
Figure 1.1 shows the building blocking of this dissertation.

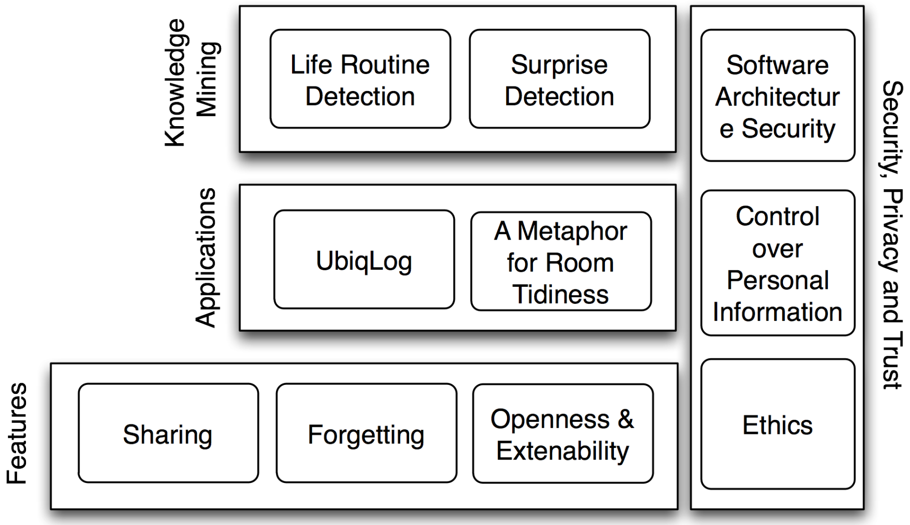


Figure 1.1: Building blocks of the thesis

Chapter 2

Related Works

There are scientific efforts focusing on recording personal experiences and life events, most of those efforts are single purposes such as health monitoring, and a few (e.g. MylifeBits [93]) are multipurpose.

What is listed here is not necessarily a life logging approach, but there are similarities to life-log systems. Those similarities include: continuous sensing and recording of contextual information, unobtrusive, automatic sensing, and archiving of personal information.

First well-known life-log approaches will be listed, then I list desktop metaphors and PIMs (Personal Information Management), which are used to assist users in reminiscence or information retrieval by continuous logging. In this context "information retrieval" means to browse a user's past activities, and this research can be interpreted as desktop activity based life-logs. Finally, I list context-aware approaches which are used for life logging or have similarities with ubiquitous life-logs.

MyLifeBits: MyLifeBits is the most famous research done in the field of life-logging. In 2001 Bell wrote the "A Personal Digital Store" article and introduced CyberAll, which is a personal ontology in contrast to the library. CyberAll [32] stored personal infomation and enabled users to do easy retrievals. It had been designed to store every individual's life time information, and thus eliminate the paper, which is necessary for storage and transmission. Bell has introduced golden formats (TXT,

HTML, JPG, PDF, RTF/DOC and TIF), which are appropriate for being kept long-term.
MyLifeBits follows the CyberAll metaphor which is a lifetime data store for human life time information. Bell uses his life [93] as an experimental corpus for MyLifeBits. This project provides rich support for desktop sensors, and their annotation tools. Date-time is the implicit link between collected data (as Bush[46] had suggested). Data objects will be explicitly linked by a typed link such as annotating a person in a photo [91]. To host the life-log information, a central Microsoft SQL server Database repository is being used [93]. MyLifeBits has been built based on four principles [92]: (I) collection and search must replace the hierarchy organization, (II) different visualization forms should be supported, (III) annotation is critical for the non-text media and it should be easy to perform, (IV) authoring should be done by transclusion.
MyLifeBits supports multimedia files preservation within annotation. It replaces the traditional folder structure, by collections that form a Directed Acyclic Graph (DAG) [93]. Moreover, standard forms of reporting [93] and different forms of data visualizations such as detail, thumbnail, timeline and clustered-time views [92] have been proposed. Furthermore, there is a story telling feature. The story telling component is a screen saver, which shows past pictures with audio via “Interactive Story By Query”(ISBQ).

Reality Mining: This research focuses on studying social behavior of a group of users by employing mobile phones for life logging. Researchers collect life-log information of a group of users and a dataset is made. This means that each user in this research did a life logging via his/her mobile phone and the accumulation of these life-logs creates a rich dataset. In simple terms, mobile phones have been used as sensors for social activities of users and measuring social interactions between individuals [72]. The dataset [69] contains social activities of about 100 users at MIT University, between 2004 - 2005. Different social studies could be done by a reality mining dataset such as epidemiology, group behavior learning, etc.
Since Bluetooth is an appropriate short range RF network, and most mobile phones support Bluetooth, the social proximity detection will be done by the Bluetooth

transmitter of the user's mobile phones. A MIDP (Mobile Information Device Profile) application has been developed, it is called "Blueware", which runs passively on the Nokia series S60 phones and records BTIDs encountered in a proximity log. Then logs will be sent to a server called "Serendipity server" [71, 72]. Bludar is a Bluetooth beacon which is coupled with a WiFi bridge and it scans for Bluetooth devices. It then transmits the BTID of the detected devices to the Serendipity server. Reality mining proposes a social matching component [71] too. Observing the social interactions by measuring person-to-person proximity and recording location changes could assist researchers in modeling human networks and group behavior. Those modeling could be used to predict individuals behavior and activities of social groups [167, 73].

2.1 Desktop Metaphors and PIMs

The following approaches are used to record users' activities on desktop. Recorded activities can be used to optimize information access, learning users' behavior, assisting users in browsing and searching their personal information, and assisting them in reminiscence.

LifeStream has been designed to manage and organize desktop information [78]. It uses a time ordered stream of documents to replace files and directories structures. Information is monitored, organized, located and summarized by stream filters and software agents. Moreover it provides search and browsing capabilities. Lifestream could be used as an electronic dairy of users desktop activities.

Haystack [17] was designed to personalize and augment the desktop search by recording the user's desktop activities. The Haystack paradigm indicates that "Personal Repository" reflects the knowledge of the owner, therefore personal knowledge resources such as the personal bookshelf a valuable resource to get information. In other words, information retrieval should be like getting a book from a personal

library, and other information retrieval approaches (at that time) are like getting a book from a public library.
Haystack follows a three steps process. The first step is gathering and recording all desktop information that users have encountered, including their metadata. The second step is gathering information about users and analyzing the information by including direct human input. The third step is modifying the retrieval process based on user interactions. Moreover, this research contains a specific data model which is generated to store different data types and links between them.

Stuff I've Seen (SIS) [68] is a precursor to MSN Desktop Search. It is a Windows based desktop optimization, which tries to facilitate information reuse. It contains five components. "Gatherer" that provides an interface for different content sources such as files and HTTP. "Filter" is responsible for decoding individual file formats for further processing. "Tokenizer" breaks the stream of characters into the 'words' and performs additional linguistic processing. "Indexer" builds a standard inverted index structure in order to support quick information retrieval. "Retriever" is the query layer and it is being used for accessing the stored information. Additionally, this approach uses meta-data annotations, landmarks and time line visualization for improving user interface and thus improving the information retrieval process.

SemanticLIFE [19] is a desktop based system, which is designed to store, manage and retrieve users' life-time data. It is a PIM that performs life logging for desktop information. This approach monitors and stores email messages, browsed web pages, phone calls, images and contacts in RDF format. Meta-data annotation and ontologies will be used to retrieve information from the SemanticLife dataset. As other systems it targets single users and social networking features are not available.

iMemex [63] tries to resolve the problem of managing or querying personal data objects, which have different data formats and are distributed among different locations, such as personal computers and network shared drives. It has been designed to be integrated into the existing operating systems by residing on the file system and exposing its functionality via a WebDAV interface [65].

Binary file formats can only be processed by a small number of the associated applications, and DBMS systems store files in a binary format. iMemex tackles this issue by introducing PDSMS (Personal Data Space Management), which uses Data Space [85] definition as a repository for the personal data. Furthermore it introduces a data model which is called iDM [64]. iDM separates the logical data structure from the underlying subsystem.
In simple terms, in addition to activity logging, iMemex proposes some features which are missing in file systems. These features include automatic back-ups, data exchange among applications, joining desktop information and improving query methods.

SLife [6] is an open source attempt that records users activities on their desktop, in order to assist them in better time management, by defining goals. Users can create categories, such as 'Reading news' or 'Doing research', and designate applications and documents as part of these categories.

Wakoopa [9] tracks users activities by recording what websites are visited, which software applications are used and what ads are seen. Users can share their acitivities with other users. The main purpose of this approach is to enable entrepreneurs to analyze and study user behavior on their desktops.

Zeitgeist [11] is a service which logs users desktop activities and events. It is composed out of four main components: Engine, Extensions, API and Data providers. It establishes relationships between logged items, based on similarity and usage patterns. The main motivation of Zeitgeist is to bring human context awareness to computer systems.

Pensive [166] focuses on augmenting human memory toward better reminiscence by using email and facebook to send memory triggers to users. Triggers will be drawn from people's activity in social media sites in an attempt to provide reminders with personal significance, while generic triggers encourage reminiscing across the entire lifespan. They result of pensive study shows that users value spontaneous reminders to reminisce as well as the ability to write about their reminiscing.

YouPivot [100] is an extension for Google Chrome browsers. It facilitates search based on contextual cues to find semantically unrelated items in users' desktop activity history. Contextual information assist human memory for better information recall, and YouPivot employs this principle of human memory to enhance search of digital information. In simple terms YouPivot assists users associate information,objects and activities during the recall tasks.

2.2 Context-Aware Systems

In the following section, efforts which continuously sense contextual information, or use contextual information to augment users memory, are briefly described. Contextual information provides a powerful cue to recall the past information. The following efforts are not listed as life-logs, but they are similar to life-logs for they continuous sensing or augmenting user's memory. As a result ubiquitous and continuous sensing life-logs could be interpreted as subset of context-aware systems. Most of the new approaches sense and record contextual information via mobile phones. Mobile phones are always carried by the user, therefore they are appropriate for hosting a life-log application.

Forget-me-not Forget-me-not [124] is the first pervasive effort towards recording users contextual information. The main goal of this research is to record users activities in their workplace via a wearable device which is called "ParcTab" [67]. ParcTab devices communicate with transceiver devices by infrared, and those transceivers provide information about the context in the workplace. Forget-me-not has been designed to be a memory aid and helping users to solve their every day episodic memory problems.

Context has been used as a key to index information automatically. ParcTab senses and records automatically the personal location and encounters with other users, workstation activities e.g. emails, telephone calls, file exchange and users printing

activities.
ParcTab has an interface which enables users to search and display recorded information by using an iconic language (to represent actions). It also shows locations and individuals who are involved in the communication.

Remembrance Agent [188] was one of the earlier attempts that uses the pervasive approach for memory augmentation. In this research a wearable device will be used, and this device reminds users with potential information. The process of reminding will be done based on their physical and virtual context.
Rhodes introduces characteristics and a design philosophy of the wearable computers in his article "The wearable Remembrance Agent: A System for Augmented memory" [188]. According to his definition these devices have sensors and interface(s) for user input. They are always running, pro-actively conveying information to users, provide hands-free use and are portable while operational.
The "Remembrance Agent" prototype is composed of two parts: a desktop application, which runs in the background of the user's computer, and a wearable device. The desktop application observes what is typed or read in the word processor. Then it suggests old emails and other text documents which are relevant to the current typed/read text. Another part is the wearable device, which is a pervasive version of the Remembrance Agent and has the same characteristics as the desktop version. The wearable version contains a video recorder sensor and an infrared receiver that communicates with location beacons, which are placed around the Lab.

Nokia's Life Blog [4] was a mobile application for the Nokia S60 series. Nokia called it a "multimedia dairy" and it was recording user interactions with their Nokia phones including: captured photos and videos, sound records, multimedia messages and text messages. All of those information objects were stored with date and time. Nokia life blog enabled users to upload their recorded information to social media web sites or their blog. However, this application has been excluded from the recent versions of Nokia phones.

I Sensed [56] focuses on learning human's life patterns by examining and analyzing collected data from wearable sensors. In order to sense and collect data, a wearable microphone and a video recording apparatus are used. The recorded data from audio and video sensors is examined by machine learning methods and a user model is created which can assist researchers in life pattern learning.

Every Sign of Life [95] tries to encourage users considering their health via biofeedback exercises. Biosensors are mounted on wearable accessories, such as batting belt and helmet, and they continuously sense and record the biological information. Target group of this approach are healthy individuals who would like to prevent health problems.
This approach records biosensor information continuously, instead of periodically, which is unusual for biosensors. Moreover, bio-analytical and bio-feedback games have been proposed to improve users health.

iRemember (What was I Thinking) [226] collects, indexes and organizes a users' everyday audio conversations via a wearable audio sensor and a desktop application. It has been described [227] that the new generation of the personal memory aids are portable computers. It tries to address blocking memory problems by providing a tool to find memory triggers that remedy these problems [229]. In a more technical sense, a microphone records the daily communication of users and keeps them temporarily in the hosting PDA, then these audio files are sent to the user's personal computer. Afterwards, they are converted to text by an automatic speech recognition (ASR) tool. These text based information objects enable users to browse and search them. Moreover, this framework uses local weather and news services [228] to augment the quality of data by providing landmarks for memory recall.

Pensive [21] is another approach which tries to augment the episodic memory of users. The microphone and camera (for taking pictures) of mobile-phones are employed sensors of the research. Users can decide to capture which experiences and manually to do so. Pensive uses date-time and geographical location, to organize the collected data. It provides some social features, e.g. it enables users to blog or upload their captured photos to a website. These features can assist users in sharing their information with others and benefit from social annotation facilities such as tagging.

ContextPhone [173] is a software platform that is composed of a set of open source C++ libraries. It has been designed to sense contextual information and it contains four modules: “Sensor”, “Communication”, “Customizable application” and “System Services”. The sensor module senses and reads contextual cues, The communication module connects to external services via Internet, the customizable applications are a set of components such as context logger, context media, etc. and the system Services are responsible to automatically launch background services, error logging, etc. This platform has been used as a sensing tool in Reality mining projects for context sensing.

MyExperience [88] is another open-source platform for mobile sensing. It enables users to record their phone usage and data from connected sensors. Specifically, it passively logs device usage, user context and environmental sensor data. MyExperience has been designed to collect quantitative and qualitative data. The architecture is based on a sensor-trigger-action and the event model is asynchronous and unidirectional. In order to invoke actions, triggers combine a stream of sensor data with conditional logic. Actions are code snips that are triggered to execute, based on sensor events.
The MyExperience implementation is based on an object oriented approach in C#, and it can be used as a stand-alone application or as a library within other applications.

Experience Explorer [31] is the most recent life-log approach, which uses the mobile phone to senses and capture contextual information. Those context information objects include: Bluetooth proximity, WiFi signal, cellular location, GPS coordinate and date-time. It provides an open interface to communicate with third parties, such as the Flickr photo service [1], to enable users to share media content. It has been designed for life logging and not just continuous sensing, therefore it provides metadata annotation, user interface for browsing information and etc.

[1] http://flickr.com

Affective Dairy [212] is not named as life log, but it employ the life logging style to collect users' emotion and physical experiences. They have used GSR sensor for users' emotion arousal, mobile phone SMS and Bluetooth proximity (social activities) and accelerometer for bodily experiences. It provides users a visualization, designed for tablet PCs, which used to reflect users about their past activities. Moreover they have studied users' behaviors and patterns after reflecting them with their emotions and body experiences.

Jigsaw [138] stated that long-term continuous sensing on mobile phones is a challenge, because of the lack of resources. Jigsaw contains a design and an implementation of a continuous sensing application which balances the performance needs of the application and the resource demand of the continuous sensing on the phone. This is done by resilient data processing, smart admission control and adaptive sensor stream processing (judiciously trigger power to a sensor pipeline). Jigsaw has been implemented for GPS, Accelerometer and Microphone sensors. The Jigsaw itself is an engine which runs in the background of mobile devices. Two applications have been built based on Jigsaw: JigMe which provides a log of the user's daily activity (a simple life-log), and GreenSaw which provides carbon footprints and caloric expenditure information.

Mobile Lifelogger [54] is designed to provide a digital memory assistance to users via life logging. It collects accelerometer and GPS sensor data and tackles the issue of a large life-log dataset by indexing those data objects using an activity language. The indexing approach supports easy retrieval of life-log segments representing past similar activities and automatic life-log segmentation. In general this framework is composed of three parts: "Mobile Client", "Application Server" and "Web Interface". The Mobile Client installed as a mobile application on the user's phone and do the sensing. The Application Server is responsible for storing, pre-processing, modeling recorded data as an activity language and finally retrieving similar activities of the user. The Web Interface is a web application that enables users to browse and access their life-log information.

CenceMe [151] is a Symbian based platform implemented on N95 Nokia Mobile phones. It is designed to infer a user's sensing presence (e.g. dancing at a party with friends) and share this presence through a social network site such as Facebook. As sensor, it employs GPS, microphone, captured picture, accelerometer and Bluetooth. CenceMe contains a mobile application and a backend infrastructure on a server machine. The mobile application performs sensing, classifies raw sensor data in order to produce meaningful data and finally presents the user's presence directly on the phone and uploads it to the backend server. CenceMe challenges the resource limitation of mobile devices by defining an idle interval between sensing cycles and thus reducing the amount of battery usage, besides handing over resource-intensive tasks to the server.

In the next chapters a tool will be introduced that use mobile phones as a life-log platform. Therefore a comparison between the architecture of recent context-aware systems might be useful and table 2.1 lists recent context-aware approaches that use mobile phones for context sensing. They are capable of being used for life logging, but none of them supports *long-term digital preservation* and they did not provide any facilities for *forgetting*.

	Extendable	**Information Retrieval**	**Security and Privacy**	**Sharing**
ContextPhone	n.a.	sensors data used to annotate media	status display keep users aware of whether they are revealing information	users can notify other users and share their data on the server
MyExperience	sensors can be added via the plug-ins	combination of self-report and sensor data enrich the sensor data	n.a.	n.a.
Experience Explorer	throght 3rd party Internet services	visualization and aggregating context with content	secure API authentication for 3rd party services	automatically created multimedia content from sensor information is shareable
JigSaw	n.a.	it is used as sensing engine for other applications	since the JigSaw is an engine and the application which use it should consider them it is not relevant	JigMe is built by based on JigSaw and shares contextual information in social network

continued on next page

continued from previous page

Mobile Lifelogger	n.a.	indexing data by activity language, annotation, information browse interface	n.a.	n.a.
CenceMe	n.a.	through customized tags, and using sensor classifiers	privacy settings GUI, which controls sensors and sharing policy	automatically created content will be shared in social network

Table 2.1: Mobile phone based context-aware approaches

There are other context sensing systems which are being used for a specific purpose and unlike the described approaches they are not frameworks. Those systems include ViTo [157], UbiFit [58], UbiGreen [89] and PEIR [155] which employ mobile phones or PDAs to continuously sense and record contextual information. ViTo uses a PDA as a home entertainment remote control device which records usage duration of TV, music player, etc. It assists users to alter their behavior by embedding behavior change strategies into normal interaction with the device. UbiFit is another ubiquitous based approach, which uses the mobile phone to record physical activities of users. It provides an aesthetic representation of the physical activities to encourage users to perform physical activities. PEIR (Personal Environment Impact Factor), which uses GPS of mobile phones to collect users location changes, and by using an accelerometer sensor data and HMM-based activity classification, determines users transportation mode. UbiGreen proposes a personal ambient display on mobile phones to give feedback about transportation behaviors. This causes personal awareness about transportation activity and reinforces user commitment to ecofriendly behaviour. UbiGreen uses the MyExperience [88] for sensing and learning users transportation behavior.

Chapter 3

Framework Features

This chapter introduces the main features of the framework. First it introduces conceptual features of the framework, which are “Sharing” and “Forgetting”. Then I describe a main technical feature of the framework, which is “Openness and Extendability”. Technical features are not limited to Openness and Extendability, but this is a most salient one. Here I describe its advantages and in the next chapter I go deep into its implementation details. In contrary to the Openness and Extendability Sharing and Forgetting are non technical features of the framework, which have been inspired by our memory functions.

To my knowledge there is no life-log platform which enables users to share their information [177] and sharing life-log information or similar data streams require a specific data model, visualization, etc. The similar scenario is also applicable for forgetting. There is no life-log system (or in general PIM) which exists that enables users to implement forgetting. Here I just describe them in abstract and in the next chapter some minor technical features of the framework in terms of software design and how these features were implemented will be described.

I start by describing the sharing of life-log information with society then risks and benefits of sharing will be described. Afterwards an introduction about forgetting in human memory will be described and then the framework’s approach for implementing forgetting in life-logs will be described. At the end, a brief description of advantages of Openness and Extendability will be given.

Features which are related to security and privacy are not explained in this chapter.

They will be explained in the “Security, Privacy and Trust” chapter.

3.1 Sharing

For the majority of us the Internet is a necessity in our daily life. Lots of our personal information is available online and is easily accessible to other users via the social oriented technologies such as social networking sites (SNSs) or online communities. Sharing information is a fundamental behavioral mechanism of humankind. Users benefit from using features of these society-aware technologies by sharing their personal information with other users. For instance a user can share content, find new friends based on the shared content (common interest) and stay in contact with old friends, etc. These technologies allow users to enrich content via different mechanisms such as ranking, tagging, commenting, etc. In simple terms, content will get enriched by sharing. Most famous content sharing platforms such as Youtube and Flickr offer social network functionality to use crowd annotation and enriching their content.

Soley [209] stated there should be an specifications that individuals’ e-memories could be used and shared by other consumers. Furthermore Berslin and Decker [43] predicted that social networking will go beyond ego surfing in the future. They introduced a social networking stack to let users share information beyond the SNSs domain, e.g. desktop environment. Based on their prediction we can conclude that all users’ digital objects and documents can be shared with society, with enough consideration to the access control limitation. On the other hand, exposing personal information outside private space increases the risk of misuse by criminals, governmental institutions, businesses institutions, and any other third parties. Social computing environments lack governance by their nature [165] and potentially these environments are very prone to misuse. There is always a compromise between sharing information and considering the privacy of the users. There is also less research done on privacy in social communities [159].

User and society can both benefit in many ways by sharing life-log datasets with society such as story telling, historical studies, crowd health monitoring and group behavioral studies [73]. These potential advantages will lead to more business and scientific attention in the near future. There are some art projects [66] which have exposed life-log information to the public, but to our knowledge there are fewer scientific efforts being made in this area. Reality mining [72] is one of the pioneer projects which has this feature. As has been stated before, it focuses on studying social behaviors of individuals in a group. Reality mining uses a life log frame work [123] to learn the social interactions between users via their phone [71]. Eagle (a researcher in Reality mining project) [70] stated following in "Behavioral Inference Across Cultures: Using Telephones as a Cultural Lens" article:

By continuously recording behavior over an extended period of time, it will become possible to quantify peoples reactions to unanticipated events such as an election or an earthquake.

Here first I describe an overview about social media and why they are not a life logging platform. Subsequently I list potential risks and benefits of sharing life-logs information with society. Then sharing life-log information in social network prototype will be explained in the next chapter. A data model which supports a fine grained access limitation and enables users to share their life-log information with society will be introduced (in the next chapter).

3.1.1 Social Media Overview

A Life-log dataset is a collection of the one's life event information. Information is a composition of data in a meaningful way. Furthermore a life-log's data structure must deal with the social networking systems (SNS). In order to be able to define a data model that enable users sharing their life-log information we need to identifies roles in a social community. Also it is important to note that available SNS such as Facebook are not life logging tool, but they provide a rich data source that can feed into a life-log dataset. Here I describe why they are not capable of hosting life-log data. However the current SNSs architecture and data model has inspired my work. More about implementation detail and data model will be described in next chapters.

Roles in a Social Community

In order to be able to define a model for sharing information, we need to identify roles in the context of a social community. Here roles have been identified based on existing operational SNSs, in a very abstract form. As a result I did not consider the business process of developing a tool during the role identification. In a technical sense, unlike Collins et al. [57] model I did not consider software buyers, software developers, etc. Here three roles in a social community has been identified: *end user*, *community owner* and *third party*. The end user is the person who benefits from the system in return for exposing his personal information to the community. The community owners are people who own and manage the social community and benefit from having users and in all likelihood use their information. Increasing the number of users raises the community value. Community owners are responsible for the service that they provide to users. I assumed that the service will be created, implemented, administered, owned and managed by community owners and will be used by end users. The third parties are groups or individuals who benefit from communities by using a community environment to advertise, perform research on the community, study users' behavior, monitor community member activities, etc.; they are able to access user's information in the community with or without the end users' awareness; they could vary from marketing businesses, security agencies to hackers. In order to use the user's information, third parties must come to an agreement with community owners. Community owners should inform users about this in their 'Terms and Conditions' section of the community's membership agreement criteria.

End users share their information with firms and other community members based on trust [160]. From users' point of view third parties have the least credibility in the community; they observe users' activities in the community and benefit from it. They can also be the business focus of the community owners. In other words, community owners benefit from providing community information to third parties.

Social Media and Life-logs

Schneier [199] identified five classes of shared information in SNSs. These classes are: service data, disclosed data, entrusted data, incidental data and behavioral data. He discovered that disclosed and entrusted information is being created by users. Incidental data will be created by other users about the user. Service data are information that users give to the SNSs during the registration, and behavioral data are information that is being collected by the service provider about users' activities. Although there are some similarities between SNS and life-logs, this categorization is not applicable for a life-log social network. A life-log social network similar to other social networks includes service data and behavioral data but the sharing mechanism of life-log SNSs could not be similar to traditional SNSs.

Social media services such as Twitter, Facebook and Friendfeed are designed to enable users to share their online activities and personal information. Although they enable users to share information we cannot consider them as life-log tools since they have major differences with life-logs. I list their differences in the following.

The main use of life-log tools is in the private domain and not in the social domain. SNS, by their very nature, only provide social networking features and they do not provide enough functionality to host private information. In other words, whatever we give them as an input is not private. However they could be interpreted as an output channel of life-log data.

Despite the existence of automatic sharing capabilities, users share information via social media services manually. Their design is based on manual user interaction with the system and users' data are not streamed automatically. Life-logs sense contextual information of the user automatically and it is not possible to rely on manual user inputs. It is notable that the content provided by users to SNSs might be assumed as a life-log sensor.

Their information visualization or reflection style is another salient difference, which makes them different to life-logs since, based on their design criteria, they should host a restricted amount of information in comparison to a life-log. They might be used as an output channel for a few life-log sensors, but they cannot host life-log data. A life-log with sharing facilities will have similarities with SNSs. They can both share some information with the public and some with a list of users based on

the owners' preferences. Users use them to perform self-reflection and share information about themselves with society. The proposed data model is inspired by the available SNSs, but is designed for life-logs. Therefore, we will introduce a private scope, use timestamp as a fix factor and define a 4-tuple data entity. These factors guarantee the granularity of the data in the life-log dataset.
Moreover available social media does not support forgetting, and a shared information object should be removed from shared status by the user manually. A platform for sharing life-log information with forgetting feature will be explained in more detail, at the end of this section.

3.1.2 Risks and Benefits

In order to be able to analyze the cost of sharing the life-log information, Risks and benefits have been classified separately. Risks and benefits mostly have been identified based on theory. This means it is possible that in the future more risks or benefits will be introduced in this domain, because few experiments have been done with the real operational system and life-logs are in the introductory phase. Collins et al. [57] proposed a rather similar strategy based on John Rawls theory of Justice [186] toward assessing the quality of software products. First they identified roles in creating and using a software application and then they performed risk assessment for each role. Later, ethical considerations for each social role in their scenario were proposed. My approach is similar to theirs, but here the intention is focused on a specific type of application (life-logs). Here risk-benefit assessment has been performed based on the study of available research and similar systems such as SNSs [24, 29, 217]. Baloi and Price [29] noted that many different classifications of risks have been developed such as legal, natural, logistic, social, etc., which are based on source criteria. Tah et al. [217] suggested risks can be classified as dynamic/static, corporate/individual, internal/external, positive/negative, acceptable/unacceptable and insurable/non-insurable. I do not suggest any classification here because it might be possible that over time more risks will be identified which will not fit in any of the defined categories of life-log risks and thus the classification may become outdated.

Benefits

In the previous chapter benefits of sharing life-log information have been described, disregard sharing. Here I describe advantages of sharing life-log information with society which could benefit individuals in educational, business, health cares and social relations. Some of the identified potential advantages of sharing life-log data to society have been listed as follows.

- *Software Personalization*: Software applications can be personalized based on interests of the group (social interest). Software personalization could facilitate search and information retrieval. Studying and mining group preferences can be done based on the past activities (life-log dataset) of the group. For instance, when a user searches for “ant” in a software development environment, it means “Apache ANT” [1] and not the insect ant. In simple terms, applications can be personalized based on the groups characteristic and group characteristic is identified by studying life-log datasets of group members.
- *Learning Social Patterns and Behavior*: Life-log data can be used to learn the social behavior of a group. For instance in the case of earth quakes, location changes of the individuals can be logged by the location sensor of their mobile phones [70], to assist scientists studying how they can reduce the damage.
- *Match Making*: The first requirement to start a new social relationship is to have a common interest. Common interests between users can be extracted from their life-log dataset. Again Reality mining [71] performs matchmaking based on the user profiles and behavioral data of the user. There are more potential commonalities, which can be used to suggest a match, such as the user’s habits (e.g. eating habits of the subject can be extracted from locations of visited restaurants).
- *Recommendation Systems*: A recommendation system can study and extract knowledge from the life-log dataset of the users. For instance place ’A’ has been visited more often than the place ’B’. A recommendation system can suggest that the place ’A’ is worth visiting more often than the place ’B’.

[1] http://ant.apache.org

- *Health and Medical Studies*: Biological sensors can be used on a subject's body to sense and record biological data such as body temperature, skin conductance, body heat, etc. This data can be recorded and used as the life-log information. The health status of society can be monitored via provisioning and monitoring life-log information of the individuals. Sharing personal health information with society can assist prevention or determination of an epidemic [163] or it augments the monitoring of epidemic distribution.

- *Historical Studies*: Memory augmentation is the main goal of life-logs. When life-log data is shared, it can assist individuals in sharing their life events. History composed from the life events of individuals. Maintaining a life-log dataset in the long term is a valuable source which might be used for historical studies [49].

- *Sousveillance*: This terminology is different than surveillance. Surveillance means recording the individuals activities by technologies on behalf of an organization. Surveillance systems are not hazardous by default, they support safety, welfare, health, efficiency, speed and co-ordination. A life-log could be interpreted as a form of personal sousveillance [146, 163] which is bidirectional, not like surveillance unidirectional (government monitor individuals). Sousveillance can democratize the process of surveillance via supporting the monitoring of the authorities (surveillant).

Risks

As has been described in the introduction, sharing personal information has risks and benefits; Losing privacy is the most important risk of sharing personal information. Unfortunately life-logs have a controversial history, such as a DARPA's lifelog project [203] which was canceled in 2004 because of criticism of the system's privacy implications. The Life-log dataset contains sensitive personal information such as the user's location, which requires more privacy concerns, because sharing personal information with this level of detail can be hazardous. Anita L. Allen [24] identified two important potential hazards of life-logs: pernicious memory and pernicious surveillance. In order to be able to provide a data model and identify ethics that reduce the potential hazards, I describe them here in a different classification and I try to identify more risks. As with benefits, I can not argue that all potential risks have been listed. More risks may be identified, when a life-log tool is used in a real operational environment and for business purposes.

- *Surveillance*: Sharing life-log information with society might be interpreted as a form of surveillance. The surveillance, which limits our behaviors without our desire, is not acceptable. Surveillance has some potential disadvantages such as increasing the number of suspects who are not guilty [140]. Security agencies, governmental organizations, business organizations, criminals and other types of organizations or industries which are benefit from social control can misuse surveillance data. For instance a recruitment company can check a candidate's psychological and physical health and his life style during the recruitment process, in order to find out that if the candidate is suitable for the offered job or not. If the candidate performs extreme sport, the chance of having time off for medical purposes will be high [238]. Existing laws and policies do not provide an appropriate limitation on the unwanted use of personal information. In the U.S. CAELA (Communications Assistance for Law Enforcement) enforces communication technologies to allow government access and surveillance [12]. Also, in March 2006, the European Union adopted a Directive, which mandates that the content of electronic communications services will not be deleted (remain for not less than six month and not more than two years). They can be used for marketing and provisioning proposes [13]. Observing users in a community with out their awareness is the worst form of surveillance. This form is longstanding. First time Jeremy Bentham (1838) defined it as Panopticon, then Michel Foucault [83] compared Bentham's ideas with modern society.

- *Memory Hazards*: It has been proved that life-log tools can assist human memory [201]. Life-logs can record all life events, disregarding the content. It means a life-log can prevent individuals from forgetting their errors [66]. Individuals naturally try to distance themselves from their errors and misfortunes because, psychologically, they need to forget their misfortunes. Life-logs could be harmful in this case and they can also cause pathological rumination for unipolar and bipolar depression [18, 24]. Exposure of personal mistakes to society might be more harmful than keeping them private. For instance children by their nature are weaker at bearing misfortunes, they can go through their parents' life-log and remember a misfortune which happened in their family. Another problem can occur by exposing the mental problems of individuals, which could have a negative impact on group mentality in a team or a group of individuals sharing a common interest. To handle the memory hazards of life-logs, Dodge and Kitchen [66] suggested that the life-log should forget like real memory, based on the Scharter's forms of forgetting [195]. I believe fading life log memory will distance it from its main goal as an ultimate memory aid because we will lose the reliability and persistence of digital memory. Therefore I propose forgetting based on the users' demand in the next section.

- *Long-term availability of personal information*: The ideology and personality of an individual can change over time. Besides, an individual's lifestyle may change over time. Sharing life-log information with society can be a permanent record of our mistakes. For instance, imagine a group of teenagers gathered at a party and posing for a photograph in an embarrassing way. These pictures go online and everybody can see them. At that time there is no problem. Thirty years later one of those teenagers is now going to participate in an important political campaign, these pictures could harm his career. These problems are not only relevant to life-log systems; other online traces have these problems too. Taking another example, consider a writer who has changed his mind over time and disagrees with his past opinions. In the era of the Internet the chance of removing his old ideas from public view is very low. Miller [150] identified three methods which have been used by third parties to gather individuals' personal information from the Internet. The first method is direct inquiry, such as the registration information of the individual on a website. The second method is by aggregating information about the individual from different sources such as an Internet search of his name. The third way is by observing individuals' activities, e.g. cookies which reside in the user's browsers to track his web page navigation. Based on the second method, long-term availability of the user's information could be harmful. How should we deal with long-term availability of the personal information on the Internet? Is there any chance of manipulating our online traces? How can we protect our social traces on the Internet? These are some of the open questions in this subject.

- *Stealing life-log information*: Sharing life-log information increases the chance of loss or theft. This risk can be very harmful for the person (victim). Strahilevitz [214] claimed that most private information consists of sensitive personal matters such as sexual encounters and bodily functions, sensitive medical information, knowledge of the owner's fundamental weaknesses, etc. The life-log can sense and record this information, therefore a life-log dataset is a very sensitive object from a privacy point of view.

The described risks indicate that life-logs could be harmful as well as being useful. With the exception of memory hazards all other risks are related to the sphere of privacy. Therefore, any associated data models for life-log tools must carefully manage privacy related issues.

3.2 Forgetting

Humans by their nature try to remember and preserve knowledge, therefore they have created many devices and mechanisms to assist them in challenging forgetting. Forgetting is a loss of information objects that are already in long-term memory. A major form of forgetting is decay, which means that when an information object is not retrieved for an extended period of time it fades away. In fact the forgotten information object is not removed from our memory, just the direct link to that information object will be removed. Moreover it is important to note that forgetting is not a weakness of memory and it is a necessary feature, because by forgetting we gain the ability and freedom to generalize, conceptualize and act. Anderson et al. [25] stated that memory performs an optimal adaption to the environment through forgetting. This means that the brain constantly reconfigures the memory based on the present preferences and needs. Besides there is a trade-off between reducing access to an unused information object and the annoyance of costs of forgetting.

Sigmund Freud [86] is wellknown for his theories about the mechanism of repression (unconscious form of forgetting) and unconscious mind. Hermann Ebbinghaus [75] is one of the first scientists who studied the mechanism of forgetting by using himself as a subject for his experiment. He designed a forgetting curve which illustrates the decline of memory retention in time, and he found that forgetting occurs in a systematic manner and his premises are even true today.

A recent theory for remembering and forgetting has been proposed by Schacter [195] in his book "How the Mind Forgets and Remembers - The Seven Sins of Memory". He identified and classified memory errors in a framework, which conceptualizes causes and consequences of memory's imperfection. Schacter stated that these errors are necessary for human memory and he classified memory errors to "transience", "absent-mindedness", "blocking", "misattribution", "suggestibility", "bias" and "persistence". Transience, absent-mindness and blocking are the outcome of omission. Transiences means weakening or loss of memory over time, e.g. we do not remember what we have for dinner six months ago. Absent-mindness is the cause of breaking interface between attention and memory, e.g. misplacing keys or forgetting and appointment. Blocking is a result of unsuccessful try to retrieve an information from memory e.g. we can not retrieve a name of a familiar face.

In contrast to these three malfunctions misattribution, suggestibility, bias and persistence are all outcome of commission. In particular some form of memory is present but it is either incorrect or unwanted. Misattribution is assigning a memory to the wrong source, or in other words mixing fantasy and reality. Misattribution is very sensible malfunctions and has a profound implication in legal settings. Suggestibility refers to memories that call events which might have happened but they are not happened in reality. Bias reflects the powerful influences of our current knowledge and beliefs on how we remember the past. In other words we often edit or rewrite what we now believe or know. Persistence refers to repeated call of disturbing events which we prefer to vanish from our minds. In some extreme cases persistence can be life-threatening and causes depression or traumatic experiences. However remembering disturbing events assist us in avoiding the re-occurness in future.

In order to implement forgetting for life-logs, I use the well known classification of forgetting, which is used in self-knowledge literatures [236]: unconscious forgetting or repression and conscious forgetting or suppression. The goal of this dissertation is to challenge unconscious forgetting by life logging, and enable users performing suppression which allow them to remove unwanted information from their life-log dataset. In simple term life-log tools should hinder unwanted forgetting (e.g. misattribution, suggestibility, bias) and assists users in the forgetting, which is based on user desire.

3.2.1 Digital Forgetting

Mayer-Schoenberger [148] in his book "Delete: The Virtue of Forgetting in the Digital Age", provides a profound clarification about the need for digital forgetting and its necessity. He traces the importance role of forgetting in human history and described that digital technology and global network override humans' ability to forget. According to his definition durability, accessibility, comprehensiveness are three factors in the digital era that disable individuals from forgetting. In the analogue age noise acted as a barrier against mass copying and this could be interpreted as slow but acceptable form of forgetting. Digital forgetting is a major issue of our age, especially after the Web 2.0 evolution which converts the traditional role of Internet from information *access* tool to a tool for access and *share* information.

What we are reading or hearing from news constitute the foundation of society's shared common memory (public news become the foundation of our society memory) and remembering does not just apply to personal memory, it even applies to corporate memory. He warns that:

"By recalling forever each of our errors and transgressions, digital memory reject our human capacity to learn from them, to grow and to evolve."

Sharing an information costs the information owners to lose their control on the shared information, because information spread quickly in the digital environment and number of individuals who access those information could be very large. Furthermore the digital copy of an information mostly is a perfect replica of original information object. And in the digital age preventing copy is somehow impossible. On one hand, as has been described before, sharing memory [2] is not just harmful

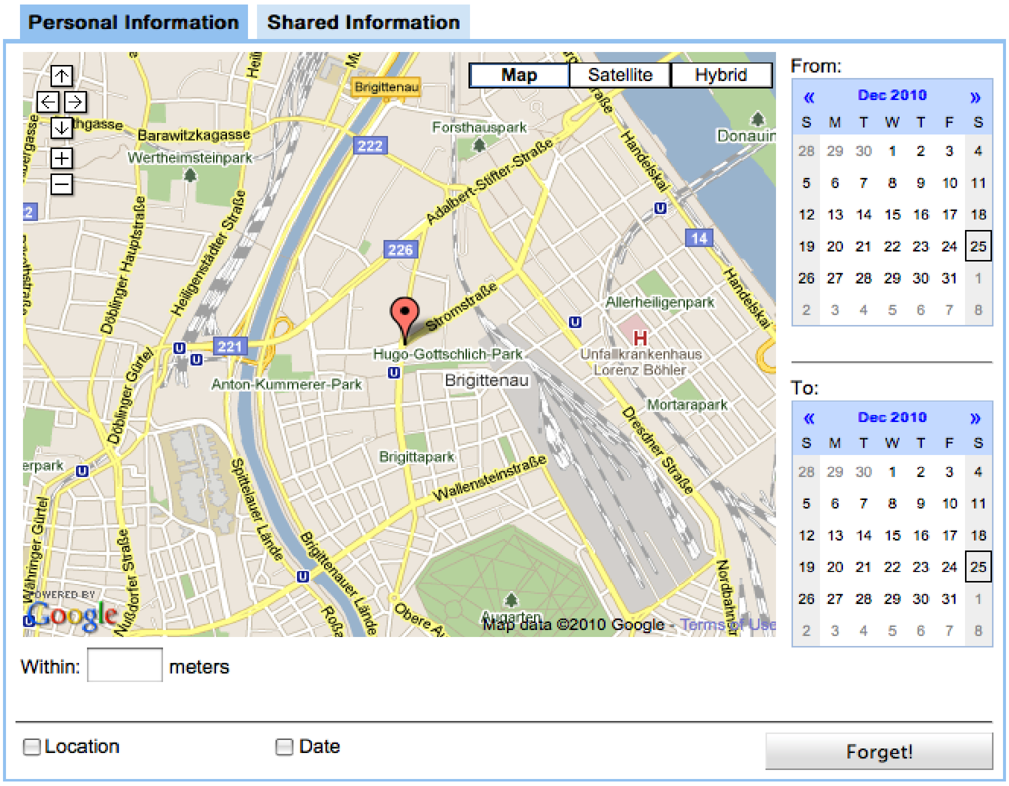
[2] Here by "shared memory" I mean sharing personal memory with society and by "community memory" I mean a collection of shared memories such as twitter posts in a city.

and different parties can benefit from shared memory. For instance it can change industries and enterprises behavior, which in the end results in more accurate decisions by decision makers e.g. politicians keep their promises. On the other hand, according to Mayer-Schoenberger, an e-memory with no forgetting supports may (1) undermine users' ability for reasoning, (2) exacerbate the difficulty of putting users' past life events in proper temporal sequences, (3) confronting them with too much information about their past and thus remove their ability to decide and act in time and (4) users may lose trust in their memory when their human memory is in confront to their digital memory.

The result of life logging creates a personal digital memory, and in terms of forgetting they have two major differences [148] with human memory. First digital memory is not more trustable than human memory, this due to the fact that digital memories are malleable and alterable but the human memory can not be altered as easy as digital memories and this feature makes them not trustable. For instance somebody can hack and get access to the digital memory and manipulate its content. Second, when the brain refers to a past event it *reconstructs* the event, but e-memory *recall* the past event as exactly it was stored in its underlying dataset. Since e-memory operates based on recall and it can not exclude random piece of events, reminiscing by using combination of e-memory and human memory can confuse users with conflicting memories, and thus affect their judgement.

This framework defines two type of forgetting: Personal Forgetting and Social Forgetting [182]. Forgetting for the shared information (social forgetting) will be achieved by defining *information expiration timestamp* and for the personal information will be achieved by *annotation*. All life-log information objects which will be shared (based on users demand) should contain expiration timestamp. Other information objects which remain private do not need expiration timestamp and they could be annotated for forgetting manually by users. In a more technical sense users can decide what to forget based on location and/or date-time of the information object as shown in figure 3.1, the interface has been inspired from our spatio-temporal life, more on spatio-temporal life will be explained in the next chapter.

Users can select a specific area on the map or/and specific date-time (figure 3.1) to forget the shared or the private information object(s). After the user selects an area on the map or a date-time period, she can click on the forget button and infor-



From:

«			Dec 2010			»
S	M	T	W	T	F	S
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

To:

«			Dec 2010			»
S	M	T	W	T	F	S
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Within: meters

□ Location □ Date Forget!

Figure 3.1: Spatio-temporal based forgetting. User can specify location and date period, then based on these two factors, all information can be marked as forgotten in the dataset.

mation objects from the selected location or date-time period will not appear in the query result anymore. When users mark an information object to share, they should define the expiration date-time for the shared information object. This means that any shared information object will be removed from the target community when the expiration date-time reached.

Figure 1 shows forgetting for shared information, it shows that User A created a data object at time t0, then shares this data object at time t1 with User B and a group of Users (User Group C). User B's access to data object A will be expired at time t3 and Users in Group C cannot access to this object after time t5.

When users query or search their dataset, through their query interface, the query engine will check the annotation file and bypass forgotten records. In simple terms the forgotten information objects are still in the dataset, but annotate as forgotten.

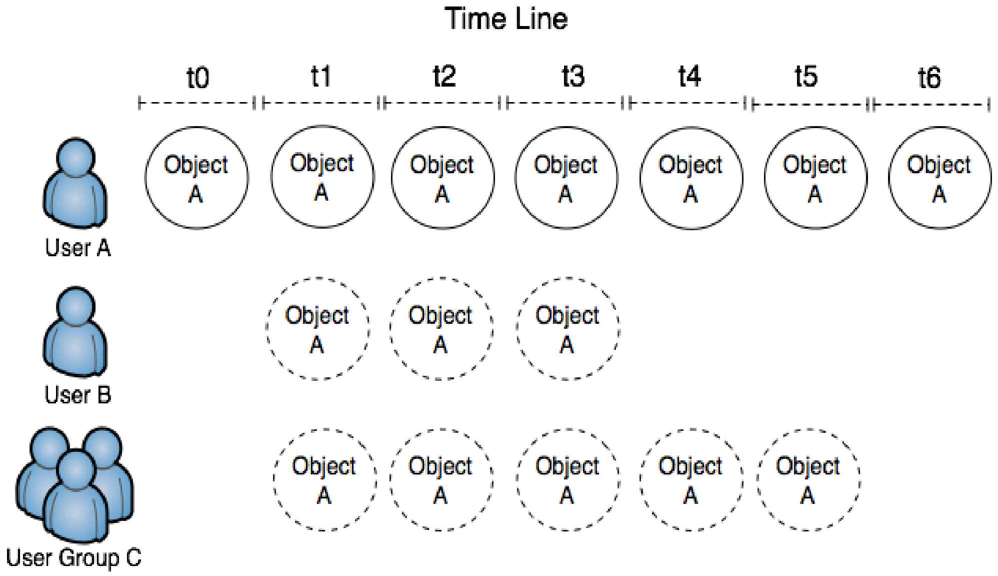


Figure 3.2: Access expiration definition for a shareable information object. However the implementation must consider limiting the digital copy too, because a digital copy create an exact replication of the digital object without any owner control on it.

Although forgetting is necessary, a major role of life-logging is to preserve memory and move against unconscious forgetting. Bell and Gemmel [35] defined the term *digital immorality*. They predicted that in the future, digital memory of an individual could be inherited to heirs and they can communicate with an avatar of the digital memory owner. The avatar has similar visual and behavioral properties as the digital memory owner and simulate his or her presence.

3.3 Openness and Extendability

Life-logs sense and store users' contextual information from their environment through sensors, which are core components of life-logs.

To date there are only few life-log tools with very restricted features available on the market. This restriction is because of their software architecture. A major technical feature of this dissertation is to provide an open and extendable framework. openness and extendibility will be achieved through providing flexible (in terms of adding new sensors or removing existing sensors) and configurable sensors.

This means users can add/remove (extendability) or disable/enable (intelligibility) sensors or change settings of available sensors. The architecture of the framework will support openness and extendability, which results in a holistic framework that can be used for different purposes (multipurpose).

In terms of software design, openness and extendability from sensor perspective which gives us the following advantages:

- **Memory Augmentation:** It has been proved [201] that life logging assists humans in memory augmentation. Furthermore similar approaches [124, 226, 21] have been used to augment human memory or in particular reminiscing. The process of memory recall refers to the contextual information which has been stored within the event [38].

  Moreover, landmarks [135, 189] assists the retrieval of information from memory. Contextual cues could be stored as landmarks and therefore they lead to better retrieval or recall from the dataset. For instance consider a scenario in which a user does not remember what she had for lunch the last three weeks on tuesday, but she checks her dataset and she realizes that at that time the weather was very cold and then she remembers she was in an asian restaurant when the weather was cold. In simple terms increasing the number of sensors (cold weather in this example) in the life-log dataset increases the chance of recall and retrieval and even less memorable events can be recalled with more memorable landmarks. Gemmell et al. stated following in their well known article entitled "MylifeBits: A personal database for everything" [91]:

  The more system logs, the better the chance of having the "memory hook" that *will help you find what you seek.*

- *Multipurpose Life log Framework*: As has been stated before the result of this dissertation is a multipurpose life-log framework which can be used for different purposes. For instance one one hand it can be configured to record health related information and monitoring users' health and on the other hand it can be configured to monitor individuals location changes and working patterns in a firm. This will be achieved by adding or removing sensors such as biosensors or location sensor (GPS).

- *Wide Usage via Extension*: Empirical studies show that platforms which are extendable and capable to accept new components have been converted to de-facto standards in their domain. For instance Eclipse IDE [3] is now widely used as programing IDE, Mozilla FireFox browser is the first browser that enables developers to extend it through its Add-ons. Openenss in terms of sensor configuration might benefit this framework in this aspect.

- *User Modeling Enhancement*: In life logging scenarios the quality of the data will be increased by increasing the quantity of data resources (sensors). This is due to the fact that increasing the number of contextual resources results in gaining better perception about the environment [90, 53]. User modeling and other related fields which rely on analyzing users' behavior and past activities benefit more through having more information about users.

- *Self-Insight*: Life-logs assist users in answering "what", "where", "who" and "when" questions about their past activities. Since we are living in Spatio-Temporal world sensors' data can be linked based on location and timestamp. A combination of sensors' data provides more precise information than single sensor data about users [55]. To my knowledge there is little research about querying a life-log dataset based on combination of sensors. For instance Beat Story [161] combined pictures from a head mounted camera with the heartbeat rate of the user. Or combining a location sensor with a health monitoring sensor assists care givers in finding individuals with a matching blood group [225]. These examples rely on fix number and type of sensors, it

[3] www.eclipse.org

might be possible that we discover novel information about users' emotional state and behavior by increasing the number of sensors. For instance consider a scenario in which a user employs location, social proximity and biosensors and she decides to measure her emotional state with her bio sensors. She can realize that appearances in a specific location (location) or visiting specific individuals (social proximity) have a significant impact on her emotional state (bio sensor). Moreover sensors data might assist her in better self-awareness, for instance what is the effect of weather changes on her emotions ? As a result I can claim that a combination of different sensors can lead to correlation and multi-dimensional behavioral pattern identification.

How the implementation instances of the framework support this features will be explained in the next chapter.

Chapter 4

Framework Description

This chapter describes the framework architecture in detail and based on the proposed architecture different implementation of the framework within their evaluation will be described. First I start by defining requirements of the framework and then I will describe the technical architecture and related evaluations. In order to describe requirements of the framework I start with defining a sensor classification, then I describe life-log data structure and afterward the data model. Moreover I describe design challenges and considerations which will be handled during the system design. Next I describe the system, which has a client-server architecture. Each client and the server has their own component and these components will be explained in detail.

In order to prove the framework tools and prototypes based on this framework architecture will be described and evaluated. In general two prototypes will be introduced. Each represents an implemented instance of a sensor class from the framework sensor classification. However I do not propose any prototype for desktop sensors, because there are many tools available which support life logging for desktops.

At the end of this chapter I describe two knowledge detection approaches. This can benefit a life-log system in annotation, prove the usability of the system and can be used in altering users' behavior.

4.1 Sensors Classification

In order to achieve a flexible and configureable life-log framework, the flexibility of sensors should be a major consideration. I use the term sensor to refer to an autonomous tool, device, application, etc. which senses or reads data from its surrounding context. Sensors should not be embedded in the framework and they could be distributed in environments which users participate in.

It is not possible to describe precisely which sensors can be used for life logging. This is due to the fact that there are many sensors available that can be used for this purpose and in the future more new sensors will be introduced to this domain. Therefore I provide a sensor classification which is based on the physical proximity of sensors. The sensor classification does cover functionally of the sensor and does not consider the type of sensing such as direct or indirect sensing [90].

Here life-log sensors have been categorized in the three classes: *Desktop* sensors, *Proximity (Environmental)* sensors and *Mobile* sensors. It should be considered that there is no specific border between the categories and one sensor could be used in more than one category. Usually a sensor belongs only to one category, but it is possible that the sensor information will be acquired from more than one category. For instance a weather sensor could be a web-service (desktop sensor) which provides weather information of the target environment or it could be a temperature sensor which is wall mounted (proximity sensor) or integrated into the user's watch (mobile sensor). As another example it might be a location sensor which can be identified from the WiFi signal strength (Proximity sensor), desktop application (desktop sensor) or users' GPS device (mobile sensor) [108]. As a result the framework is flexible enough to accept and manage any kind of sensor.

Mobile sensors of life-logs can be interpreted as a subset of context-aware sensors, therefore life-log sensor classification has some similarities with context-aware sensor classification. Indulska and Sutton[108] classified context-aware sensors as Physical, Virtual and Logical. Physical sensors are ubiquitous sensors which are hardware e.g. GPS and virtual sensors are equivalent as desktop sensors here. Logical sensors are sensors which get their information from physical and virtual sensors to infer information such as location. This framework is capable of aggregating sensors' information together and extract new information objects, but this will be done on the server and not on the Data Reader side, this is due to the computational overhead of sensor data aggregation. As another example of life-log sensors I can refer to OH́ara et al. [163] who identify information sources that are worth to feeding into a life-log system. They did not provide any sensor categorization and they refer only to specific sensors.

In the following each sensor class will be described in more detail with examples.

4.1.1 Desktop Sensors

A significant amount of time of many individuals is spent in front of their computers, this means personal computers involve many daily activities of users. It might be argued that term sensor is not appropriate to be used for the desktop sensors, but as I have stated before that sensors here refers to any component or application that reads and records data. Likewise that component or application can be plugged or unplugged into/from the life-log framework. In particular any software service or application which monitors and records users' activities such as application usage (e.g. Wakoopa[9] and SLife [6]), file exploration, chat content and timestamp, users' online activities such as their Facebook activities, blog posts [99, 21] and users' comments, etc. on their personal computer could be a desktop sensor.

PIM (Personal Information Management) applications rely heavily on desktop sensors. They employ desktop sensors to monitor and track user activities such as working with documents, pictures and the calendar. Information retrieval in desktop is a major challenge for desktop optimization and many PIM tools target this challenge such as iMemex [65], SEMEX [48], Haystack [17], Life stream [78] and Stuff I've seen [68].

There are plenty of tools available which can be used as desktop sensors, and I have introduced them in "Related Works". Therefore I did not implement any desktop application for feeding into a life-log dataset. However some information such as calendar appointments will be used to annotate the dataset in the prototype implementation.

4.1.2 Mobile Sensors

These are sensors which are always carried by the user. For instance body or watch mounted sensors and mobile phone sensors. Weiser [235], who is known as the father of ubiquitous computing, stated that humans live outside desktop computers and most of their life events happen outside computers. Since ubiquitous devices are always carried by the user and they are not restricted to users' desktop activity, they provide very rich information about users' lives such as location changes, biological information, etc. These sensors collect contextual information. According to Dey and Abowd [62]: *Context is any information that can be used to characterize the situation of an entity.* Rhodes [187] described the importance of wearable computers or recently called ubiquitous and pervasive devices as follows:
"Unlike desktop computers, wearable computers have the potential to "see" as the user sees, "hear" as the user hears, and experience the life of the user in a "first-person" sense. They can sense the user's physical environment much more completely than previously possible, and in many more situations."
Mobile sensors which can be fed into the life-log dataset could be: Mobile phone information which includes Call log and content, SMSs, consumed Media, application usage, location sensors (could be extracted from GPS, CellID, Location badge e.g. Bluetooth, and WiFi), Audio, Picture and Video.
All of our life events happened at specific location, hence location is a very important information resource. However it is not always possible to read location and there is research which uses a combination of different sensors and approaches to identify a precise location such as Placelab [197].
Biosensors are other examples of mobile sensors such as skin conductance, heartbeat, body temperature, blood pressure and accelerometer (motion detection). Accelerometers can be used to identify users' physical position such as walking, sitting, standing [115, 20].
Life-logs are well known for using video, picture and audio. Recording users' communication through an audio recording sensor assists users in memory augmentation such as iRemeber[226] and Pensiv[21] approach. Video and pictures provide rich contextual information for life-logs. Startlecam[102], SenseCam[103] and eyetap[145, 144] are examples of using a body mounted mobile sensor to capture

daily life events by recording video or taking pictures. The data will be captured frequently and automatically, but manually stopping or starting the recording process should be possible [103]. Although these sensors provide rich information about users' context, they create large amounts of data. Therefore storage and retrieval of this type of data is a major challenge in life-logs [20] and there is much research focused on challenging this issue such as [99].

Perhaps video, picture and audio sensors could be placed in the biosensor category too, as it has been described before there is no precise border between sensor classes. Another important sensor in this category is a social proximity sensor, which senses proximity of other individuals near the user. Low frequency radio and low coverage domain sensors such as Bluetooth, ZigBee and RFID [219, 169, 191, 129] can be used as the social proximity sensor. Since Bluetooth has been widely used in mobile phones, nowadays most social proximity sensors rely on Bluetooth.

Further in this chapter I describe UbiqLog, a tool which employs smart phone sensors for life logging. Implementation of UbiqLog is a proof of concept for this framework.

4.1.3 Proximity (Environmental) Sensors

Proximity sensors are distributed in the users' environment. In other words, similar to other sensors, they are located in the context of users. Kim et al. [117] introduce gadgets that are located in the environment near the user and attach smart sensors to the daily supplies such as cup, chair, etc. Here proximity (Environmental) sensors use similar terminology. In particular these sensors will be used for life logging but not from the the user point of view, they are distributed in users' environment. For instance a still camera which is used to point at the user, and not pointed by the user [66] or an external hardware or a software component which records radio and T.V. usage of users from their home media server.

More about proximity sensor will be described further in this section. A metaphor will be introduced which performs life logging on a proximity sensor (wall mounted camera).

4.2 Life-Log Data Structure

Life-log tools are used to capture users' daily life events. Life-log information can be interpreted as a subset of personal information. A life-log contains a dataset of a subject's life events. Each life event of the subject is a record in this dataset. In the proposed data model, each life event (life-log record) has an access scope and each shared object has expiration timestamp. The expiration timestamp is used to implement forgetting for a shared information, private information can be annotated to get forgotten manually by users. Life-logs use sensors to read and capture contextual information. In particular, they acquire data from reading sensors. Information flows to the life-log from multiple sources (sensors) in a continuous manner. There is no guarantee for the sequence of received data and the sequence varies over time. Additionally, life-log dataset size is unbound. All these properties indicate that a life-log dataset is a kind of data stream [26].

We are living in a spatio-temporal world. This means that all of our life events except dreams happen in a specific location and at a specific date-time. Based on the currently available technology it is not always possible to sense the location, because location sensors such as GPS do not function in every environment. For instance GPS cannot work indoors. There are other approaches such as A-GPS (Assisted GPS) to solve this problem, but they are not always able to sense location and they are imprecise. On the other hand, most operating systems have date-time which is accessible as long as the target device has not been turned off. This means most devices with computing capabilities can provide timestamps. As a result date-time is a necessary field for any lifelog record and all life-log information objects will be stored with a timestamp.

4.2.1 Annotation and Data Collection

Life-log datasets host large amounts of contextual information about owners and their activities. In addition, dataset size increases continuously and information retrieval is a major challenge (WORN problem). Since the data comes to the life-log dataset from heterogeneous sensors, basic issues that are being solved in RDBMS are critical challenges in the context of PIM and also life-log systems [112].

In this framework sensors' data interpreted as the contextual data and annotation (users' input) as the content. Usually raw sensor data does not contain semantic and therefore annotation used to semantically enrich it. Annotation is attaching extra information to a piece of information [106]. Annotation can be done either manually or automatically. Due to the fact that life-log dataset structure is always growing in size, manual annotation is not feasible and cumbersome. According to Hunter [106] annotation can be stored embedded in the information objects or separated from the target information objects.

Annotation can be used for different purposes such as indexing, bookmarking, assigning meta-data, etc. Here by annotation I mean tagging and thus metadata creation which is data about the data. Metadata information objects could be extracted from other resources and they will be linked to the sensor information based on date-time and - if possible - location. Location and timestamp are two links that bind different information objects together.

Wheeler and Reis [122] stated that collecting data about life events and experience could be performed in three ways: 1) Interval-Continent recording: Subjects record their life events in regular predetermined interval. 2) Signal-continent recording: Subjects record their life events whenever signaled by researchers. 3) Event-continent recording: In this method events will be recorded every time an specific event happened (with predefined conditions). Most life-log systems support combination of these three recording style. For instance Sensecam [103] is taking picture in interval-continent style and it supports event-continent by enabling users to click a button and take a picture. Usually annotation (content) which requires user intervention are event-continent recording, and they used to augment the quality of raw sensor data (context). Annotation is not necessarily coming from event-continent recording or in simple term manually, it could also be done automatically e.g. sex and number of speakers can get extracted from the recorded audio data [117].

Life-log data types vary based on the sensor output. Data can be a text or a binary object such as a movie or a picture. Here all records of a life-log dataset should be stored with a date-time without considering the data type of the record. They can be recorded in a continuous-time manner or a discrete-time manner. Audio files and video files are examples of continuous data. In simple terms if there is a start and end time for an event, then it is a continuous-time event but when there is a unique time, it is then a discrete-time event e.g. location, picture, blog entry. The proposed data model has no dependency on the type of life event (discrete or continuous). Continuous-time data can also be called time series data too [96]. Figure 4.1 shows continues and discrete sensors data collection based on the time.

Another important fact to consider is that all sensors in the implementation should be able to record data without or with less user intervention. This is due to the fact that life-log dataset structure is always growing in size, therefore manual annotation is not feasible. In some special occasions persuasive approaches such as games could be used to motivate users annotating their life-log information, but this is beyond this research research. Nevertheless manual annotation is possible in this framework, because tags associated with a particular data object represents how users' associate their knowledge to that data object [239].

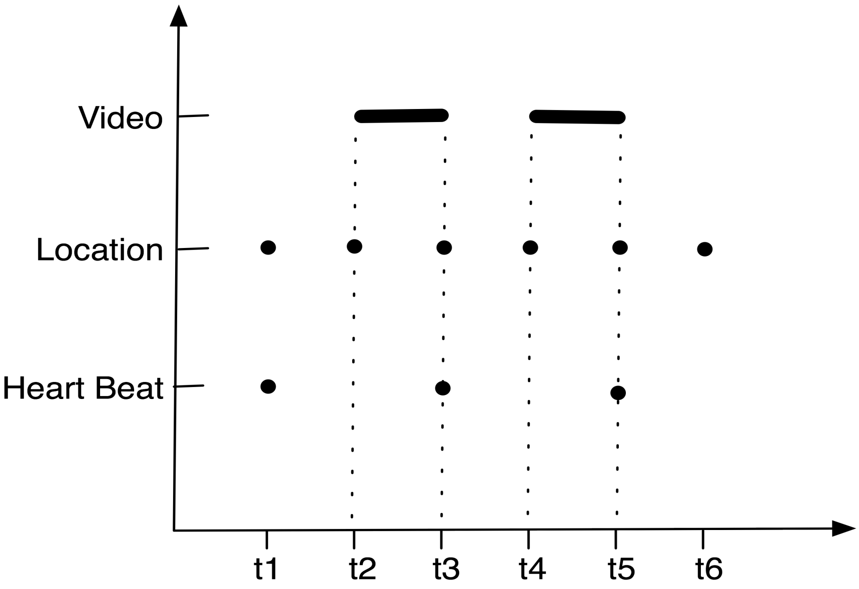


Figure 4.1: Video sensor is sampling in the continuous-time manner and other sensors are sampling in the discrete-time manner

According to Gustaven [28], context-aware data is either external or internal, sensors data is external data and annotation data called an internal data.

Some sensors such as smart objects [130] should be able to record and annotate data based on users' demand or in a more technical sense users' signal. For instance a user is able to record special program on the T.V. and she would like to manually annotates it.

4.3 Data Model

In this framework *Data entity* is the unit of data. Each life event could be a set of data entities. For instance eating a lunch in a resturant is a life event with data entities that represents location, start and end date-time of eating, etc. A data entity is a data object which can be identified by date-time and sensor ID. A data entity has four main components: *annotation*, *social scope*, *timestamp* and *sensor data*.

In order to reduce risks of sharing life-log information, this model provide appropriate access limitation to user's life-log information. It defines privacy and access scope for the smallest information object. Smallest information object here means atomic information block such as an email address, location at the specific timestamp, etc.

An access scope should be defined for each data entity. As previously described, each data entity is a record in a life-log dataset. Atomicity is the main property of a data entity. The dataset is composed from a set of infinite data entities, which come from heterogeneous sensors. The dataset size is infinite because the user's life is ongoing and during his life the dataset size increases. In addition, there are a wide variety of sensors that can be used by the life-log which provide different data types with different structures. As has been noted before this dataset is a data stream. The proposed model should not have any dependency on a sensor's data type or data structure. In other words, there should not be any boundaries between the access scope definition and the sensor structure.

This conceptual definition of the data model enables users to share their life-log information. It is not dependent on technology and a life-log dataset can be in any data format such as XML, JSON [1] or RDBMS [2].

Life-log data are date-time (timestamp) dependent. Therefore, I use timestamp data as a mandatory field for each data entity record. It is notable that in the life-log domain nobody other than the owner of the dataset is able to manipulate the data and the only access right is 'read', nor any other data manipulation rights.

Each life event is a set of data entities or records in the dataset. The life-log dataset of a person P represented via $L(P)$ and $L(P) = \{E_1, E_2, ...E_n\}, n \rightarrow +\infty$. n is going to infinity because the human life is ongoing and since the user is alive the boundary is not specified. The life-log data entity E is a 4-tuple $< D, T, S, An >$, where T is the timestamp, which can be continuous or discrete; if continuous, it has the start timestamp and the end timestamp. D is a 2-tuple $< Do, ID >$, which Do is the data object and ID is the sensor identification. Do can be binary data, e.g. image, audio or textual data, e.g. GPS location, microblog content, etc. An is the annotation which can be stored inline with the data object in each record, or it could be a link to an external data object which is the annotation for this data object (Do) . S is a 2-tuple $< A, Ei >$, where A is the access scope and E is the access expiration timestamp. I defined three access scopes, *private*, *public* and *friend*. Gross et al. [98] also identified these three scopes and named them private, semi-public and open-ended. Private means that the data entity is not shared and nobody other than the owner can access this information object and Ei is null. Public means that everybody in this social domain can access this information object and no access limitation has been defined for it but Ei cannot be null and there an expiration timestamp should be defined for that data object. Expiration time stamp is used to simulate forgetting in a life-log system. Friend defines users who can access this information object. The user can define which user(s) or group of users from the social domain can access this object and Ei must be defined for access expiration date. In this case $A = \{P_1, P_2, ..., P_n\}$, which is a finite set of users or group of users who can access this information object.

A social relation description model such as FOAF [3] can be used to extract the list of users who can see this data entity, or it can be specified manually by the user.

[1] Java Script Object Notation

[2] Relational Database Management Systems

[3] http://www.foaf-project.org

How to extract the user list is not in the scope of this research. Developers or end users can define it based on their interpretations of the system. The public and friend scopes are also inspired by social media such as Facebook or Twitter, but as has been explained before SNSs have only two sharing scopes, either friend or public and there is no private scope.

Please note that only one access right is granted to users which is the 'read' right. Unlike other access control systems I do not have other permissions such as delete, update, etc. It is the authorization process and not the authentication process, because only 'read' can be granted to the user. It is tedious for the user to specify the scope for each data object, therefore in the implementation model he can grant access to a user or a group of users for a group of objects, based on time. For instance all videos from the beginning of March up to the end of April can be marked to be shared within a family scope, which is a group of users.

In order to maintain the granularity of the model, the timestamp cannot be the unique identifier for defining the share scope, it should be used with the data object. For instance consider Figure 4.1, which shows that three sensors have been used. The user intends to share his video between the time t3 and t4, and not his heartbeat rate. Thus a share scope must be defined based on the timestamp and the data object and not the timestamp alone. This assumption leads us to make each data entity record a 4-tuple.

Elements arrival order to the $L(P)$ dataset are based on the timestamp, but there is no guarantee. As an example that shows elements arrival are not based on a specific order, consider a buffering mechanism that can be used for reading a sensor. This buffering mechanism is used to reduce the processing cost. It will gather sensor data and when the buffer size reaches a specified size then the data are sent to the dataset.

This model is a conceptual approach to enable users share their life-log information with society. It is designed to be independent of implementation technologies and it is flexible enough to allow developers to manipulate data entities, but I think these fields are necessary components for each data entity.

In addition to access control, other security aspects of the system must be taken into account during implementation. For instance, communication and data transfer are very sensitive from a security point of view and sharing information with society requires communication. This demonstrates that during the communication, data should be transfer encrypted. More about securing the process of life logging will be described in the “Privacy, Security and Trust” chapter. Following shows examples of data entities in XML format.

Listing 4.1: This example shows a phones’ Bluetooth data which is used as a social proximity sensor. The user shares this information with her family user group and SaraH, with one week expiration timestamp for both.

```
<entity type=’discrete’ stored=’local’ date_time=’Friday␣
   March␣7,2009␣11:33’ id=’mobileBluetooth’>
    <sensor name=’social’ scope=’friend’ class=’mobile’>
      <friendslist>
        <friendID exp_dateTime=’Friday␣March␣14,2009␣11
           :33’>’family’</friend>
        <friendID exp_dateTime=’Friday␣March␣14,2009␣11
           :33’>’SaraH’</friend>
      </friendslist>
      <data> <friendID=’JohnBTMobile’ bondstatus=’10’
      address=’00:1E:45:1B:71:37’/> </data>
      <metadata> <name>’John␣Smith’</name> </metadata>
    </sensor>
</entity>
```

Listing 4.2: A video recording example from a home media server, which has been marked private. The recorded data has not been uploaded on the server and only the link has been sent to the sever.

```
<entity type=’continuous’ stored=’remote’
                start_date_time=’Wednesday␣July␣13,2011␣
                   14:27’
                end_date_time=’Wednesday␣July␣13,2011␣16
                   :00’ id=’homeMedia’>
    <sensor name=’video’ scope=’private’ class=’proximity
       ’>
      <data>
```

```
        <link source='file:///192.168.0.1/records/videos
          /13jul20111427.mpg'>
      <data/>
          </metadata>
    </sensor>
 </entity>
```

Listing 4.3: A weather web service, which has been read from a desktop computer and shared with all users in the target social network. An annotation component annotates it with the city and country name.

```
<entity type='discrete' stored='local' date_time='Friday␣
   March␣7,2009␣11:37' id='weatherWebservice'>
    <sensor name='weather' scope='public' exp_dateTime='
       Friday␣March␣28,2009␣05:00' class='desktop'>
      <data>
        <wheater source='http://foobarweather.com'>
           <temprature value='20' type='C'/>
           <pressure='29.18␣in'/>
           <humidity='59%'/>
         </wheater>
      <data/>
          <metadata><city>'Vienna'</city><country>'
             Austria'</country></metadata>
    </sensor>
 </entity>
```

4.4 Design Challenges and Requirements

The result of life logging is a personal digital memory. Kröner et al. [120] categorized electronic memories into: object memory, community memory, application memory and personal memory. Therefore the following design considerations and theories have been targeted for a personal digital memory only.

Bell et al. [34] introduced some challenges of personal archives which we call life-log dataset in this research. These challenges include controlling the access to the information and personal security, browsing and searching the archive, and the requirement that the archive should support long-term preservation of information. Since life-logs are worth keeping at least until the end of owner's life, log-term preservation is an important requirement, .

Briefly, the design requirements of a life-log system are: *seamless integration into daily life*, *resource efficiency*, *security*, *long-term digital preservation*, and *information retrieval* from the life-log dataset.

On one hand the framework should be an open and extendable platform for life logging with sharing capabilities, and on the other hand life-logs are highly privacy sensitive. To resolve these issues I identified and list the followings approaches and design considerations:

4.4.1 Spatio-Temporal Data Objects

As has been described in the data model, all of our life events except dreams happen in a specific *timestamp* and specific *location.* Based on the currently available technologies it is not always possible to sense the location, because location sensors such as GPS are not functional in every environment. However, most of the operating systems have a timestamp, which is accessible until the device is turned off. This means that timestamp is a necessary field for any life-log record and all digital information objects should be stored with a timestamp. This fact leads to the conclusion that indicates, first the timestamp, and second the location are two links between different information objects, which came from different and heterogeneous sensors.

4.4.2 Controversial History of Life-logs and Security

Since life logging promises the recording every individuals' activity, life-logs are very privacy sensitive tools. They have a controversial history, e.g. the DARPA lifelog project [203] which was canceled in 2004 because of the criticism of the privacy implications of the system. Allen [24] identified two important potential hazards of life-logs: pernicious memory and pernicious surveillance. In the previous section I proposed another classification of the risks of sharing life-logs.

However, life-logs could be a complimentary assistance to our biological memory and they are highly capable of augmenting our life. Therefore, these risks should not hinder technological development and they can be reduced by considering users' security and privacy in the implementation of a life-log system [183]. This framework tries to reduce these issues by securing the sensing and collecting processes, handing the control to users over their information, etc. More about dealing with security and users' privacy will be described in more detail later in the "Privacy, Security and Trust" chapter.

4.4.3 WORN Problem

According to Bell and Gemmell [33] a major challenge of life logging is collecting too much data and never be able to retrieve information from the dataset. This called the "Write Once Read Never" (WORN) problem. Non-digital diary writing is based on users demand and thus there is a filter for users on what they would like to store. This filter facilitates the information retrieval. Sellen [202] suggested: *Rather than try to capture everything, system design should focus on the psychological basis of human memory.*

Moreover reflection methods should consider trimming information for users in a way that facilitates information retrieval such as using visualization. This framework challenges WORN effect by (1)introducing annotation which can be done either automatically or manually and (2)simple visualization methods which assist users in gaining an overview of their logged data. Moreover, a surprise detection and life event detection approach will be introduced that can challenge the WORN effect. These methods will be described at the end of this chapter.

4.4.4 User Intervention

Life-logs operate continuously and passively [94], in other words sensors continuously sense and collect contextual information and a service is always running in the background. Ideally a life-log application runs in the background of a device 24/7, hence user administration or user intervention should be reduced as much as possible. Therefore any interruption and resumption should be addressed [16].

Furthermore users should have appropriate understanding and enough control over the application. While users should be able to configure sensors, but sensing and logging (recording) processes should not be dependent on any user intervention. According to the expectancy theory [230], performing administration tasks has a negative impact on the user experience. The desired life-log system should not require any user intervention or supervision during the sensing and recording phases, which is assumed to improve usability. Only administration tasks, such as configuring sensors or managing temporary disk spaces, could require user intervention. Users should have sufficient control on managing their life-log tool, but it should not interrupt them and should be unobtrusive.

In essence, there is no restriction on capturing data only automatically, some sensors could also capture data manually and ask for user input or start capturing data based on user demand. For instance a daily self survey could be performed a couple of times per day and it could asks about the users' mood. This can be assumed to be a sensor which collects data about users' status by manual user input.

4.4.5 Long-Term Digital Preservation

Digital preservation is maintaining information, in a correct and independently understandable form, over the long term [59]. Likewise it can be interpreted as method to preserve the content in an understandable form in the long term, therefore digital preservation prevents data format obsolescence. As has been described, life-log information is worth keeping during the owner's life.

A digital preservation process composed of different processes such as migration, evaluation, emulation and etc. [234]. Migration is a process of changing the format of a digital object to a another format which is long term archiveable [15]. Migration eliminates the need to retain the original application for opening binary files. Bell [32] named long term preservable data formats “Golden” data formats. For instance, TIFF is a golden data format for long term preservation of image files. Text files such as XML are golden data format. Life-log files are either in text format or binary format. Binary files format needs to be checked and if it is not long term preservable it should be changed to a long term preservable format.

Bell et al. [34] noted that this problem is one of the major challenges of using life-log tools. There are two challenges with long term archival of digital data, hardware and software. The hardware problem is not in the scope of this research but following solution used to tackle the software in this framework:

In order to consider a long-term preservation of the digital objects for pervasive devices. The file conversion can be handed to an external Web-Service, because file conversion is a resource intensive process. Then binary objects which are not long-term preservable will be converted to a preservable format. Therefore users do not need to have any interaction with the system and file conversion can be done in the background. How this framework will handle digital preservation will be explained in more detail later in this chapter.

4.4.6 Performance and Storage

Battery, storage and network bandwidth are resources that are used during the process of life logging. Life-logs are always running and collecting contextual data, but pervasive devices which are being used for collecting contextual data are suffering from resource weaknesses e.g. battery weakness [194], and they are not reliable storages that can host life-log data. Therefore there should be a reliable storages media (RSM) that host life-log data for the long term. Here RSM is the server, which guarantees reliable data storage. Later in this chapter I will prove that the implementation of the Data Reader (UbiqLog) is resource efficient and how I send Data Reader data to an RSM.

Another major consideration is the location of information storage. This is due to the fact that some life-log sensors produce large sections of binary data such as video and audio and uploading these data objects to the RSM costs network bandwidth. Here I propose uploading links to the server and large data remain on the local storage. For instance consider a home media server which records all T.V. programs that users watch. It is not feasible to upload them all to a remote server, but they can be recorded on the users' local storage and a link to retrieve them within their metadata can be uploaded to the server. This removes the burden of large data upload to the server which costs significant network bandwidth. However there should be a mechanism to frequently check that data objects are available in the correct path in their local storage.

4.5 System Architecture

This section focuses on describing the technical architecture of the framework in terms of software engineering. The previous section described the design considerations which constitute the foundation of this framework. Now I describe the software architecture of the system. The framework architecture will be explained later in more detail. Afterwards two implementations (applications) of the system will be described as a proof of concept. One application has been done for mobile sensors and the other application has done for proximity sensors.

The framework architecture [176] is similar to the well known client-server architecture, but here I use the term *Data Readers* instead of clients. Data Readers reside near sensors and they are responsible for collecting sensors' information and sending it to the server. *Server* is a reliable storage media and hosts heterogenous information which comes from different sensors. It receives sensor data from Data

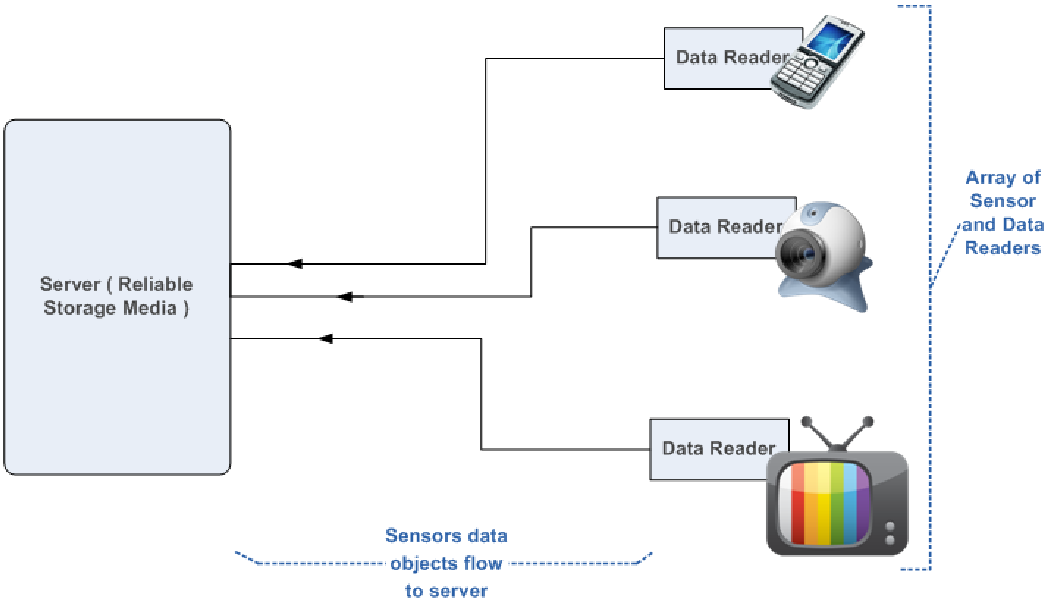

Data Reader
Data Reader
Data Reader
Server (Reliable Storage Media)
Array of Sensor and Data Readers
Sensors data objects flow to server

Figure 4.2: Abstract overview of the framework

Readers and it could be hosted in a single machine or in a cloud.

Both client and server design has been inspired from blackboard architecture [237]. This means that multiple component are running independently from each other and thus failure of a single component should not harm other components. For instance if the annotation component is not working it does not affect the Long-term digital preservation component.

Figure 4.2 shows an abstract overview of the framework and the information flow. In simple terms, Data Readers reside near sensors and collect their information, then they send them to the server which hosts data. The server is a reliable storage media for maintaining data in the long-term and provides required features for life-logs (e.g. annotation).

4.5.1 Server

The Server functions as the main repository for life-log information. The Server is used for two main purposes: 1) A reliable storage media which hosts life-log information 2) Processes which are not possible or hard to perform on Data Readers will be done on the Server. For instance a format conversion of a binary document such as video is a resource intensive process and Data Readers are not capable of hosting such a process, therefore the server will perform these processes.

The Server could be hosted on a single machine with no Internet connection, but if the sharing features of the framework is intended to be used, which means users should be connected, then the sever should be accessible via the Internet.

As it has been shown in Figure 4.3 the server is composed of eight main components: *Data Repository*, *Long-Term Preservation*, *Metadata extraction and Annotation*, *Social Network*, *Network*, *Query Interface*, *Data Management* and the *Settings and Configurations* component. Figure 4.4 shows the data object process flow when the server acquires the object from “Data Readers” and writes it to the “Data Repository”.

Settings and Configurations stores information about users settings and some general information about the framework such as external service information, etc.

Life-log data will be hosted in the Data Repository. Data Readers are temporary data repositories and the main repository for data is the Server. Furthermore data

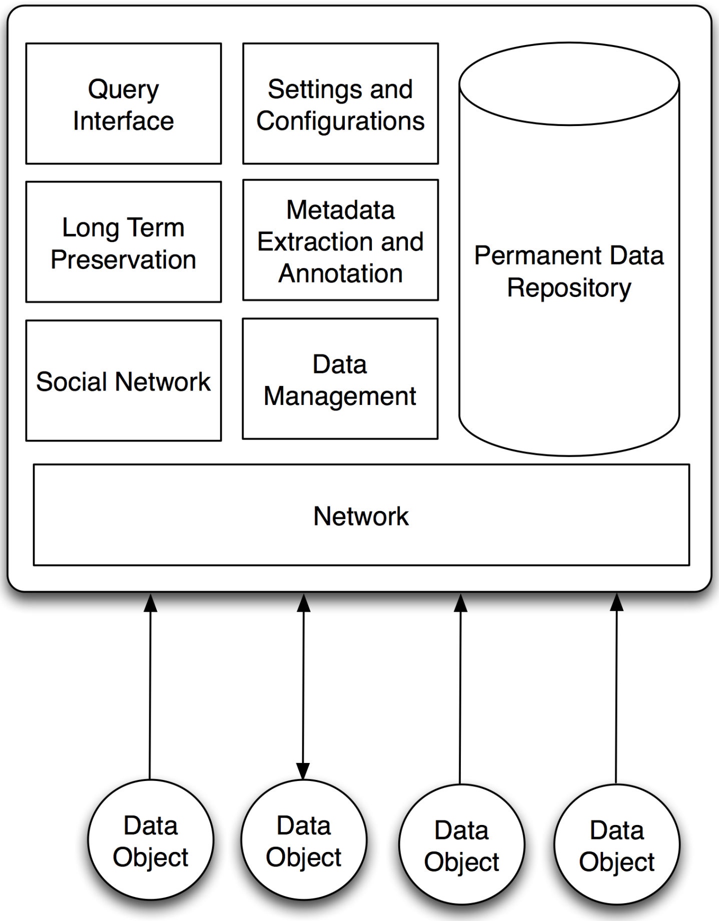


Figure 4.3: Server Architecture

will be stored on the server to support social networking features and therefore a central node should be accessible to users.

Because of the flexibility, performance [213], portability and long-term digital preservation, RDBMS has not been used and all life-log information will be stored on the server in plain JSON files. JSON has been chosen because it supports structural data elements and it consumes less disk-space in comparison to XML which is similar to JSON from a features perspective.

The implementation instances create a JSON file every day which contains users' activities (sensor information) with timestamp. Information about binary data will

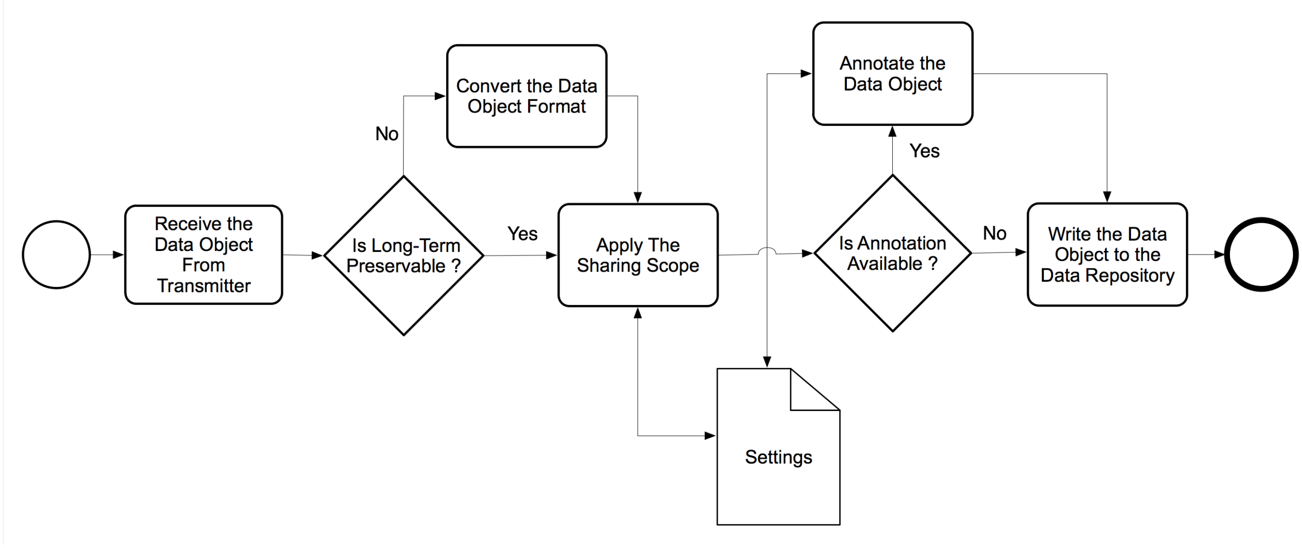

Receive the Data Object From Transmitter
Is Long-Term Preservable ?
No
Yes
Convert the Data Object Format
Apply The Sharing Scope
Settings
Annotate the Data Object
Yes
Is Annotation Available ?
No
Write the Data Object to the Data Repository

Figure 4.4: Data object process flow on the server

also be stored in the JSON file. Information objects which will be stored in the JSON file about the binary data are: meta-data and annotation, timestamp ,local path of the binary data, a flag that indicates that the binary data should be uploaded to the Server or it should be maintained locally.

The Long-term Preservation component is responsible for converting data objects' data format in a preservable form and evaluate the result of conversion. Most digital Preservation related tasks such as format conversion will be done on the server side because these tasks are resource intensive and pervasive devices which are used as Data Readers are not capable hosting large processes.

The Data Management component has two responsibilities: First it maintains and administers the life-log dataset on the server, second it hosts and updates links for remote storages. As has been explained before it is not feasible to upload some large information objects on a server. Since the life-log data is always created during the users' life they consume large amount of space. For instance Gordon Bell's life-log which includes pictures, letters, financial statements, purchase orders, manuals, etc. consumes about sixteen gigabytes from 1998-2001 [33]. The data objects which are not stored directly on the server could be large or they could be privacy sensitive. Therefore it might be better to keep them local and private and not upload them on the server. Data management checks the availability of data objects in the remote storage and if they are not available or the link is broken, it notifies users. The idea of storing links to remote data inspired from the term “Data Space” [85]. Franklin et al. introduced this term, which provides central supports and query interface for distributed data. Here the “Link” has been used in order to support distributed storage. Metadata and the links to original data objects will be uploaded to the server instead of the whole data object. A Link can be a simple file system path, e.g. `file:///C:/MyHomeMedia/2009-10-11/watched/DigitalLife.avi`.

In addition to these features the Data Management component administers life-log dataset, which includes archiving old data, creating index for data objects based on users demand to retrieve them, etc. An index optimizes the query performance and assists users in better information retrieval.

Metadata Extraction and Annotation component is responsible for annotating incoming sensor information. Simple annotations could be done on the Data Reader side and more complicated annotation could be done on the server side. This component performs the annotation which will be done on the server side. Usually this type of annotation requires more resources and thus Data Readers are not capable of performing them. As has been stated before here annotation can be done manually or automatically. For instance I have implemented an automatic annotation approach which gets information from users' google calendar and annotates dataset records based on the timestamp with Google calendar events.

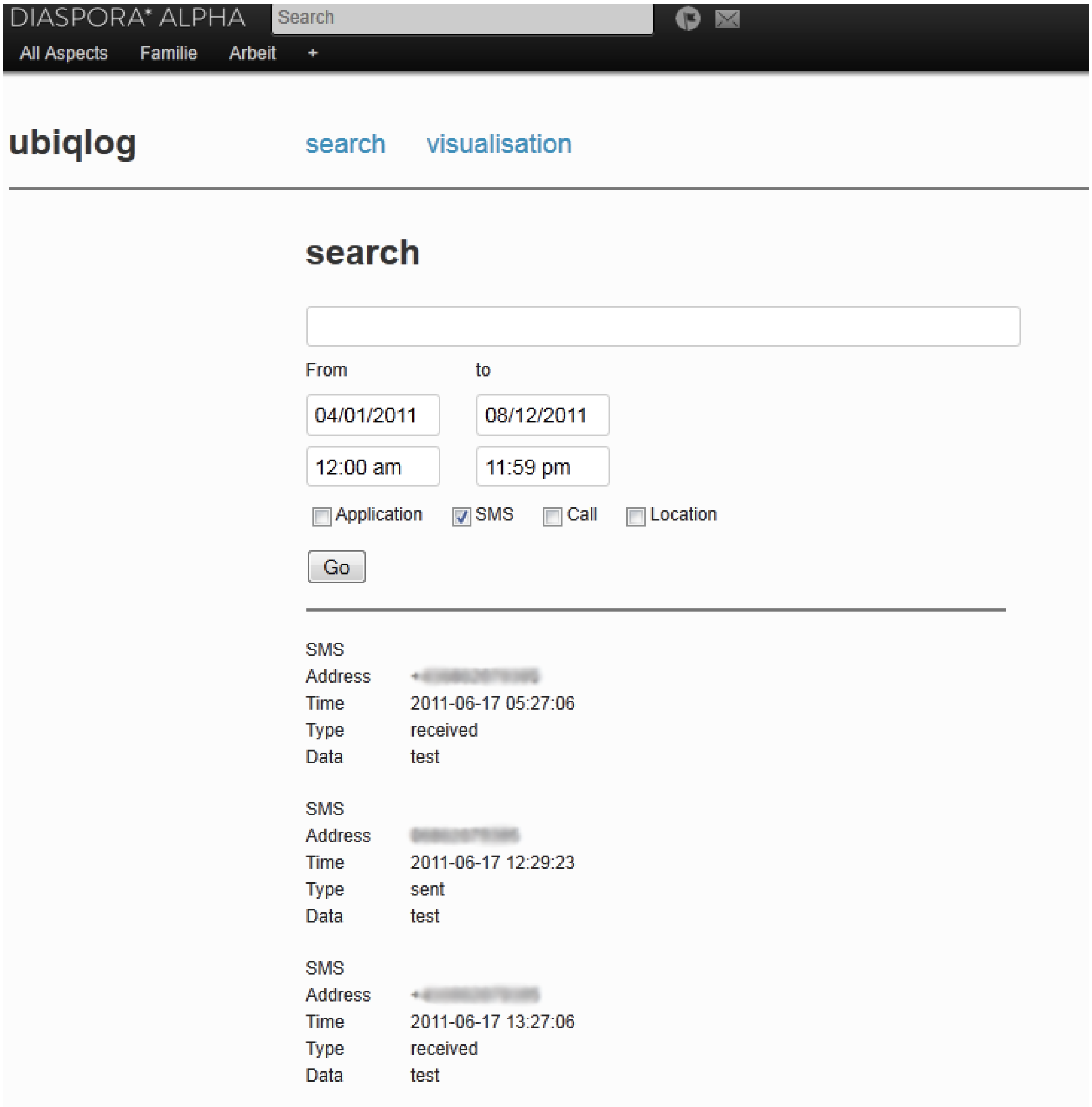
Query Interface reads users input through the GUI and assists users' to browse or perform queries against their life-log dataset as has been shown in Figure 4.5. The query interface has been implemented with Apache SOLR [4]. SOLR is a open source text search platform from the Apache Lucent Project.

The Network acts as a network layer and it is responsible to receive data object from Data Readers or sends information (e.g. annotation) to Data Readers. The transmission of data objects to and from the server with Data Readers will be done via HTTP connection.

The Social network component is used to enable users to share their life-log information with others. The social network layer is using Diaspora [5], which is an open

[4] http://lucene.apache.org/solr

[5] https://joindiaspora.com



search

From 04/01/2011 12:00 am

to 08/12/2011 11:59 pm

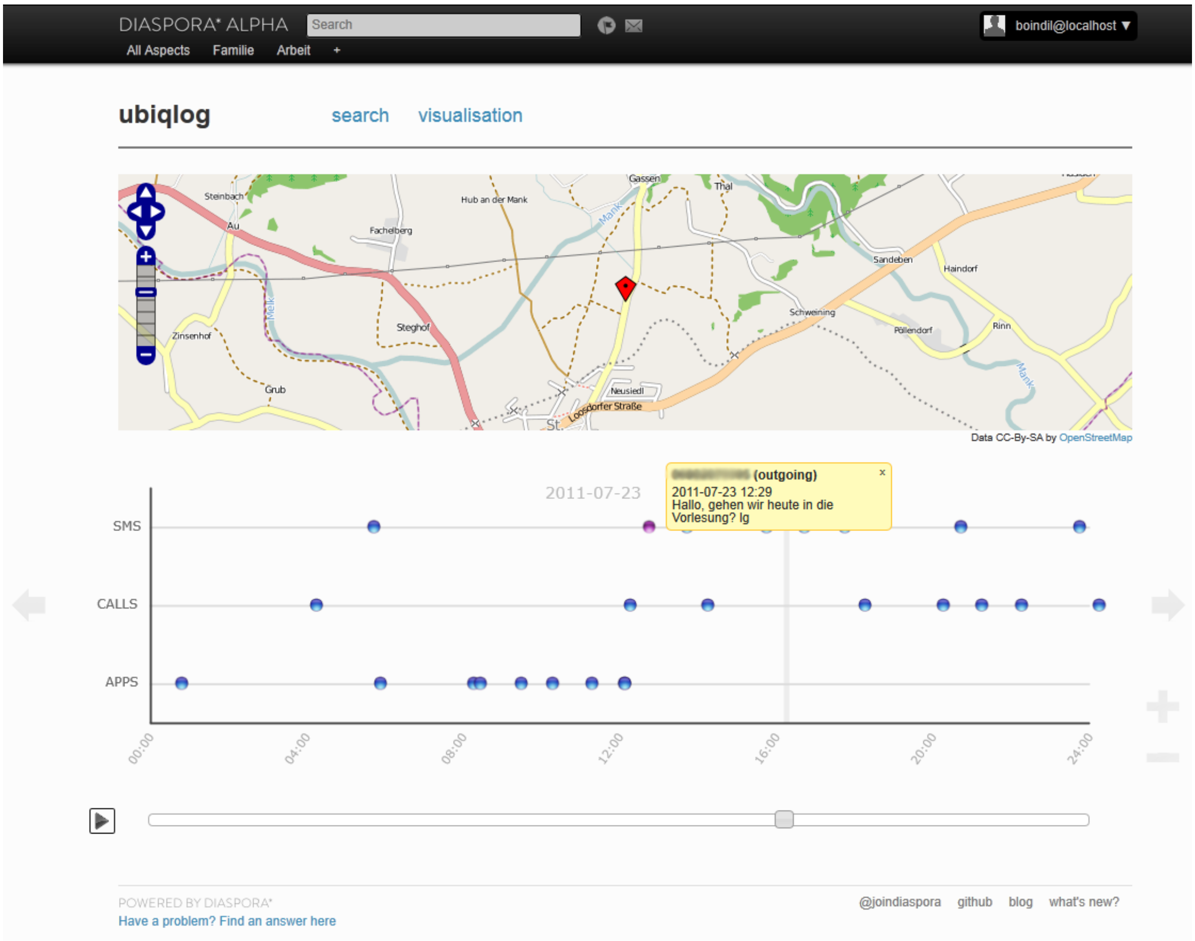
□ Application ☑ SMS □ Call □ Location

```
SMS
Address   [redacted]
Time      2011-06-17 05:27:06
Type      received
Data      test

SMS
Address   [redacted]
Time      2011-06-17 12:29:23
Type      sent
Data      test

SMS
Address   [redacted]
Time      2011-06-17 13:27:06
Type      received
Data      test
```

Figure 4.5: Query interface on the server side

source social networking tool developed with Ruby and its development is still in progress. The social networking features of Diaspora have been employed to provide usual SNS features, but layers and components, which can handle and manage life-log information, have been added to it. Figure 4.5 depicts how users access and browse their uploaded life-log information from their Data Readers. Then users can select data objects, which they intend to see more detail about, by clicking on the blue dot (Figure 4.6). They can also share with other users in their social network. When users intend to share they should chose an expiration timestamp for any shared object, otherwise default expiration timestamp which is ten days after they share will be chosen by the system. Furthermore in the implemented prototype users are enable to specify which information objects could be shared. For instance a user can choose to share her SMS with another user (in friend list of the social network) for a week.

The sharing style in the implementation is based on *push*, which means users chose what to share. However as a future work sharing with *pull* will be supported. In this context *pull* means that other users can ask a user to share and she can decide to share with them or not.



[redacted] (outgoing)
2011-07-23 12:29
Hallo, gehen wir heute in die Vorlesung? lg

Figure 4.6: Data visualization component which has been integrated in Diaspora and runs as the Social Network component of the Server.

4.5.2 Data Reader

As has been stated before the term client is not applicable here, because the main function of this component is to collect raw sensor data and send it to a reliable storage media, which is the server in this framework. Usually one Data Reader will be used to read and collect multiple sensors' data. A Data Reader and sensors could be hosted on the same device such as a smart phone, which is a Data Reader and hosts many sensors. It could be a hardware device or it could be an application which is running on a device.

Life-log systems have many similarities to context-aware systems, but it is notable that a life-log tool is different from a context-aware tool in several aspects e.g. life-log tools should archive data for long-term access and consider the annotation of datasets. Context-aware systems are typically not designed to maintain information for long-term access, annotation and privacy are less important concerns for them and unlike life-log systems, do not need to always run in the background of the device eliminating the issue of resource usage.

In order to achieve a versatile and flexible architecture on the Data Reader side, a layered architecture model [231] has been used. Components were designed to be abstract and thus it is possible to add new components or remove an existing component. One of the main design requirements is to create a flexible and extendable system, which enables users to integrate new sensors or configure current sensors. This sensor extendibility feature is called versatility in context-aware approaches [139]. Figure 4.7 shows the architecture of a generic Data Reader.

Application Log is being used to log errors and messages of the application in order to assist developers in debugging. *Extension Interface* is used as an interface for adding new sensors to the framework. An external sensor might be an application or it might be a hardware component such as a device with a Bluetooth connectivity feature. How to add a new sensor to the framework will be described in more detail in the implementation section.

Similar to other context-aware approaches the Data Reader contains two major operations: sensing and recording. Chen and Kotz suggested [53] to decouple the context sensing part from other parts of the application in order to maintain a flexible environment for sensors configuration. The Data Reader has been designed

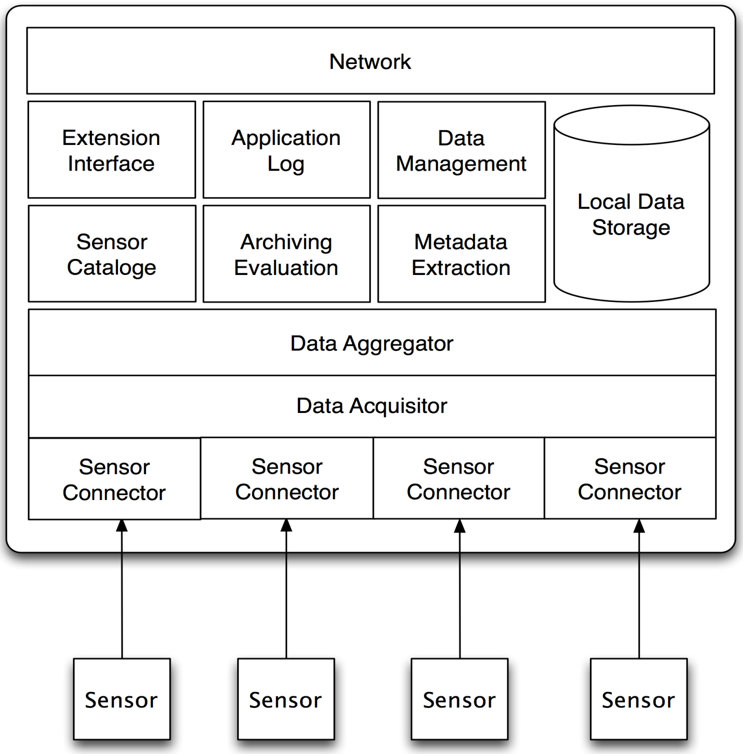


Figure 4.7: Data Reader Architecture.

based on their suggestion.

Mostly context sensing tools are composed of two types of hardware: a data acquisition device and sensors. Sensors are responsible for sensing the environment and reading the contextual data in a raw format. The data format varies based on the sensor. The raw data of each sensor should be aggregated in a format that is appropriate to gather all readings in one dataset. Therefore the data acquisition device is required, which is responsible to gather and aggregate raw data from the sensors. Aggregation here means converting raw data into a data format, which the Data Reader is able to parse. Sensing and recording are assumed to be two different phases of the life-logging data collection process. Following is the detail explanation about these two processes.

Sensing

Context data acquisition methods predefine the architectural style of the system. There are different approaches to read contextual data, in this framework sensor data will be read directly via a sensor connector and no middleware will be used.
The sensing process is composed of three component: *Sensors*, *Sensor Connectors* and *Data Acquisitor*. "Sensors" are used to sense and read contextual information. They can reside in the Data Reader as an application or hardware component. They can also reside physically outside the Data Reader, such as an external GPS device connected to the smart phone (Data Reader) via Bluetooth, or a Web service, which provides temperature based on current location of the user.
The "Sensor Connector" establishes a connection between the Data Reader and the sensor. The Connection depends on the sensor, e.g. a sensor can send data via a Web service or a sensor can write data in the file only. A security component might also be used by the sensor connector, especially when the sensor is physically located outside the Data Reader. A network connection will be established between the sensor and the Data Reader, hence an authentication or an authorization process might be necessary. In simple terms, the sensor connector encapsulates the sensor structure and reads data from the sensor. I suggest having a sensor connector for each sensor, as a sensor failure would otherwise affect other functionalities of the framework.
Information about sensors, their configuration and their connection status will be kept in the *Sensor Catalogue*. The sensor connector connects to the sensor and reads information based on the configurations, which reside in the sensor catalogue. Figure 4.8 shows the sensing process sequence diagram of this framework. The "Data Acquisitor" is a stream, which acts as a temporary queue to hold the read data from sensors. It contains raw sensor data and is located between the data aggregator and sensor connectors. It encapsulates data that has been read from sensors in order to make it available for the data aggregator. This layer exists because it is not possible to send raw data from the sensor connector to the data aggregator directly, while there is no guarantee for reading sensor data synchronously in the real time, therefore this component is required. For instance, an authentication process might take time, which means a data from that sensor cannot be read in real time. In

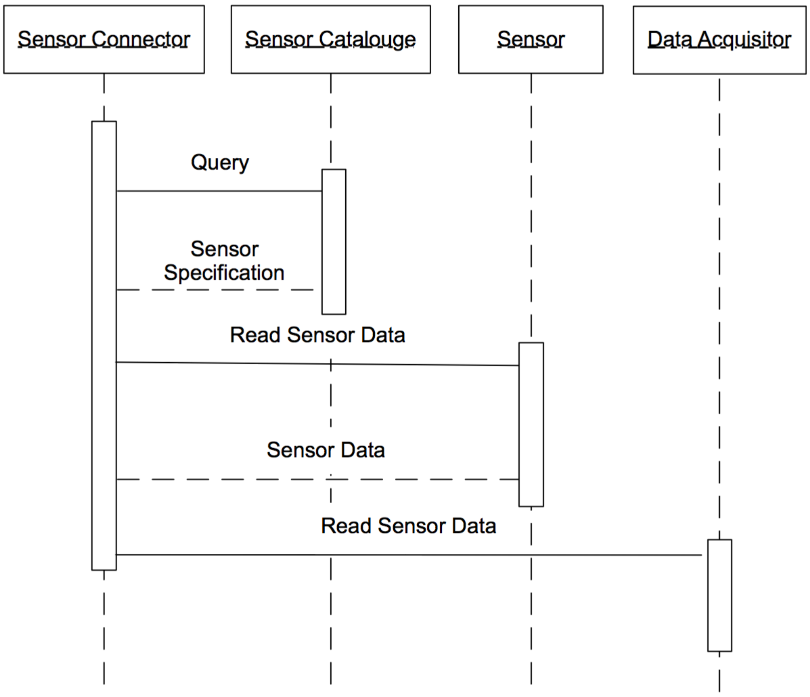


Figure 4.8: Sensing process sequence diagram.

addition data aggregation processing cannot be done in real time. Thus, the data acquisitor will be used as a temporary sensor data holder (data buffer). This leads us to conclude that the acquired data from sensors will be logged passively and not actively.

It is notable that Data Reader architecture does not comply with any sensor classification and it can support any of three aforementioned sensors classes, unlike other context-aware efforts. For instance, Raento et al. [173] has classified smart phone's sensors as location, user interaction, communication behavior and physical environment sensors. Froehlich et al. [88] categorizes phone's sensor as hardware sensors and software sensors such as application usage. Schmidt et al. classified sensors as logical and physical [198].

Sensors data should be read in parallel, which means when a sensor is not available or is not working it should not raise any errors and it should not affect other reading processes. Sensors and the Data Reader are connected mostly in a unidirectional way. If an authentication or authorization mechanism is required connection can be bidirectional between the mobile phone and the sensor.

Refining and Recording Data

The refining and recording part consists of five components: *Data Aggregator*, *Metadata Extraction*, *Data Management*, *Local Data Storage* and *Network Transmitter*. This phase considers scalability, manageability and reliability of a life-log information.

Raw data from different sensors needs to comply with a consistent data format. Furthermore, the data needs to get annotated in order to facilitate browsing, search and further data retrieval. The "Data Aggregator" receives raw data from "Data Acquisitor" and transforms data to a consistent structure, which is readable for the framework. In particular, Data Aggregator is a middle layer, which enriches sensor data via annotation and convert its format. Most other context-aware approaches [28] use XML for their data format. In the implementation of this framework I convert data into the JSON[6] format, because a JSON document consumes less disk space in comparison to similar XML documents while being as flexible as XML.

The following shows two examples of the aggregated data, one from the SMS sensor and the other one from the Application sensor:

```
"APPLICATION":
{"ProcessName":"com.android.browser",
 "Time":"Oct 15, 2009 6:21:40 AM"}
```

[6] http://www.json.org

```
"SMS":
{"Address":"4444444", "Type":"send",
 "Time":"Dec 24, 2009 11:23:01 PM",
 "Body":"test message" }
```

These JSON data entities comply with the described data model, because each entity has a date-time, a data object and an annotation. Binary data has the same structure but instead of text they hold the location of that binary data entity. For instance a picture data entity is as follows:

```
"CAMERA":{
"picturename":"2009-12-29 20.42.05.jpg",
"localPath":"/sdcard/dcim", "storage":"server",
"Time":"Dec 29, 2009 8:42:07 PM"}
```

Converting a raw data to the desired data format, will use the information from the sensor catalogue, e.g. find and use the related annotation class. Metadata Extraction component in the proposed architecture (figure 4.7) will be used to annotate data in order to make them searchable and browsable. Annotation here is adding a textual data to the JSON record. For instance an annotated data entity from the call sensor might be as follows:

```
"CALL":{"Number":"11111111",
"Duration":"13", "Time":"Dec 24, 2009 9:23:04 PM",
"Type":"outgoing",
"metadata":{"name":"John Smith"}} }
```

The “metadata” element of the above JSON record represents the annotation. The Data Aggregator uses the “Metadata Extraction” component to annotate data of the sensors. The annotation process is not only restricted to be done during the data aggregation phase, it can also be done during the data acquisition phase.

After the data getting annotated and converted to the data entity format, the resulting information object will be stored locally on the data reader (First data will be stored locally, then users can transfer them to a reliable storage). These information objects will be written to the “Local Data Storage” by the Data Aggregator. The write process will be done passively and not actively or real time, because raw data must be converted to a readable data format. Furthermore, the data aggregation process cannot be done in a realtime e.g. an annotation could be done via an external web-service call, which costs time. The reason that first data is stored locally is the network connection, which is not always available and previous research efforts [55] show that transferring data automatically to anther storage, increases the risk of packet loss and requires data compression. Pervasive device storage has limited capacity, therefore a feature has been provided which allow users to manually upload the data to a reliable storage media (RSM) or even automatically. The RSM is the server, which is capable of hosting life-log information. Uploading data to the RSM will be done by the "Network Transmitter" layer.

The life-log dataset consists of files, which are either binary files (pictures, videos, calls, etc) or text files. Manual upload to the RSM requires user intervention, which is not desirable. Therefore, "Data Management" will be used. Data Manager is responsible to check if the specified maximum size is reached or not, if it reaches the maximum size a secure network connection will be established by the "Network" and it uploads data-set files to the RSM. After a successful file receive, RSM acknowledges the framework. If files are successfully uploaded to the RSM then Data Manager removes files from the local storage. Moreover, Data Manager is responsible to compress the content of text files. For instance, Data Manager can check the log file and remove redundant records or compressing files if there is lack of disk space on the local storage.

4.6 Proof of Concept and Applications

The framework architecture and data model have been explained previously. In order to evaluate the proposed data model and the architecture I describe two implementations in this section. Each of these implementations focuses on a sensor class, one implementation (UbiqLog) is for the mobile sensors and one is for the proximity sensors. I do not provide any implementation for desktop sensors, because there are many comprehensive implementations available. UbiqLog is a lightweight, configurable and extendable life logging framework that uses mobile phones as a device for life logging. Another implementation proves proximity sensors and it is a metaphor that motivates users considering their target environment tidiness status.
These two implementations is a proof that this framework is *holistic* and *multipurpose*. This is due to the fact that they have been designed for two different use cases and they have a very similar architecture.
At the end I will explain two methods that assist users in extracting knowledge from the dataset, one is surprise detection the other is life routine detection.

4.6.1 UbiqLog

Smart phones are conquering the mobile phone market; they are not just phones they also act as media players, gaming consoles, personal calendars, storage, etc. They are portable computers with fewer computing capabilities than personal computers. However unlike personal computers users can carry their smartphone with them at all times. The ubiquity of mobile phones and their computing capabilities provide an opportunity of using them as a life logging device. The proposed application is the implementation of a Data Reader which extends previous research in this field and investigated mobile phones as life-log tool through continuous sensing. Its openness in terms of sensor configuration allows developers to create flexible, multipurpose life-log tools. In addition to that this mobile based Data Reader can be used as reference model for further life-log development, including its extension to other devices, such as ebook readers, T.V.s, etc. In the past most life-log tools were custom-built devices or applications using external hardware as sensors (e.g. [192], [55]). With the advent of smart phones, the mobile phone has become a feature-rich mobile device, creating an opportunity to use it as platform for life logging. It is not easy to convince users to carry something other than their usual digital devices such as mobile phones or watches to perform a novel activity which in this case is recording their daily activities. Smart phones are small computers with computing capabilities, but more importantly they are equipped with sensors and networking technologies, such as Bluetooth, which can sense and communicate with the user's environment. Their networking capabilities enable these devices to be connected to a larger computing or storage media, becoming a hub for personal or body area networks [190]. Further, smart phones can provide useful information about our daily activities such as received and dialed calls, text messages, application usage, phone camera pictures, etc. A disadvantage of dedicated life-log devices is that, users need to carry yet another device with them just for the purpose of recording their daily activities. Based on smart phone capabilities and features we believe that they represent a promising platform for life logging. Therefore the focus of this study was to create a data reader, which resides in the user's smart phone and does not require any additional hardware (except that hardware is a new external sensor). However smart phones are not designed to be used as life-log tools, leading to a number of challenges when using them for this purpose. For example the performance of the phone could be affected by the amount of resources required by a life-log application running on the phone.

Since UbiqLog is the implementation of a Data Reader the focus is on the collection stage of life-logging and therefore this research does not delve deep into reflection methods. Although, for the purpose of demonstration, we [185] provide visualization features, which will be examined in the evaluation section. Additionally, the prototype features a search facility to enable users manage their life-log dataset and remove undesired information from their dataset directly on their mobile device. This is an important requirement due to the fact that users are the owners of their life-log dataset and thus they should have full control on their personal information [180].

The proposed application allows users to configure the sensors they want to use for recording their life-logs by disabling or enabling a sensor, or changing a sensor settings. Additionally users can add a new sensor to the application or remove an existing sensor from the application. Flexibility to this extent requires a specific data model and architecture, which we propose as a part of this research. Based on the proposed data model and architecture, we implemented a prototype of a life-log tool running on the Android 2.x platform for Android phones. The prototype is flexible enough to be used for other platforms just by changing sensors according to the target platform. So far Android has been used for mobile phones and tablet computers, but it could be used on other things such as T.V.s [7], and vehicles. Although the implementation have been done on the Android platform, the architecture and data-model is not Android dependent and can be used on other platforms as well.

In simple word this work provides a novel approach to life logging by enabling users to configure sesnors and add new sensors. Other efforts, which employ mobile phones for life logging, provide a closed architecture, making it difficult or impossible to customize the life-log tool. Related projects that use an open architecture such as MyExperience [88] are not designed for life logging. Their focus is on other research aspects such as context sensing, and to my knowledge there is no other open architecture for a life-log framework available.

Since life-log systems are similar to context-aware systems, it is notable that a life-log tool is different from a context-aware tool in several aspects. For example, life-log tools should archive data for long-term access, consider the annotation of datasets, and so on. Context-aware systems are typically not designed to maintain information for long-term access, making annotation and privacy are less important concerns in most cases. Moreover context-aware systems, unlike life-log systems, do not need to always run in the background of the device eliminating the issue of resource usage.

As related work for UbiqLog I can refer to Context-Aware approaches which has been described in the “Related Work” chapter.

[7] http://www.google.com/tv

Design Principals

Mobile phones (or in general pervasive devices) by their very nature are not designed to host large computing applications. There are some restrictions such as resource limitation, user interface, etc. in comparison to personal computers. Life-log tools are very privacy sensitive and this research proposes to bring this tool on mobile phones, which are prone to loss or damage [194]. It is therefore important to take some design principals into account when implementing a life-log application for mobile phones. Most of design principals have been previously explained in the "Design Challenges and Considerations" section of this chapter such as storage, performance, etc. However followings are design principals which are not described yet and they belongs to the specific type of implementation for a mobile based Data Reader or life-log in this framework. I did not list them in previous section because they are UbiqLog or in a more technical sense implementation specific.

- *Sensor Extendability*: A major shortcoming of smart phone sensors are their limited precision and weakness compared to external sensors. For instance an external GPS device is more powerful than the phone's built-in GPS or the quality of a photo from a digital camera is better than the quality of a photo taken with the phone's camera. On the other hand, mobile phones are increasingly equipped with more capabilities and features. It can be anticipated that in the future, new sensors will get integrated into mobile phones. Therefore, it is a crucial requirement to provide an open interface that allows adding new sensors. In order to deal with this problem we propose an integration interface to enable developers integrate external sensors into UbiqLog. For instance, an external GPS can be connected to the phone via bluetooth.

- *Support for Multiple Use-cases*: As described earlier the information recorded by life-logs can be used in many different domains. The focus is to provide a generic data reader, which is capable to be configured based on the users' requirements. Users can add/remove (extendability) or disable/enable (flexibility) sensors and change settings of available sensors. This capability makes the framework flexible enough to be used for different purposes. For instance, once it can be used as health monitoring device and in another use case it can be used to study a group behavior such as employees' geographical location changes in a firm. Li et al. [133] stated that an individual's life has different facets and available systems are mostly uni-faceted. However, I claim that UbiqLog is a multi-faceted system, because it is capable to be used for different use-cases.

Data Enrichment Features

In addition to the sensor extendibility and openness, UbiqLog supports annotation and checking for long-term digital preservation in order to enrich the raw data that has been collected from sensors.

Annotation: A possible approach for assigning semantics and enriching life-log datasets based on described requirements is the use of annotation. Here by annotation we mean tagging and thus metadata creation which is data about the data. Metadata information objects will be extracted from other resources and they will be collected based on date-time and (if possible) location.

We embed some annotations within the data entities in the "metadata" tag, as shown in the above examples. Some annotations will be stored separately from the life-log dataset, but those files have similar structure. Annotations that are not embedded in the data entities, will be done by calling external web services. Those services collect textual annotation from other information resources e.g. personal calendar and online news media. In the implementation, we use a service to call Google calendar and it collects calendar events in a text file. Although this feature is available, we suggest to perform this type of annotation on Server and not on the Data Reader, because establishing a connection to an external service consumes bandwidth.

Moreover, to consider the readability and flexibility of the framework we suggest to perform the annotation during the data aggregation phase and not during the data acquisition phase.

Digital Preservation: Tomitsch and I [184, 179] proposed a framework to consider the long-term preservation of the digital objects for pervasive devices. There we suggested to handle the file conversion to an external Web-service, because file conversion is a resource intensive process. We converted binary objects which are not long-term preservable to a preservable format. Therefore users do not need to have any interaction with the system and file conversion is being done in the background.

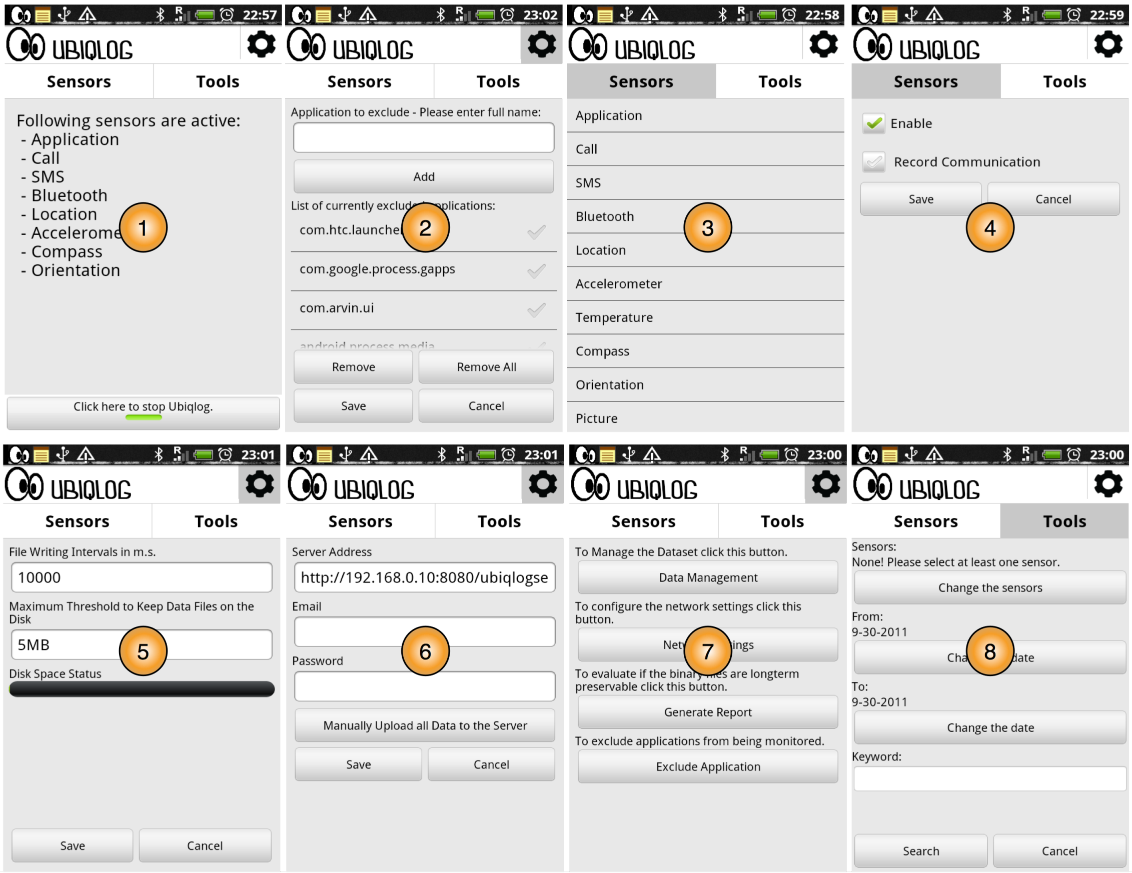
In the UbiqLog framework we intend not to bound ourselves to any external tool such as Web services, which require external service call. Thus users can choose how to handle their binary objects. Either they can use the described method or a second method, which is lighter. The second method is a simple evaluation mechanism based on the data format. If the file format is not preservable (not in a golden format category) an entry for the metadata information of that file will be created. All binary files in the framework have a record in a textual file, which contains information about the files. This textual file will be sent to the RSM with the associated binary files. File format conversion can then be handled on the RSM side and the results can be automatically evaluated by another component. In simple terms, in the second method only the file format will be checked on the mobile device. If it is not long term preservable an entry will be written in the text file. We suggest to use the second method, because it consumes less resources such as bandwidth and CPU and there is no guarantee that external services are always available for the file format conversion.

Implementation and Evaluation

The UbiqLog has been implemneted on the Android 2.x platforms. The purpose of this implementation was to evaluate and improve the feasibility of the Data Reader. Then as a longitudinal evaluation a nine month usage of this tool on user's mobile phone will be analyzed. Since the life-log tool needs to be constantly running in the background of the device, it is important to evaluate the battery efficiency during use the implementation. Finally, a usability evaluation of the application based on Nielsen's heuristics [158] approaches will be described.

We have developed an implementation of the proposed architecture on the Android 2.0, 2.1, 2.2 and 2.3 platforms. The implemented application was used and tested on a Motorola Milestone (Droid), HTC Legend, Samsung Nexus S, Samsung Galaxy S, ZTE Blade, Samsung Galaxy Mini, HTC Desire and HTC Desire HD. This application fully complies with the described architecture.

Figure 4.9 shows the GUI of the application. A major consideration during the



Following sensors are active:
- Application
- Call
- SMS
- Bluetooth
- Location
- Accelerome
- Compass
- Orientation

Click here to stop Ubiqlog.

Application to exclude - Please enter full name:

List of currently exclud plications:

com.htc.launche
com.google.process.gapps
com.arvin.ui
android.process.media

☑ Enable

☐ Record Communication

File Writing Intervals in m.s.: 10000

Maximum Threshold to Keep Data Files on the Disk: 5MB

Disk Space Status

Server Address: http://192.168.0.10:8080/ubiqlogse

Email

Password

To Manage the Dataset click this button.

To configure the network settings click this button.

To evaluate if the binary files are longterm preservable click this button.

To exclude applications from being monitored.

Sensors:
None! Please select at least one sensor.

From:
9-30-2011

To:
9-30-2011

Keyword:

Figure 4.9: GUI screenshots. Frame 1 shows the first screen that appears when the user opens the application. The feature shown in Frame 2 enables the user to exclude applications that she do not intend to log. Frame 3 shows the list of default sensors, which were implemented in the application. Clicking on one of the sensors from this list takes the user to the configuration screen, shown in Frame 4. Frame 5 shows the data management setting screen. Frame 6 shows the screen for configuring network settings. Frame 7 shows the setting options. Frame 8 shows the search facility.

GUI design was to provide sufficient functionality to enable users to easily configure sensors without any knowledge about the lower layer of the application. When users click on any sensor in the sensor list, as shown in Figure 4.9, Frame 3, they will be shown the configuration screen for each sensor such as Frame 4.

The user navigation flows between the different screens of the application is shown in Figure 4.10. Each box represents a screen in the application. Each Sensor Reader has been implemented as a separate Android Service. Since Android services run in the background, no GUI is required to run these services. For each sensor there should be an associated sensor reader class. In order to add a new sensor into the application a developer must implement the "SensorConnector" java interface in the "com.ubiqlog.sensor" package. If developers intend to add the associated annotation class they should create a class in the "com.ubiqlog.annotation" package and this class has to implement the "Annotation" java interface. Currently, in the implemented application we annotate incoming and outgoing calls and text messages (SMS) with the recipient's or sender's names which are read from the contact list. After the developer creates the annotation class they need to rebuild the framework and installs the new apk file on the target device. This apk file contains newly added sensor. There is no possibility to add new sensors dynamically or during the run time to the framework, which means endusers can not do them. They can only configure existing sensors and enable/disable them.

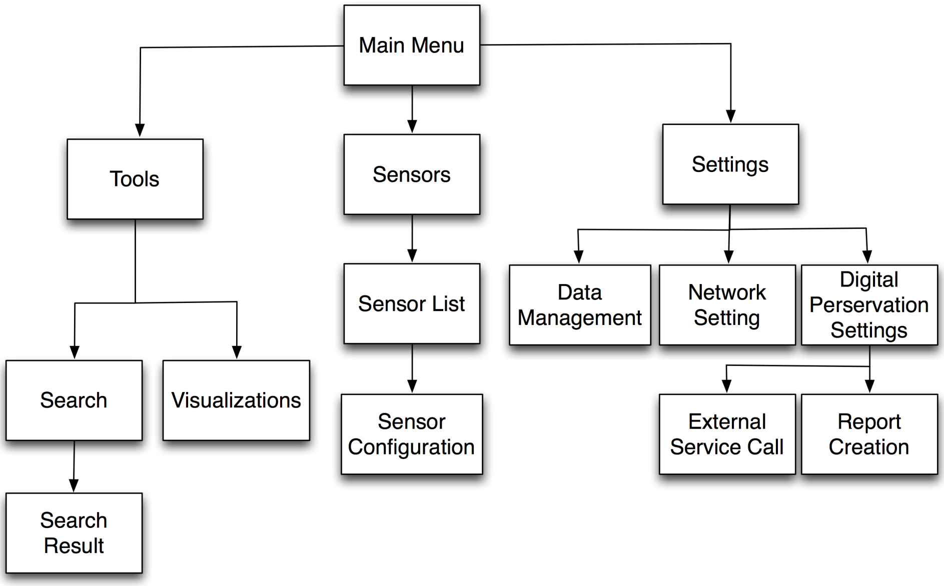


Figure 4.10: GUI screens navigation flowchart.

Android provides listener classes for some sensors, when the sensor senses a new

Sensor	Configuration Values
Application	Enable = yes, Record Interval in m.s. = 10000
Telephone	Enable = yes, Record Communication = no
SMS	Enable = yes
Location	Enable = yes, Update Rate in m.s. = 10000
Accelerometer	Enable = no, Rate = Delay Game, Force Threshold = 900
Picture	Enable = no, Capture interval in m.s.= 1000
Temperature	Enable = no, Measurement Unit = Celsius
Compass	Enable = no
Bluetooth	Enable = yes, Scan Interval in m.s. = 60000
Orientation	Enable = no

Table 4.1: Default sensor configurations

data, the listener will be notified e.g. calls information can be read via using "PhoneStateListener" Listener. We use these listeners to log sensors' data, but for some sensors such as application usage there is no such a listener. Those sensors will be checked frequently based on a specified time interval via their associated sensor readers.

As stated before the Sensor Catalogue contains information about sensors. A table in the SQLLite database of the Android platform has been used for the Sensor Catalogue implementation. This table contains sensor names, sensor reader class, sensor annotation class and a list of configuration data for that sensor.

Table 4.1 shows default installed sensors and their configuration values in the implemented application of the UbiqLog framework on the Android smart phones. Users can list sensors from the application GUI as shown in Figure 4.9 Frame 3. the Accelerometer and picture sensors are disabled by default, because they consume high amount of resources and thus drains the battery. Other sensors are disabled, based on the result of our user survey, in which users stated which sensors have higher priority (enabled sensors).

In order to implement the Data Acquisitor, which is a temporary data buffer, a static Java Collection object (ArrayList) was used. The sensor connectors write their data directly to this object.

Data Aggregator runs as an Android service, it reads data from the Data Acquisitor and it performs the annotation.

Life-log data is stored as a set of text files on the SD card of the phone in a folder called “ubiqlog”, binary files such as photos can be stored in same folder but in a subfolder called “binary”. Location and metadata about each binary file is stored in a the same file, which will be created for every day the application is being used. Users can use the “Search” menu and access their life-log data. They can also manipulate the search result (modify or remove an entire record).

Life-log files will be uploaded to the RSM by the HttpClient package of the Android. Password and server address are required, as shown in Figure 4.9 Frame 6, to allow the application connecting to the Server (RSM) and uploading files. On the RSM side we have designed a simple Java Servlet which reads files and stores them locally on the user’s personal computer. After a successful file receive, the RSM acknowledges the application in the Http response (HTTP Status Code is 200) that the files have been successfully uploaded to the RSM. Afterwardss the Data Manager remove the files from the SD card of the device. Users can specify the maximum size for the Ubiqlog folder as shown in Figure 4.9 Frame 5. The Data Manager can check if the threshold has been exceeded or not and if the threshold has been exceeded, it will upload life-log files automatically to the RSM. The Data Manager has been implemented as an Android service, which continuously runs in the background, with the sensor readers and other services.

“Archiving Evaluation” is another Android service, which checks whether binary files are in a long-term archive-able format. As described above this service can be used in two ways, either to call an external Web service to convert file format or mark the binary files that they are not in the long-term archive-able format by adding a corresponding meta-data for this file in the associated record.

In order to check the long-term archive-ablitiy of life-log dataset binary files, users can manually generate the report by using the “Generate Report” button on the “Setting” Frame as shown in Figure 4.9 Frame 7. Calling an external Web service to change the format of the file is another method which we described it in another research [184].

Longitudinal Evalutation

UbiqLog was used by six users over a period of one to fourteen months. Three users immediately installed the application after they purchased the new phone. This means that they used the application from the first day of using their phone. Every day a text file was created containing sensor information. It contains location of the binary objects and their meta-data. Figure 4.12 shows the size of log files for about three months. Those logs are generated by using a Motorola Milestone phone. The reason why the file size was larger in the beginning of the study period was since the user described that he is keen in using a new phone and discovering the device features or surfing the market for new applications. This behavior is repeated by two other users with new phones. Therefore we conclude this is the reason of having a large log files in the first days of usage. Besides when users have lots of location changes file sizes increases. Figure 4.12 shows two days in this period with zero file size. On those days UbiqLog was not running, therefore no file was created. The visualization shown in Figure 4.11 demonstrates how the data recorded with the UbiqLog tool can be used to visualize the user's social interaction based on sending and receiving calls. This visualization approach was inspired by Song et al. [211]. Individuals' names have been pseudonymized in respect to the users' privacy, therefore phone numbers are shown with Person and a number. The longer the call duration, the closer the person is to the center of the circle.

Resource Usage Evaluation

Jain [110] described that the application performance is composed of five factors: usability, throughput, resource usage, response time and reliability. Evaluating the performance of an application is necessary for evaluating the implementation of the framework. Since the UbiqLog application will always run in the background (and therefore always consume resources). As mentioned above resource consumption is one of the major challenges of mobile computing [193], which is especially relevant for life-log application, since they typically need to constantly run in the background. Thus the performance of the application can influence the general functionality of the device, which is not designed for this type of application usage. Therefore re-

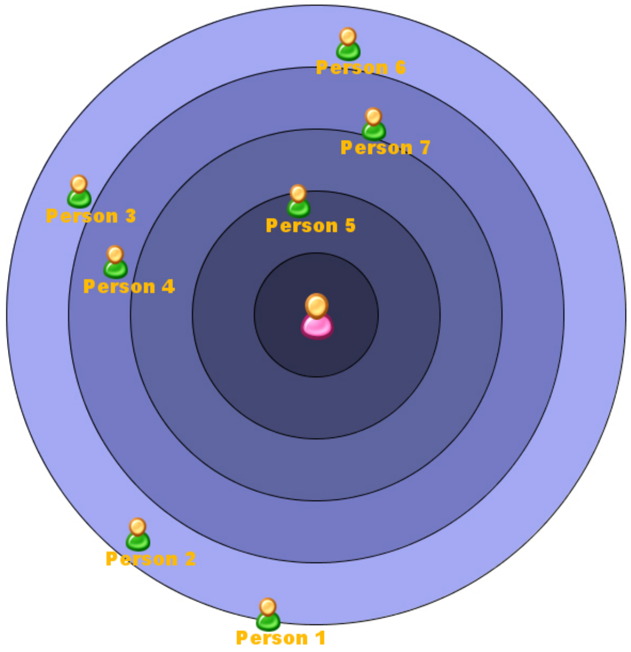


Figure 4.11: Social Interactions of the user based on the duration of received and dialed calls.

source usage monitoring is an important factor that developers need to consider while designing such applications.

The application was tested directly on the device at the end of the implementation cycle to eliminate any data redundancy and sensor reader malfunction.

In appendix I have described that mobile resources are: battery usage, CPU usage, memory usage, disk activity and network activity [175]. To measure mobile phone resource utilization I used a resource monitoring tool [178] which will be described in “Appendix A” in more detail. I specifically measured battery, CPU and memory usage of our application. Network activity and disk I/O of the life-log were not measured, since the application did not have heavy network or disk activity. Disk write happens infrequently because we buffer data and then write them in the file.

When all sensors were active, result of the CPU utilization monitoring, showed that our application consumes less than 3% of CPU in average, when administration activities are performed and the GUI is active, it consumed about 10% of CPU in average. Average VmRSS (Virtual Memory Resident Set Size) was 15728 Kb

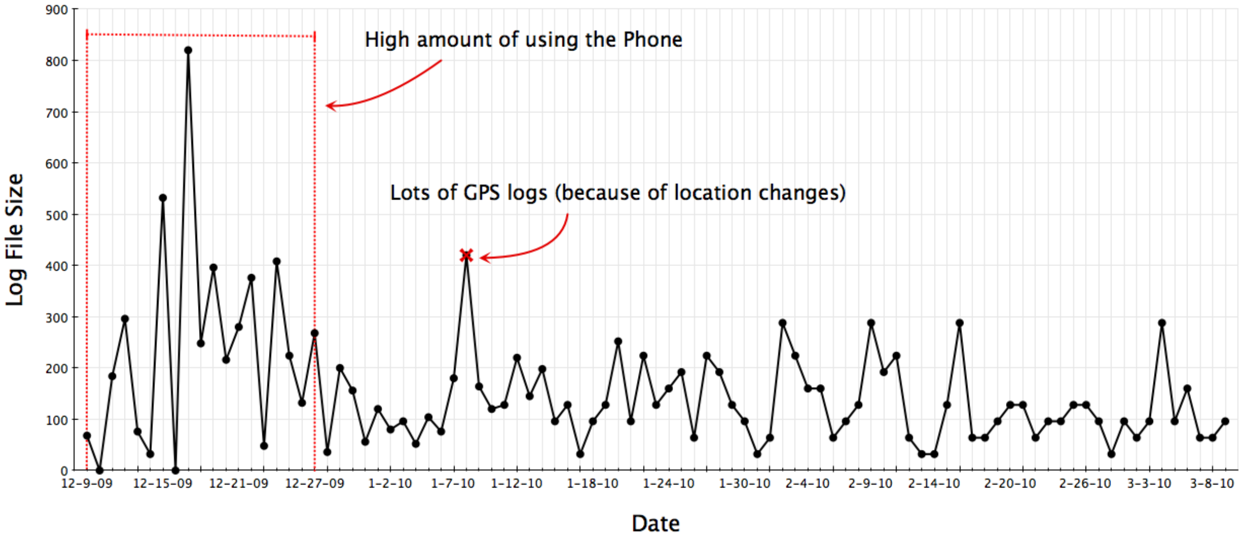


Figure 4.12: File sizes of the life-log dataset. The unsteady part represents that user is playing with his new phone features(the area is marked with red). Other days when the file size is big mostly is because of using GPS outdoor and many location log entries have been generated.

and VMSize is 127268 Kb. It is notable that disabling some sensors (i.e. reducing the number of sensors) did not affect the CPU or memory utilization. This result shows that the implementation of the UbiqLog consumes fair amount of the CPU and the memory. Battery utilization cannot be measured per process. The GPS sensor reads the user's current location and the Bluetooth sensor scans the environment for discovered devices, based on a configurable time interval. The default Bluetooth scan interval is six minutes and the default GPS location read interval is ten seconds. Both Bluetooth and GPS are highly battery consuming, but they produce important information for the life-log dataset. Therefore we needed to investigate whether the implementation of the framework had any significant impact on battery consumption. In order to perform this study, we investigated battery utilization under different conditions, where different sensors were activated. Each test was started at a set time in the morning with a fully charged battery. Tests were run until the battery level reached a level of the 20%. Device usage condition have been kept constant during the entire test period. This means that the device was still used to answer phone calls or SMS, but not for any unusual purposes, such

Battery Discharge Duration	UbiqLog Application	WiFi	GPS	Bluetooth
21:00 Hours	deactive	on	off	off
20:00 Hours	active	on	off	off
7:30 Hours	active	on	on	off
6:30 Hours	active	on	on	on
7:00 Hours	deactive	on	on	on
14:30 Hours	active	on	off	on

Table 4.2: Approximate time it took for the phone to reach a battery level of 20% (starting from 100%) in different scenarios. When WiFi and Bluetooth are on, they were idle and not active.

as playing games. However there is no guarantee that the device usage is exactly the same for all tests, because the user can not control incoming calls or etc.

Table 4.2 shows the approximate time it took for the phone to reach a battery level of 20% in different scenarios. It is notable that Wi-Fi, which consumes high amount of battery, was always on in all scenarios. Furthermore to preserve more battery network connection of the phone was set to 2G (not 3G).

The intention of this study was not to evaluate battery usage of different hardware settings. Instead I intend to prove that the implementation of the proposed UbiqLog framework does not have a large impact on battery utilization. As shown in Table 4.2 there is about one hour difference between using the application and not using it while Bluetooth and GPS are disabled. With Bluetooth and GPS both enabled the application decreased the battery time for half an hour. This might be due to I/O operation increase (writing GPS and Bluetooth log on SD Card)

Usability Evaluation

This framework and its implemented prototype targets life-log system developers, but it can benefit end users by providing self-insight. In order to enable end users benefit from the framework we propose following visualizations Figure 4.13 which assist users in better self-awareness and self monitoring about their device usages. In order to evaluate the usability of the framework implementation we have employed Nielsen's usability heuristics [158]. The implementation that users evaluate contains those visualizations under the "visualization" option of the "tools". The usability of the implementation has been evaluated by six users (2 female, 4 male) between 25

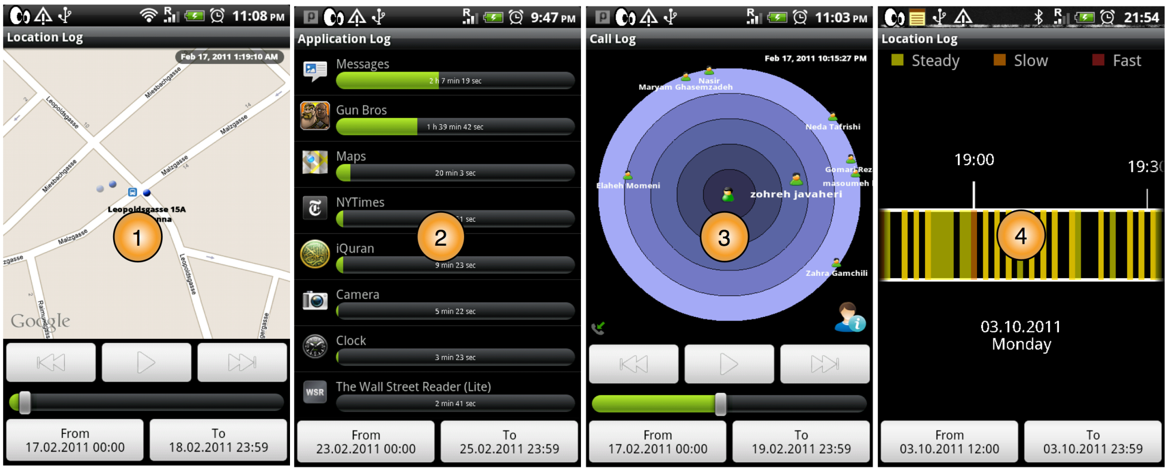


Figure 4.13: (1)Location, (2)Application usage, (3)Call visualizations and (4)Movement

and 37 years of age. Four of the participants stated having strong computer knowledge; and other two stated having basic computer knowledge. All participants use computers and mobile phones in their daily life. Participants owned HTC Desire, Motorola Droid (MileStone), Samsung Galaxy S, Samsung Galaxy Mini, HTC Legend and HTC Desire HD. We installed the application on the participants' phones and asked them to use the application for a period of four weeks to one year. After this period we conducted interviews and asked them to fill out a survey. The survey included Nielsen's principles for user interface design [158]. We adapted those principles in form of questions and asked participants to rank the application accordingly. The result of the evaluation shows *Help and Documentation* received the lowest score (2 from 5), but other heuristic factors received satisfactory scores (4 or 5 from 5).

API Extension

UbiqLog is a platform which enables users to sense and collect mobile phone data and contextual information. On one hand this information can benefit users in many aspects such as behavior learning [152], studying community influence on a behavior [168], etc. On the other hand these facilities can assist developers and researchers to use this platform for further development e.g. pervasive games. Since we have argued this tool is flexible, extendable and it can be used for different purposes, an API has been provided. The API enables developers and researchers to use the UbiqLog as a sensing and collecting tool, without delving deep into the UbiqLog architecture. Table 4.3 shows UbiqLog functions and their associated classes. The *Engine* class is used to start or stop UbiqLog and searching the content of log files. The *Sensor* class is used to activated or deactivate and configure the sensors. *DataManager* represents the "Data Management" component of the Data Reader. *Network* represents the "Network" component and *Annotation* represent "Metadata Extraction" component of the Data Reader architecture.

Class Name	Method

`Engine`	`void startRecording()`
	`void stopRecording()`
	`String[] searchContent(String text)`
	`String[] searchContent(String text, String[] senorNames)`
	`String[] searchContent(String text, String[] senorNames, String starDate, String endDate)`

`Sensor`	`boolean initSensor(String sensorName,boolean status, string params)`
	`boolean isSensorActive(String sensorName)`
	`void activateSensor(String sensorName, string params)`
	`void deActivateSensor(String sensorName)`
	`void setSensorParams(String SensorName, String params)`
	`void setSensorParams(String SensorName, String[] params)`
	`String getSensorParams(String sensorName)`
	`String[] getSensorParamsArray(String sensorName)`

`DataManager`	`boolean removeFiles(String path)`
	`void setThreshold(int size, string unit)`
	`void archiveCheck(String reportLocation, String dataPath)`
	`void excludeApp(String appName)`

`Annotation`	`void annotate(String sensorName, String metadata, boolean inline, String timestamp)`
	`void annotate(String sensorName, String metadata, boolean inline)`

continued on next page

continued from previous page

	void annotate(String metadata, boolean inline, String startTimestamp, String endTimestamp)
Network	boolean auth(String username, String passWord, URL networkAddress) void upload(URL networkAddress, String username)

Table 4.3: UbiqLog classes and their associated methods

4.6.2 A Metaphor for Measuring Room Tidiness

Nowadays, we benefit from using computers in many ways such as information processing, entertainment, social communication, information retrieval, etc. Behavioral psychology (behaviorism) describes behavior as anything a human can do such as feeling, acting, etc. The tidiness of an environment can provide important information about individuals who are involved in the target environment, such as their behavior, health and mental condition.

In order to prove the definition and usage of “Proximity sensor”, we provide a metaphor [181] which uses a wall mounted camera (proximity sensor) to measure a tidiness status of a target object and persuading users toward changing their behavior and being obliged to tidiness. The data from the metaphor feed to the server and treated as a sensor for the life-log dataset.

According to the behaviorism philosopher, Skinner [207] individuals learn behavior by reinforcement and punishment. The process of getting reinforcement, and producing a new behavior by reinforcement, is called “Operant Conditioning”. In simple terms, operant conditioning forms an association between the behavior and the consequences of this behavior. Operant conditioning constitutes the foundation of this work. Here an application will be employed to continuously record a personal behavior and control consequences of this behavior by encouraging users to change their behavior and become more obliged to tidiness.

Unlike children, mature individuals are clean and tidy their indoor environmental objects such as desks or rooms as a personal preference and no or little external input can affect their behavior. However, tidying an indoor environment has high potential for procrastination. Self-monitoring which causes self-awareness reduces procrastination [162]. Moreover, Fogg [82] described that self-monitoring is a type of persuasion because it can help users to become self aware, which assists them in behavior modification. He described three things that can prevent a behavior to be changed [81]: lack of motivation, lack of ability and lack of a well-timed trigger to perform a behavior.

Here we assume that the target users are individuals who are motivated towards tidiness. They should be physically and mentally able to change their indoor environment or there should be an ability to perform this task, but they need extra motivation and willingness to change this behavior. Lack of motivation and lack of a well-trigger to perform the behavior are assumed to be reasons that prevent individuals from considering their tidiness status or in particular changing this behavior. Fogg proposed a terminology [80] that indicates that persuasive technologies and methods can employ computers to assist individuals in altering their behaviors. He described the use of computers as persuasive media having many advantages over other media such as interactivity, being more persistent than human beings in interacting with humans, etc.

As has been stated before we employ a still camera to capture pictures from the target environment daily and continuously. The camera is supposed to capture pictures from the indoor environments of users and not the outdoor environment. Then a method that can measure the target environment tidiness will be described. The measurement process is based on comparing the recent picture with a picture that has been taken when the target environment was ideally ordered.

Persuasive based Related Work

Related works can be analyzed from two perspectives. First approaches that record an activity or a set of users' activities in the long-term will be described. These systems give feedback to users based on their past activities. In a more technical sense, they are persuasive approaches which record information continuously, and the persuasion process is done based on past records. Second, approaches which have been used for image comparison will be described.

- Persuasive Behavior Modification:
  ViTo [157], UbiFit [58] are ubiquitous based approach, which uses the mobile phone and PDAs to record users' activity as has been explained in the "Related Work" chapter.
  show-me metaphor [113] employs a simple ambient display to show how much water users use when they take a shower. The ambient display shows in the form of LEDs on a stick. This display is used to encourage users to use less water.

Nakajima et al. [156] propose two persuasive approaches, “Viral Aquarium” and “Mona Lisa Bookshelf”. Viral Aquarium is composed of an ambient display and toothbrushes with an accelerometer sensor. It is designed to motivate users to consider their toothbrushing pattern. If users brush their teeth according to a specific pattern, fishes in the aquarium multiply otherwise they die. Mona Lisa Bookshelf uses a still camera, video camera, distance sensor and a flat display to monitor the book shelf and record changes on the book shelf. It tries to encourage users to keep books on the bookshelf organized and return missing books.

- Vision-Based Change Detection:
  Different type of applications are relying on image change detections, especially remote sensing and video surveillance [172]. Our approach is very simple, because the background of images do not change and due to a fix camera position no geometric adjustment is needed. Here we refer to research which uses either a video camera or a still camera to record and extract information from image series or videos.
  VIOLAS [107] proposed vision-based sensing for object location in sentient buildings. It employs different types of cameras to extract contextual information from objects in a target environment. It provides a model to support facility management and indoor-environmental controls.
  Kidsrooms [39] is an interactive play space which focuses on kids entertainment in a room. A room similar to a child’s room was built in the Lab and developers proposed some interaction scenarios to study construction of complex spaces. This project employed video cameras for tracking children and detecting their motions.

Design Principles

As a principle of design, the sensing and persuasion process should easily integrate into individuals’ daily life. Besides, ethical issues must be taken into account which will be described in the next chapter. The following are other design considerations which constitute the foundation of this work.

- *Low cognitive load:* Clutter of an object can be perceived immediately by users. Unlike other measurements which are based on persuasive approaches, here there is no need to provide visualization. Similar ubiquitous approaches [113, 156, 215] employed ambient displays to provide perceivable information to users. These types of information presentation are used when the information is not easily perceivable by the user and it contains a high cognitive load. Our approach has a low cognitive load because the target object status is easily perceivable by a psychologically healthy user. In particular, the target environment tidiness can map to a numerical range, therefore there is no need to reduce the cognitive load.

- *Sharing Influence:* According to Zimbardo's *Social Influence* theory [242], sharing information (in our case tidiness status) will have better effects on motivating individuals than not sharing. Besides, Zajonc proposed *Social Facility* theory [241], which describes that individuals perform better when other individuals observe them or participate in performing the same task. These theories lead us to conclude that sharing the output of the system in a social community motivates users to alter their behavior more. In the implementation of the prototype, users are able to send their target object tidiness status to their social network (The Server, Twitter or Facebook). We performed a small survey to establish if sharing this information in a social community increases users' motivation to consider their tidiness more or not. This will be described more in the "Evaluation" section.

- *User Intervention:* A persuasive approach which requires self-survey or any form of user intervention burdens the user unless it encourages the user to perform the required task. According to the *Valence Instrumentality Expectancy* theory [230], humans by nature are not keen to perform tasks which do not entertain or engage them. With this system we tried to prevent such a burden. Therefore we need a mechanism to continuously notify users. This mechanism should require no or little user intervention. In the implemented prototype there is no need for user intervention, after the initial setup of the system.

- *Notification Policy:* As has been described, indoor tidiness status is something that can be easily identified by the user and unlike electricity or water consumption it does not require any additional effort to make the target information perceivable. Here the intention is to notify users about something of which they are already aware. Hence there is no need to give additional information to users. According to the *Operant Conditioning* theory, individuals change their behavior based on reinforcement and punishment. Here, as a personal motivator, we use an interrupting mechanism, which can be interpreted as a punishment. In the implemented prototype if users' target environment is untidy they will get an email on a daily basis, which notifies them to order their target environment. An additional email in the inbox or an SMS on the mobile phone might cause a little discomfort to users. Since this level of discomfort is not coercive and causes persuasion, it is appropriate motivation for users to consider their target environment tidiness. In addition, based on the operant conditioning theory, using a stimulus repeatedly might cause satiation. Satiation means users have less motivation to change their behavior and the former motivation approach does not encourage them to change their behavior. Hence we believe a light form of obtrusion such as sending an email is an appropriate form of obtrusion and prevents satiation. In other words, creating a strong emotional response or implying force might cause psychological reactance and prevent users to change their behavior [42]. However our approach is a small trigger (stimulus) to assist users in altering their behavior.

- *Output Channels:* To our knowledge, this information object is not listed as privacy-sensitive information [210]. However in order to abide by ethical issues and respect users' privacy, sharing this information in a social community is not enabled by default and during the prototype installation users can decide to use which output channel they would like e.g. email, Facebook, etc. Moreover studies [114, 218] revealed that users are skeptical to use new tools or systems for their home computing. Therefore, no new communication channel will be introduced in this research and we will use users' available communication channels. Nonetheless the architecture is flexible enough to accept new output channels or configure existing output channels. Output channels could be

Date	Tidiness Status
12-5-2010	Perfect
13-5-2010	Perfect
14-5-2010	Good
15-5-2010	Good
16-5-2010	Good
17-5-2010	Good
18-5-2010	Good
19-5-2010	Bad
20-5-2010	Bad
21-5-2010	Bad
22-5-2010	Jungle
23-5-2010	Perfect

Table 4.4: Tidiness status of the target environment

software applications or external hardware devices.

- *Maintaining Data:* Captured images are stored in temporary on the file system and will be uploaded to the server. Keeping a history about the target environments status might be more persuasive to users and provide them with better self-insight. For instance, in the implemented prototype an email will be sent to the user which contains a report as shown in Table 4.4. This table shows the tidiness status of a user's target environment during this time.

Implementation and Evaluation

Pictures will be taken based on specific time intervals (the default is one picture per day). In order to establish the tidiness status of the target environment images will be analyzed, and if the target environment is messy, the system will encourage users to tidy it. The process of encouragement will be done by sending a message to users' predefined output channels.

This system tries to reflect the long-term consequences of the behavior, therefore feedback is created based on the channel type and usage. For instance an email will be sent once a day, and a Facebook status, which contains the tidiness status, will be created once every three days. We hypothesize that this frequency of status update on Facebook, motivates friends to comment on users' status. However these are default system settings and users can configure feedback creation frequency as well as output channels.

Figure 4.14 shows the conceptual architecture of the system. The architecture is inspired by the Data Reader architecture, but we have changed the names to enable users using this metaphor independent from the life-log framework. The "Change Detection" component is used to detect changes based on the target environment (desk or room). Settings and configuration of the system such as location of images, etc. will be kept in "Application Configuration". "Application Configuration" has the role of both "Sensor Catalouge" and "Data Management". "Tidiness Calculation" is responsible for converting the result of image comparison to a human readable number (it could be interpreted as "Meta-Data Extraction" in the Data Reader). The "Feedback Generator" component is used to create a message and the "Output Channel Manager" streams the message to the related output channels or sends the data to the server. Output Channel Manager could be mapped to the "Data Transmitter" of the Data Reader.

In the implemented prototype the camera position is supposed to be fixed. This means that the camera should be mounted at a fixed location, such as a wall, to take a picture of the target environment. Therefore we do not need to perform the image registration, which means no geometric adjustments are required. Image registration is the process of aligning several images into the same coordinate frame [172]. Additionally, our method handles intensity adjustment, but we suggest to users to capture images with a fixed amount of light (e.g. in the night when the light comes from lamps).

The ideal picture is a one that has been captured from the target environment when it is tidied perfectly. Users should manually select the ideal picture. We suggest tiding the target environment as much as possible and then to start using the system. All other pictures will be compared to the ideal picture and the number of changes indicates the clutter status of the target environment.

This metaphor could be interpreted as a sentient artefact [114], because it is an actuator for tidiness and it provides information about the state of the target environment. It provides information about a part of users' personality which is based on the clutter status of their target environment. Furthermore, these pictures are a valuable information source for a life-log dataset, especially when they are annotated and combined with other information. The result of the combination might enable researchers to extract novel information about users' life styles. To implement the prototype, we employ the A4Tech and LogiTech C260 webcams. The Image processing and change analysis is done by MATLAB Image Processing Toolbox [8].

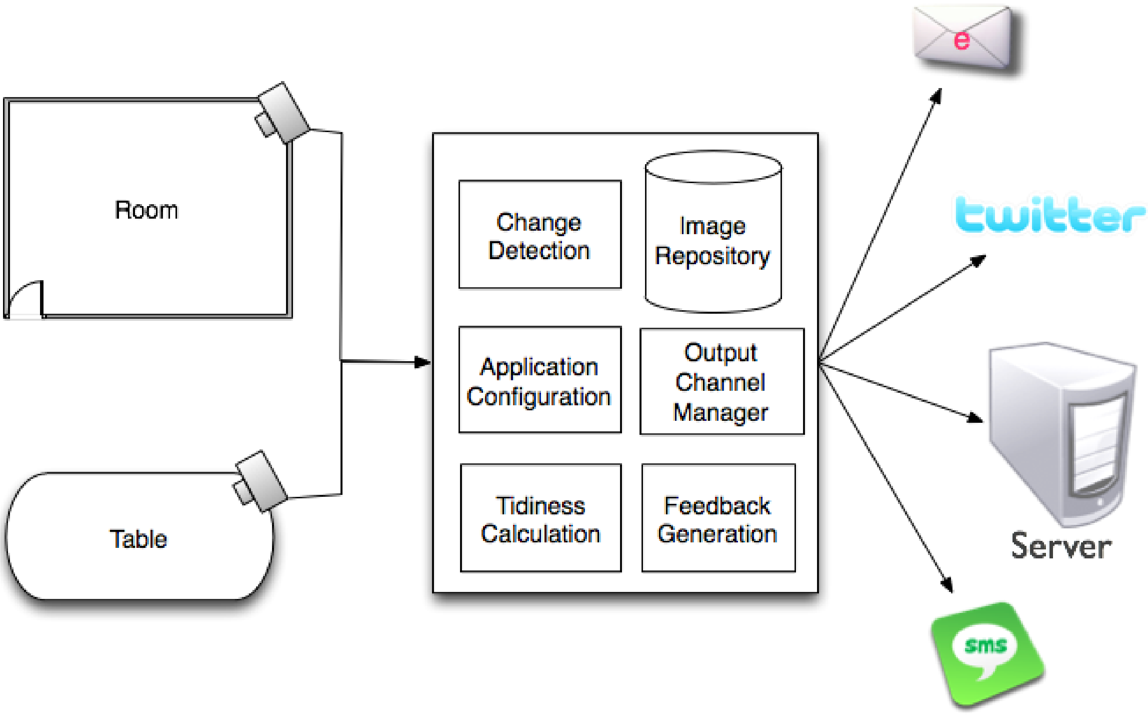


Figure 4.14: Conceptual architecture of the system

As has been described, the camera location is assumed to be fixed and thus no geometric adjustment is needed. Additionally the camera is connected to the personal computer and the implemented prototype takes pictures automatically with no user intervention (when the picture is being taken, the hosting system should be turned on). In order to ensure a fix amount of light pictures will be taken at specific time intervals. The system has light adjustment capabilities, but a fix amount of light could improve the change detection quality. It is possible to use any webcam or even a still camera which can be connected to the computer and is able to capture pictures by using the command line.

[8] http://www.mathworks.com/products/image

The system will take a picture every day, based on the predefined time. Pictures will be compared to the ideal picture and clutter status will be calculated and streamed to output channels. Figure 4.15 shows two images from a desk, the result for which returns as “Bad”.

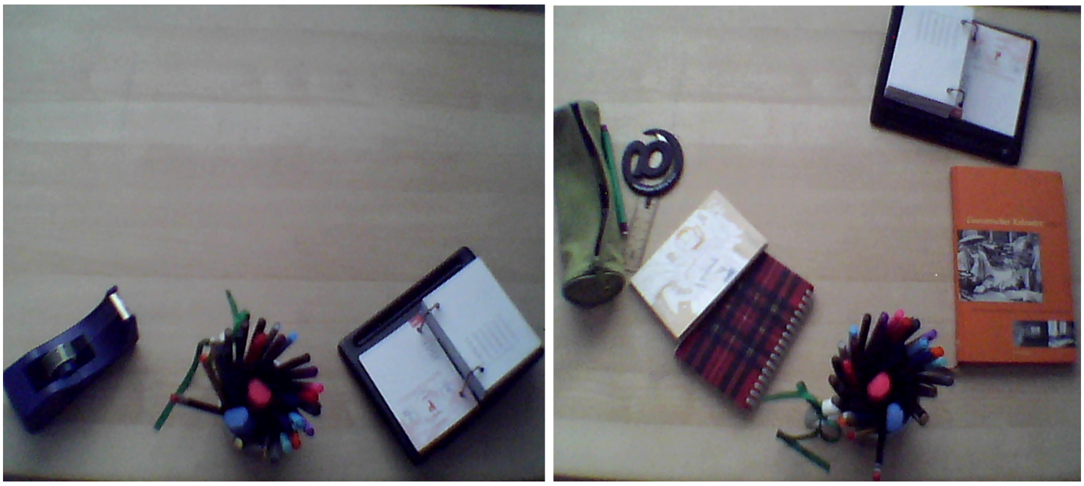


Figure 4.15: The image on the left is an ideal image from the desk (based on user’s choice) and the image on the right is the same desk when it is unordered with more than 90% changes (based on comparison with the ideal image).

We employ users’ Facebook and email as output channels in the prototype. Facebook account is disabled by default, but users should provide their email addresses to the system. It is possible to extend the output channels and to add a new medium e.g. an application for a home media player.

The prototype sends email notification which contain (1) a table 4.4 that shows the tidiness status of last month and (2) the following message: “Your room has had the status Bad for THREE days. Take a short break to tidy up.”. The content of the message which will be streamed to the social output channel could be: “My desk has had jungle status for TWO days”. We use low-controlling and concrete messages because high controlling language and long messages could increase reactance and diminish persuasion [149, 171].

It is notable that the architecture is flexible enough to allow developers to add new output channels. We assume that email is a useful channel in comparison with others, because users are keen to check them and keep them in order. Output channels depend on the target group who use this system, e.g. if they are children, the output might change from email to another such as a software application with visual features.

Measuring Tidiness

In order to measure the tidiness of a target environment, differences between the ideal image and the recent image will be calculated. The calculation will be done based on two methods; one is for the desk and one is for other environments such as the whole room. In particular, we measure color intensity differences between two images [232], in percentage. First this method converts the image to a gray scale, then it compares two pictures pixel by pixel. The intensity difference is calculated between pixels of the same coordinate. Then we calculate the arithmetic mean of intensity differences of all pixels. Nonetheless the change percentage is not shown to users and instead some adjectives will be used. If the result is more than 50%, the term “jungle” will be used. Between 50% and 40%, “Bad” between 40% and 20%, “Good” less than 20%, “Perfect”. These thresholds have been identified through a formative evaluation [200] on three users who had been involved in the evaluation.
This approach is our general approach, but because of the importance of desk tidiness, we develop a specific method to calculate the tidiness status of users’ desks. The desk tidiness calculation will be done by detecting changes of objects in two images using SIFT algorithm [136], which can detect objects on the desk. First we threshold the image, then objects on the target desk are identified by detecting their edges and stored in a repository. This process will be continued for all objects that are located on the desk. Afterwards, objects in the ideal image will be compared with objects in the current image by using SIFT algorithm. The tidiness status of the target environment is measured by calculating the percentage of object changes between images.
To realize the desk scenario let N(i) be the number of objects, which were on the ideally ordered desk and let N(c) be the number of objects, which are currently on the desk. N(s) is the number of objects, which are in both ideal and current situation images and N(n) is the number of objects, which are only in the current

situation. The tidiness status of a desk (T) will be defined as follow:

$$T = N(s)/N(i) - N(n)/N(c) \tag{4.1}$$

T is a real number between -1.0 and +1.0. We assume that the desk has one of the following statuses:

1. **Perfect:** 1.0 $\geq$ T $\geq$ 0.66

2. **Good:** 0.66 $>$ T $\geq$ 0.33

3. **Bad:** 0.33$>$T $\geq$0

4. **Jungle:** 0 $>$T $\geq$-1.0

In order to evaluate the system, five users installed and used the prototype for one month. They are between 24 and 31 years of age, and active in using social network sites. They have good or average computer skills. Three of them were women and two were men. Two of them installed the prototype on their desk, one of them in the room, and one for both. The system was tested for one month without changing the predefined settings. It sent an email a day (if the result was not Perfect or Good) and a Facebook status update once every three days.

After using the system for one month, we conducted a small survey to evaluate the system and asked them to answer some questions about (1) prototype quality and (2) persuasive effect. Four users ranked the prototype quality good or very good and one ranked it below average. In another question we asked them to describe the prototype usability and problems. Two users argued that keeping the lights turned on at a specific time is a cumbersome task.

In another question we asked them to describe the influence of this system on their environment status. Three of them described that sharing the status in their social network was interesting and they would like to continue using the system. They described that other users' comments and feedback motivated them to change their behavior. One user decided not to share anything on the Facebook and she argued that sharing was a threat to her privacy. Another user claimed the system was not useful at all, but she shared her information. Although we can not generalize

the results of this small user group evaluation, the results show that sharing this information could have better persuasive effects than not sharing it at all.
Assigning a number to the tidiness status is not easy, because individuals interpret tidiness differently. Therefore we asked users about the quality of the prototype for tidiness calculation. Four users stated that they were satisfied with the result and it was acceptable for them.

Restrictions and Controversies

Although the framework and its functionality satisfies most users (four out of five), we have identified the following challenges and shortcomings. They have been identified based on users feedback after using the prototype for one month.

- Evaluating tidiness is superficial: It might be argued that employing technology and additional hardware devices to monitor and persuade users to be tidier is a superficial approach. But the tidiness status provides interesting information about a user's personality [206]. Moreover, this information in combination with other user's information could provide details about the user's mental state [142]. For instance it can assist psychologists in studying patient health or combining this information with other longitudinal detail of the user, enabling researchers to extract knowledge about the user's life e.g. room tidiness could have direct relation to the amount of time the user spends away from the target environment.

- Environment decoration changes: Changing the decoration of the target environment and moving objects could affect the tidiness calculation, so users should take a new ideal picture to restart the calculation process e.g. adding a new couch to the room requires creating a new ideal image.

- Different objects with similar visual properties: Another problem which has been identified is objects of similar color, size and shape. For instance, if the user puts a black doll on a black keyboard this may not be registered as a change in the picture.

- Nuisance changes of the target environment: This approach supports only rigid objects. Non-rigid objects such as fruits and flowers in the room are not covered by this research, because over time the visual appearance changes and this affects the tidiness calculation. It is usual that rooms or desks contain nonrigid objects.

4.7 Knowledge Mining Approaches

In previous sections two sensing tools, a mobile phone and an indoor wall mounted camera, have been described. In order to prove the usability of the collected information, here two knowledge mining methods will be introduced: life routine detection and surprise detection. First I describe a method to detect life events from the mobile phone life-log dataset, then I explain the surprise detection methods. The surprise detection section suggests two approaches (general and custom), which enables the system to detect surprising events from the life-log dataset automatically.

4.7.1 Detecting Life Events and Routines

The life-log dataset, which here is created by the mobile sensing tool and a wall mounted camera, can be used for learning different aspects of a user's life such as social interactions [74], transportation, environmental monitoring [125], etc. In general it could be used to create a user model which includes the user's behavioral data. In order to create such a model it is important to learn users' daily life routine. Identifying life events is a necessary step to detect life routines. The life-log dataset contains many raw or semi-raw sensor data and there should be a method that can enrich this raw or semi-raw sensor data semantically. Labeling and annotating raw sensor information within life events can be an appropriate semantic enrichment mechanism.

In a more technical sense, I describe a method for detecting life events by using location sensors (GPS) and a timestamp. Existing research has focused on detecting life events from visual sensors [47, 128], accelerometer sensors [138] or audio sensors [137].

The event detection approach will be a two step process, first recognizing events then clustering similar events. The result of clustering can be used to detect life routines.

Spatial Event Identification

In previous chapters a life event has been described as a set of data entities in a specific period of time. Data entities can be represented as E_i. A life event which is set of data entities can be presented as $\sum_i E_i$, which is a collection of data entities. Spatio-Temporal dimensions of our life constitute the foundation of this event detection method. When the user is in outdoors, there is a GPS signal available and when the user is indoors there is no GPS signal available. These location changes are assumed to be borders between life events. As the first step, this method analyzes log files based on location log start and finish time. The result of analyzing files identifies life events start and end time. For a better clarification consider the following pseudo log file, this method can extract four life events from bottom log: The first event occurs before time t_2. The second event occurs from t_2 to t_4, which the location signal appears (the user is moving in outdoors). The third event occurs after t_4 to t_j (the user is in an indoor environment). t_j is the time that location data is available, therefore it means the user is moving. Fourth event begins after t_{j+2}.

```
...
{"Application":{"Time":"t1"}}
{"Location":{"time":"t2"}}
{"Location":{"time":"t3"}}
{"Location":{"time":"t4"}}
{"Call":{"Time":"ti"}}
...
{"Application":{"Time":"ti+n"}}
{"Location":{"time":"tj"}}
{"Location":{"time":"tj+1"}}
{"Location":{"time":"tj+2"}}
{"Application":{"ProcessName":"...","Time":"tj+3"}}
```

The second step is to cluster similar events. Clustering similar events is necessary, because our life conforms to some routines and many of our life events such as going to work or university are takes place frequently. Therefore a method should be available that can identify these events and distinguish them from events which do not take place frequently, e.g. going to a hospital. Moreover the result of clustering could assist in semantic enrichment i.e. labeling and annotation. Once the user can annotate an event then the similar annotation can be used for all events in that cluster.

This method is highly location sensor dependent. The GPS sensor of some smart phones is very weak and it might be hard to receive the signal when the user is always outdoor. Moreover GPS sensor needs a few seconds to get the location and this method can not take into account the delay. However the location changes are good candidates to identify events and a combination of sensor data with GPS can be used to detect life events, e.g. bluetooth and GPS. The appearance of a specific Bluetooth signal can be interpreted as a life event, for instance whenever the user goes to her office a bluetooth signal will be detected from her personal computer there.

Another challenge is the definition of life events. Getting married and studying for a Ph.D. are both life events too. Life events could be weaved together or they could be hierarchical. For instance studying for a Ph.D. is a life event which takes several years and has many sub-events. This method has been designed for very simple daily life events because our cognition is restricted to available sensors. We can augment our cognition with a combination of sensors, but now the focus is on location (GPS) sensor only.

Temporal Clustering

After recognizing events based on location logs, we get a set of time intervals, for example a set as follows: {8:50-11:50, 9:00-12:30, 9:00-13:00, 14:00-18:00, 12:00-14:30, 13:30-19:00}. The first challenge is how to collect similar or same time intervals in smaller groups e.g. {8:50-11:50, 9:00-12:30, 9:00-13:00}, {12:00-14:30}, {14:00-18:00, 13:30-19:00}.

In unsupervised learning we use clustering method to identify patterns that collects similar data together. Therefore a clustering method is required and these data objects are in raw format. This means that they don't have any label or annotation. Han et al. [101] classified clustering method into five major classes: partitioning methods, hierarchical methods, density based methods, grid-based methods and model-based methods. Since we have a set of time intervals and we should partition them in a way that a similar start-time and finish-time stays in the same cluster,

we need a partitioning method.
A custom *graph clustering* approach can be used to cluster these time intervals. Graph clustering refers to the task of grouping vertices of a graph into clusters [196]. It is notable that graph clustering is different to clustering a set of graph with a similar structure. Here only one weighted graph is created. The graph is represented as $G(E, V)$, in which E are edges and V are vertices. Vertices of the graph have been assumed to be the time intervals (previously given). The weight of edges have been assumed to be the differences between each two vertex, which is calculated as a sum of absolute differences between start times and finish times of each time interval. Each vertex (time interval) denoted as (x, y) in which x is the start-time and y is the finish-time, then the edge weight between time interval i, j will be calculated as follows:

$$w(v_i, v_j) = \mid x_i - x_j \mid + \mid y_i - y_j \mid \tag{4.2}$$

In order to calculate the differences for all time intervals, all edges should been connected together. Therefore we have a *symmetric graph*, in which the degree of matrix is equal to $V - 1$. The clusters will be calculated in two steps: first by comparing the weight between two edges and check if weight between two vertices $(w(v_i, v_j))$ is less or equal to the predefined threshold then they will stay in the same cluster (the threshold is denoted as λ and how can we identify the threshold will be explained later in this section). Otherwise if $w(v_i, v_j)$ is larger than λ, then those edges will not stay in the same cluster. The following shows the formulation of cluster creation. Let $C(v_i, v_j)$ be the function that checks if vertices (time interval) v_i and v_j are in the same cluster or not.

$$C(v_i, v_j) = \begin{cases} 1 & w(v_i, v_j) \leq \lambda \\ 0 & w(v_i, v_j) > \lambda \end{cases} \tag{4.3}$$

Figure 4.16 shows that a daily life of a users is composed of a set of segments, each segment represent a life events. It shows that each temporal segment (life event) in a four day study passes to a cluster except segment x. At day three user remained home and does not have any location changes. How to highlight this kind of events (surprise detection) will be described in the next section of this chapter.

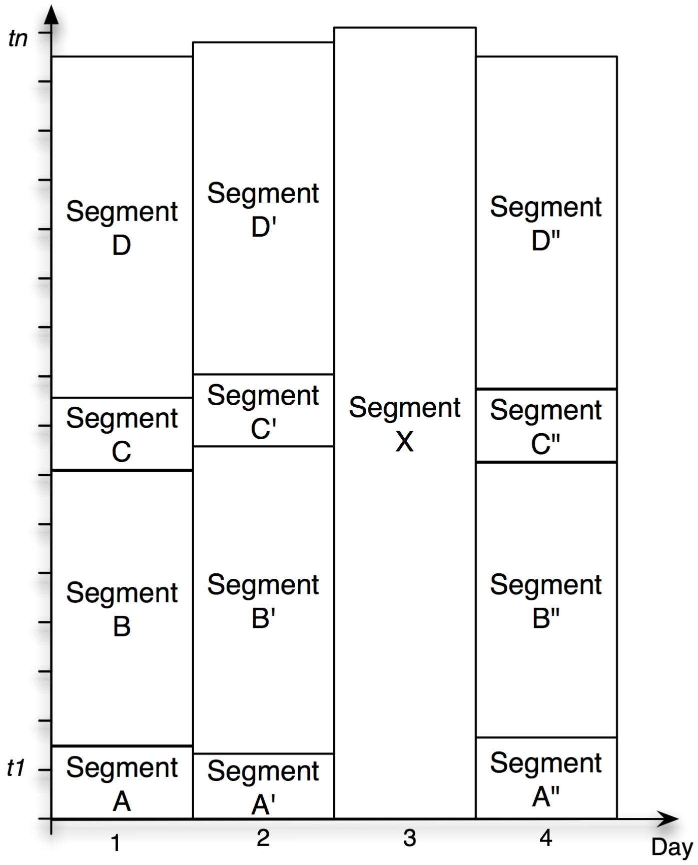


Figure 4.16: Day 3 is one segment and it has been named segment X to highlight that it does not fit into any of the existing clusters. Other days contain segments which fit into a cluster.

In order to calculate the threshold first we create another set $W = (w_1, w_2, ...w_n)$ which contains all weights between vertices. For instance, for the given time interval dataset example we have something similar to the following set:
{00:30, 00:50, 1:20, 10:20, 5:30, 10:30, 2:30, 6:30, 7:20, 1:30, 11:40, ...}. Then k-mean clustering [143] will be used to cluster weights. The result is something as follows:
{00:30, 00:50, 1:20, 1:30, 2:30}, {5:30,6:30,7:20, ...}, {11:40, 10:20,..}.
The k-mean clustering algorithm clusters $x_1, x_2, x_3, ...x_n$ data objects into k disjoint subset S_i contain j data objects:

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} \mid x_j - \mu_i \mid^2 \tag{4.4}$$

μ_i is mean of cluster S_i.
The cluster which has the smallest set (in the above example the first one) will be chosen. The maximum value there (which is 2:30) plus an error (which I assumed half an hour) will be the threshold. In summary the threshold will be calculated as the maximum value from the cluster of minimum weights, plus the error. As shown in the equation 4.5. m is the number of clusters and respectively K_m denotes the cluster that contains minimum weights. n is the member of subset S_j which has the maximum value, in our example n is "2:30". Therefore the *lambda* in this example is 2:30 + 0:30, (0:30 is error). K_n represents nth variables in a cluster with m size and $\sum_1^m$ represents all variables in that cluster.

$$\lambda = \arg\max_{n \subset S_j}(\arg\min_{K} \sum_{1}^{m} K_n) + \Delta \tag{4.5}$$

As has been described before the main motivation for life event identification is to provide semantic enrichment to raw and semi-raw sensor data. For instance annotation, which facilitates searching and browsing the life-log information. It is not feasible that everyday the user annotates and labels all of her life events. This method, which can cluster similar events into a group, enables users to annotate once an event in a cluster, then the next time the same annotation can be applied to similar events. Annotation is not the only benefit of clustering, we can learn and summarize users routines too. This feature can be used in many use-cases such as context-aware personalization or contextual based recommendation systems. For instance we can create a behavior profile for a user as follows:
Usually, during the week, the user leaves home between 7:00 - 8:00 AM, it takes about half an hour to get to work, about 12:00AM - 1:00 PM she has lunch for an hour. Then she leaves her office between 4:00 to 5:00 PM, it takes about one hour to get back home. On sundays between 9:00 to 11:00 the user starts an outdoor activity and finishes it between 13:00 to 13:30.
This behavior profile can be converted into an RDF [126], and a SPARQL query can be used to predict user's context. For instance the following SPARQL code predicts

the user's location at 10:30:

```
SELECT ?location ?x
WHERE {
  ?x rdf:type ontology:UserProfle .
  ?x onto:time "10:30"^^xsd:string
  ?x onto:location ?location .
  }
```

4.7.2 Surprise Detection

Eric Horwitz [105] stated that combining massive quantities of data, insights into human psychology, and machine learning can help manage surprising events as follows:

Surprise aimed at developing and fielding sensing and reasoning systems that have the ability to detect and to alert users when current or future events will likely surprise them.

Surprise detection could be interpreted as a specific type of anomaly or outlier detection in a dataset, which users do not expect and it trigger users' attention. It is notable that surprise is something subjective and depends on users' preferences. Therefore two approaches has been introduced here, a *general* surprise detection approach and a *customized* approach.

The general approach uses an existing method [109] for surprise detection and I describe how this method can be applied to the life-log dataset. In other words this method has been used for visual information but here it is used for textual time series information.

General Approach

Most surprise detection methods are based on statistical analysis. Frank [84] introduced the surprise value (Überraschungswert) based on the concept of entropy, He defines the anomaly of a data object on the total entropy of a system. From the information theory perspective information entropy [204] is the measure of uncertainty of a random value, which quantifies the expected value of the information contained in a message. Equation 4.6 shows the Entropy function, where $p(x_i)$, is the probability mass function of outcome x_i, n is the number of possible values.

$$Entropy(x) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \tag{4.6}$$

Frank described that the surprise value is being used to get the attention of subjects, for instance it can highlight the size of an area which is different from other parts of the picture and gets the viewer's attention. Chakarabatri et al. [50] proposed the notion of surprise through multivariate time series analyze of the target dataset. They extract departure from the base model by characterizing the surprise based on the number of bits. Itti and Baldi [109] described a formal Bayesian definition of surprise. The Bayesian network requires knowledge of many probabilities and this can be estimated by previously available data [153]. The Bayes theorem is as follows:

$$\forall h \epsilon H, P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)} \tag{4.7}$$

They have evaluated their theory by checking users gaze at surprising items while watching television and video games. Their surprise theory is applicable across different spatio-temporal scales, and it is based on the difference between the posterior and prior distributions of beliefs of a user over the available class of models or hypotheses about the world. In simple terms, the new data observation contains no surprise if the posterior observation is identical to prior observation. This theory can be used for life-log datasets too. The background observation defined as prior probability distribution $\{P(h)\}_{h \epsilon H}$, h is the hypothesis and H is the model space. The new data observation D is to change the prior distribution into the posterior distribution $\{P(h \mid D)\}_{h \epsilon H}$. Relative entropy (or Kullback–Leibler divergence) [121] is the result of dividing the entropy by the maximum entropy, as shown in the equation 4.8. It has a value of 0 when two probability distributions are the same.

$$D_{KL}(p \mid\mid q) = \sum_{x \epsilon X} p(x) \log \frac{p(x)}{q(x)} \tag{4.8}$$

By employing the bayesian theory and using it for the calculation of relative entropy the surprise value will be calculated as equation 4.9.

$$S(D \mid H) = D_{KL}(P(h \mid D) \mid\mid P(h)) = \int_H P(h \mid D) \log \frac{P(h \mid D)}{P(h)} dh \tag{4.9}$$

According to equation 4.9 a surprise unit can be defined for a single model h as the amount of surprise which is corresponding to a variation between $P(h \mid D)$ and $P(h)$. In a more technical sense, surprise quantifies the differences between the posterior and prior distribution.

This approach can be used to detect events in life-log datasets, which involves more users attention. For instance we can calculate surprise to realize the abnormal social communication, through analyzing the dataset if there are significant calls or SMSs which have been made to a specific person in a short period of time.

Customized Approach

Another approach for detecting surprising events is the customized approach, which is based on individuals settings. Here individuals means either end users or system administrators. We have extended the room tidiness metaphor to detect unusual objects on the desk and mark them as a surprising event. For instance a shoe on the table is a salient anomaly on the desk and if the camera is able to detect it, then the system can annotates that specific record (picture with timestamp and tidiness status) as surprise. This means that on that specific date-time a surprising event has happened.

To implement the surprise detection algorithm on the table we extend the tidiness metaphor to have another database which keeps the recognized objects in it. If (1) an object appears on the table which have different visual characteristic than existing objects in the database and (2) the object disappears after a specific time (in this case one day), then this object represents a surprising object. As has been described before the visual characteristics of objects on the table will be recognized by SIFT[136] algorithm. Figure 4.17 shows objects which have been identified on the table (green border) and they exists in the database. There is another object, a shoe (red border), which is not previously existed in the database and has disappeared

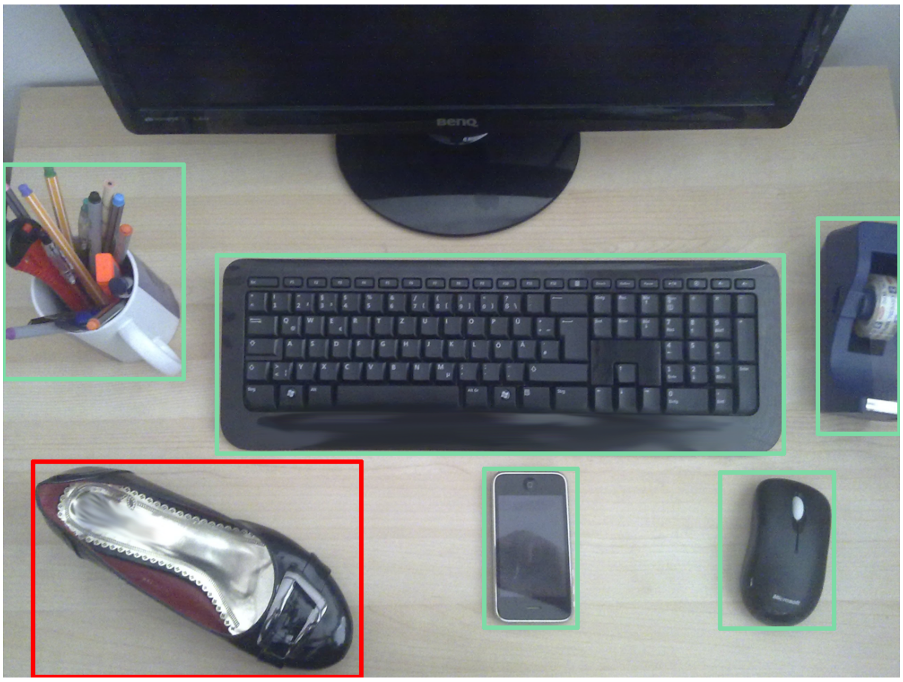


Figure 4.17: Objects with green borders have been identified and exist in the database of table objects, but the shoe, which has a red border, has not been identified before and does not exist in the database. Since the computer monitor position is fixed, it has been assumed to be part of the table.

in a day. Therefore it has been marked as a surprise evidence and the associated record will get the surprise annotation.

Even information from both prototypes can be linked together through time stamp and a surprise event can get extracted. For instance the status of a target tidiness in combination with location changes of a user, can be used to calculate significant behavior changes (surprise) in users.

Moreover, the surprise calculation can automatically annotate and highlight specific life events in the dataset, and this feature can resolve the WORN effect.

Chapter 5

Privacy, Security and Trust

It could be argued that the most important challenge of life-logs is users' privacy, which reveals the importance of security. In particular life-log information requires higher security considerations since it may contain very privacy sensitive information about the user such as biological information, location, communication logs, etc. Mitchell [152] stated the following about new technologies which sense and record human behavior (life logging):

Perhaps even more important than technical approaches will be public discussion about how to rewrite the rules of data collection, ownership, and privacy to deal with this sea change in how much of our lives can be observed, and by whom.

Previously I have described the controversial history of life-logs and why risks of using life-logs should not hinder technological development in this area. Moreover I have explained a fine-grained access limitation model, but it is not enough to reduce risks. Ethical considerations, securing the process of life logging and enabling users to have control over their information, should be considered for users' privacy too. These solutions could increase trust between the user as the service consumer and the service provider. A basic level of trust will be required to motivate individuals using these services. Collins et al. [57] proposed a rather similar strategy based on John Rawls' Theory of Justice [186] towards assessing the quality of software products. First they identified roles in creating and using a software application and then they performed risk assessment for each role. Later, ethical considerations for each social role in their scenario were proposed. This approach is similar to theirs, but here the intention is focused on a specific type of software application (life-logs).

The remainder of this chapter is organized as follows: First I start by describing how to secure the process of life logging. Then I describe how users' can benefit from their personal information. Then I propose a prototype which enables users to manipulate (anonymization and pseudonymization) their personal information based on their own preferences, for third party usage. Afterwards I finalize this section by describing ethics for using life-log tools and sharing the information.

5.1 Software Architecture Security

Securing the process of life logging is an important issue, which should be taken into account during the implementation of a life-log. Vermuri and Bender [227] who developed one of the earliest effort in life logging, through employing pervasive devices, stated that: *Recording everything is easy part. The interesting challenge is turning vast repositories of personal recordings into a useful resource while respecting the social, legal and ethical ramifications of pervasive recording.* Here the focus is on general requirement for securing the life logging process [183], and how this requirement has been implemented in the implemented prototypes. In order to provide a general security approach the usual life logging steps will be defined. It has been suggested that developers should address these security issues during the design phase of a life-log system. In the following I describe which parts is supported by the implemented prototypes.

Usually, a life logging process has three stages and each stage requires specific security considerations. The first stage is sensing the information from the user environment with sensors; the second stage is collecting the sensed information; and the third stage enables users to browse and retrieve information from their life-log dataset. Figure 5.1 shows the generic life-log component architecture. There I have highlighted the parts of a life-log system that need to be secure.

As has been described in implementations, users should be able to define what information object they intend to collect. Users might need to configure sensors in order to set their configuration parameters such as sensing interval, etc. The first stage connects the life-log system to sensors and reads sensor data. At this stage two things might need to be secure. First some sensors could require authentication, second if the sensor data contains sensitive information data transmission from the sensor to the life-log tool should be secure too e.g. encrypted data. I suggest providing dynamic security modules for the sensing stage since sensors might be added or removed dynamically to the life-log tool, therefore a non-centric dynamic security module for each sensor increases the flexibility and scalability of the life-log tool. In the recent UbiqLog implementation there is no sensor that require authentication, and all available sensors are configureable. The data transmit from sensor to data reader does not require security, because sensors are connected directly to data readers.

The second stage is to collect the sensed information in the life-log device. This stage creates a dataset of the life-log information. The dataset contains a set of life-log records. Data that comes from the sensors is mostly raw data and in order to enable users to browse and access them, some changes have to be made to the raw data. Changes might include annotation, aggregating sensors' data, migrating data from one format to another format, etc. During the collection phase developers should consider the third party tools that they are using to change data. For instance a security threat might be the use of an annotation engine from a third party which sends users' information to that third party. The implemented prototypes did not use any third party libraries at this stage.

The third stage is storing the life-log data. Storages (storing devices) which host life-log information should be secure. It has been suggested to maintain data in encrypted format if are intended to be stored as files, or if they will be stored in a database, designers should consider to define appropriate access rights on the database. The implemented server stores information in plain text which is not secure, but it is a prototype and in real business case they will be not stored as a plain text.

Life-log devices are pervasive devices, hence they are not capable of hosting personal information. Therefore, life-logs should maintain their data on reliable storage media or server. If a life-log tool does not use a pervasive device then there is no need to have a local storage and data can be stored directly on a reliable storage media. But life-logs usually contain at least a pervasive device. Transferring data from a pervasive device to a reliable storage media might not be a standard stage for a life-logging process, but it is normally required. Communications and data transfers are very sensitive from a security point of view and sharing information in a social community requires communication. This demonstrates that during the communication, data should be transfer encrypted. Securing the connection can be done by the transport layer security (TLS). However additional to TLS, using a digital signature has been suggested as an additional way to secure data transfer. The implemented prototypes use TLS connection for all communications between Data Readers and the server.

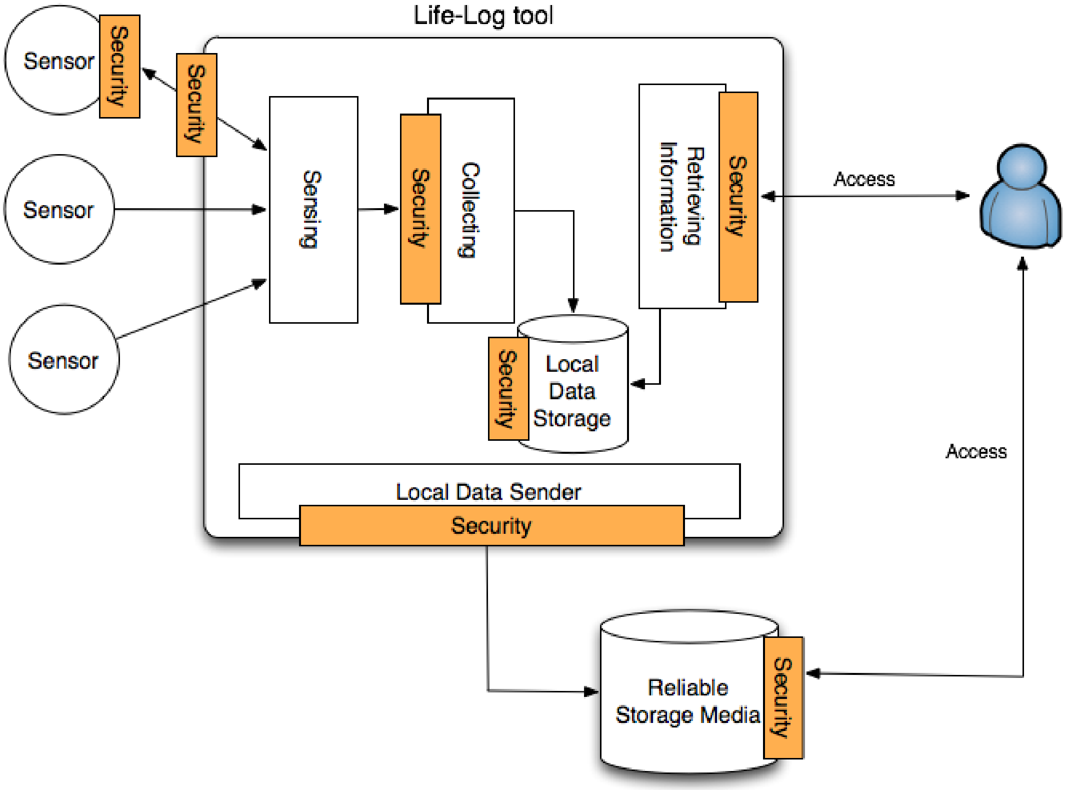


Figure 5.1: Security requirements in a generic life-log architecture.

Current SNSs (social network sites) are based on client-server architecture and, because they own servers, they have access to users' information and they probably own the users information. Although the implemented server is not completely decentralized, life-log information could be shared via a peer-to-peer (P2P) SNS such as a PeerSoN [44]. The P2P social networks have a major advantage over traditional social networks. In a traditional SNS users upload their personal information on the service provider site, but in a P2P SNS users keep their information local. Hence, the chance of misusing users information is reduced and there will be less privacy risks.

A P2P SNS requires the provision of secure access to the shared life-log information. Here secure means to perform the authentication and authorization on the users who intend to access to a life-log information. On the other hand communication between nodes in a P2P SNS should be secure. As it has been explained before, communication and data transfer are very sensitive. Here we used Diaspora social network and thus we rely on Diaspora security.

During the design phase of secure modules, other design considerations should be taken into account e.g. designers should consider that, unlike desktop applications, pervasive applications authentication and authorization processes should not to be intrusive [22, 60]. Using the standard RBAC (Role Based Access Control) model for accessing the information is an appropriate method of limiting unwanted access, because a user can chose a user or group of users and grant them the appropriate access. Each access will be granted with an expiration timestamp as it has been explained in the “Data Model and Sharing Policy” section of this research.

5.2 Control Over Personal Information

Nowadays our personal information are distributed among large and ubiquitous databases that we can not manipulate or review our information there. Moreover it is impossible to know if our information have been held securely and reliably. This means that Personal information distributed asymmetrically (service provider has more information about service consumer than service consumer about service provider). The availability of personal information online, whets the appetite of businesses that try to profit from this, e.g. for marketing purposes. A new branch of industry focuses on the collection of personal information including visited web sites, movie preferences, and purchases over the Internet, using tracking cookies or the like to create individual profiles that are sold to third parties [14]. Fried [87] defines privacy as control over knowledge about oneself', according to his definition privacy is one of the base factors in establishing new social contacts. Laudon [127] suggests that there should be an infrastructure available which enable individuals to sell or benefit from their personal information. Now users subscribing for a using a service and the service provider benefits by accessing users' personal information. The status of personal information market is based on the notion that the gathering institution is owns the personal information and that individuals have at best an interest in -not owenrship- of information about themselves [127, 224].
Technological advances, the increasing commercial profit of user profiling, and growing amounts of shared personal and sensitive information - either wanted or unwanted - will likely intensify this problem even more. To counter the privacy problem, several legal acts were introduced such as the Health Insurance Portability and Accountability Act (HIPAA) [223] that regulate the use and distribution of sensitive information to prevent fraud and data abuse. At the European level, the processing and movement of personal data is legally regulated by Directive 95/46/EC [77]. But often legal regulations are not that effective as they should be: Service providers that deal with personal information claim to preserve the user's privacy, but try to circumvent restrictive regulations by using difficult-to read terms and conditions. Social network sites hide privacy improving functions deep in the user interface such that they are easily overlooked [40]. Even if the service providers are reasonably

trustworthy in protecting the individual's privacy, there remains the issue with insider attackers. Thus, the point of the matter is to let the user control with whom to share information and to what extend, resulting in the requirement of user-centered and user-controlled privacy solutions. If users' are in control of their life-log information, rather than traditional style which service provider are in control of users' information, then users' can be able to implement forgetting on their personal information too.

In order to implement the above described requirement we propose LiDSec (Light Weight Data Security) [180]. LiDSec is a novel lightweight approach for privacy-preserving data sharing relying on pseudonymization, a technique which effectively hides the association between personal information and the data owner [52] and also reduces trust expectancies when sharing personal data with a community [141]. The main idea is to strictly control the information flow to information recipients instead of trusting these parties not to abuse sensitive information. This approach is owner-centered, i.e., the data owner specifically decides on which information to disclose by creating document-specific rules that are processed to create non-critical datasets. The proposed tool is sufficiently flexible to handle and process any type of text based information. [1]

Design Principals

To achieve our goal of ensuring privacy-preserving data sharing, we have identified the following requirements:

User Control: The most important requirement is to ensure user control, or more specifically, data owner control, i.e., let the data owner decide on the "amount" of privacy of published datasets. Unlike with typical privacy settings of, e.g., social networks, this also includes that the owner enforces her privacy demands in order to reduce the trust required in data consumers. A considerable threat is the existence of potential inside attackers, such as malicious administrators with extensive access rights that are able to circumvent user privacy settings.

[1] We focus on structured text-based data that can be processed and consumed automatically (e.g. web services). Information represented in binary form (e.g. images) is outside the scope of LiDSec.

Efficiency: The straightforward (reversible) approach of preserving data confidentiality and privacy is to encrypt the sensitive elements. However, encryption requires dedicated key management and key protection strategies and thus is always associated with a certain amount of overhead. While the execution of cryptographic operations is usually unproblematic on current conventional hardware, the limited calculation power of pervasive devices such as mobile phones or tablet computers still poses a considerable obstacle for efficient use of key-based cryptography.
Adaptability: Existing privacy-preserving solutions are usually aimed at specific application domains and designed to handle certain types of information with a predefined data structure. Regarding the diversity of text-based data representation methodologies, an effective privacy-preserving solution is required to be highly adaptable to different data structures.
High Level of Automatization: Considering the large amounts of information that are involved in data sharing these days, manually identifying and processing privacy-sensitive elements is usually quite tedious. Therefore, automatizing as many tasks as possible to reduce the manual work needed by users is a major requirement in developing a viable privacy preservation solution.

LiDSec Architecture and Implementation

The LiDSec (Lightweight Data Security) system was designed based on the following considerations:

- Apply (selective) pseudonymization before publishing personal information.
- Let the data owner (in the following simply denoted as user) decide on which elements to pseudonymize by creating privacy rules depending on the document type.
- Then let the tool automatically screen for privacy-compromising elements within the documents to reduce manual pseudonymization work to a minimum.

The architecture (Figure 5.2) is composed of five distinctive components: Data Adapter, Rule Settings, Converter Engine, Validator, and Reporting Module.

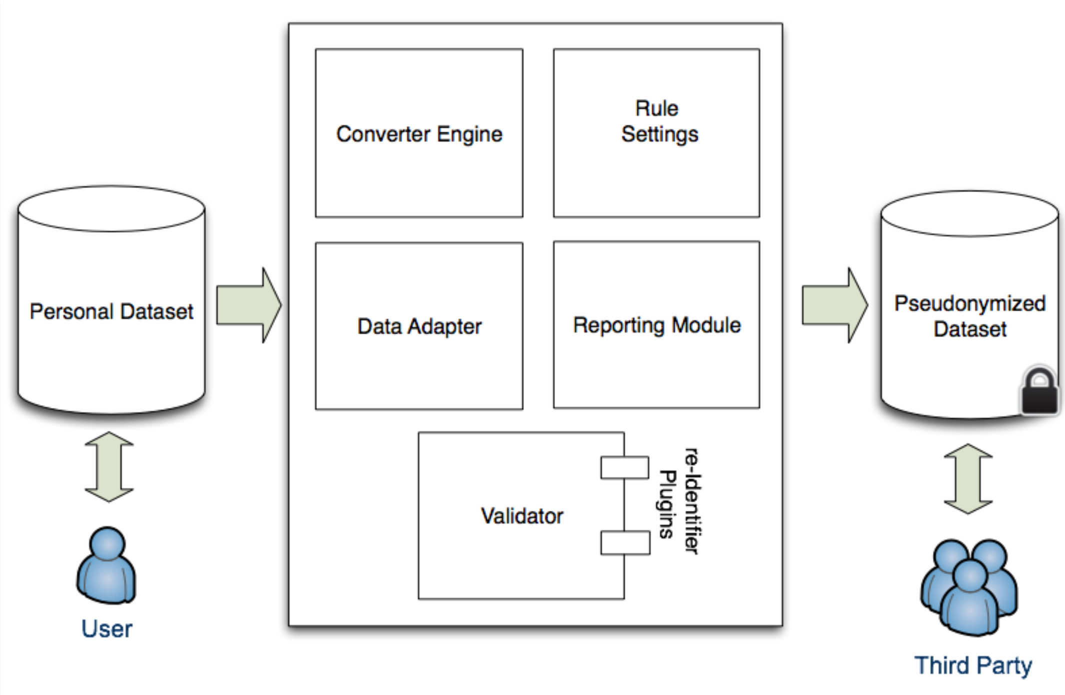
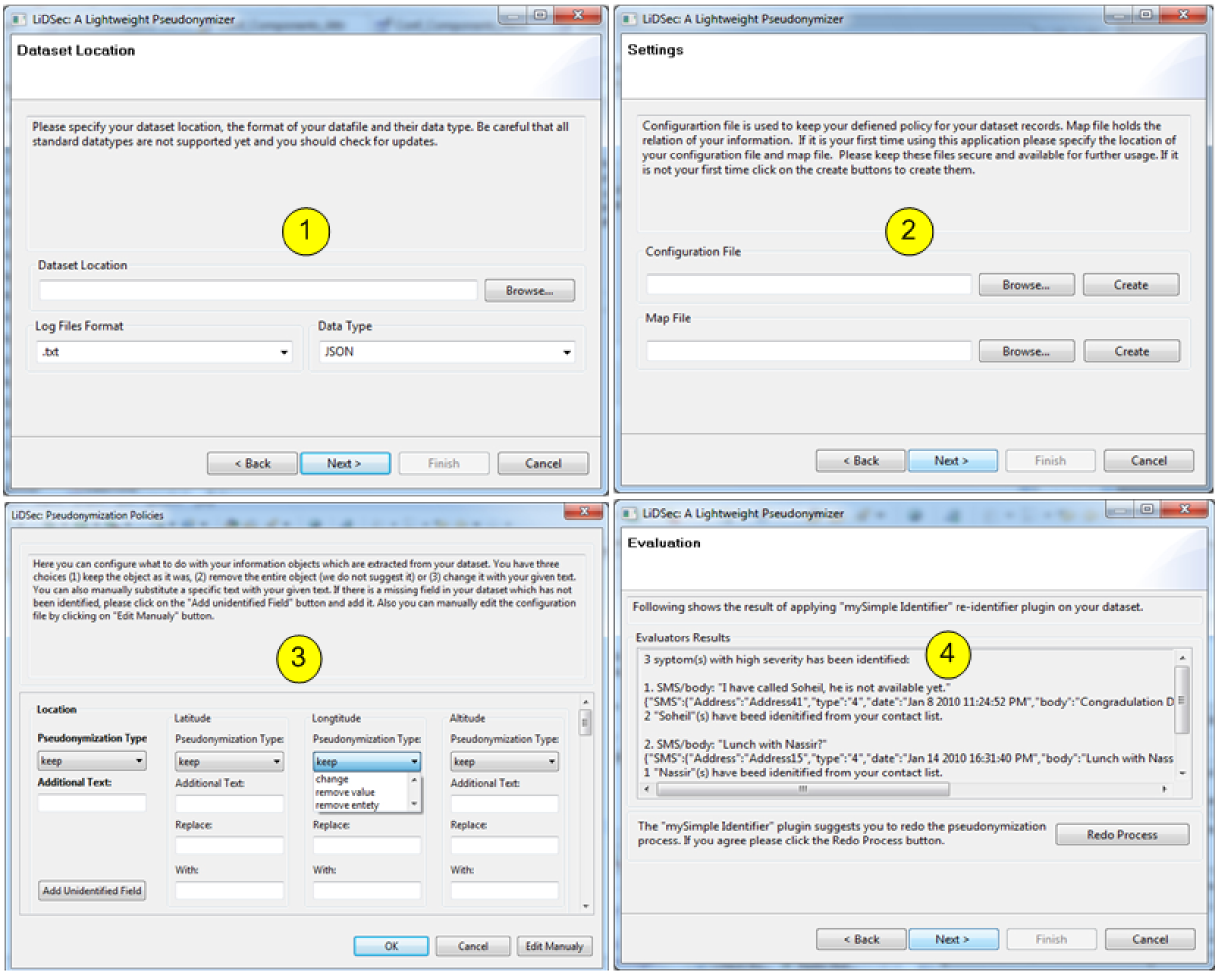


Figure 5.2: LiDSec Architecture

The *Data Adapter* is responsible for connecting to the source dataset and for extracting the data's structure in an attribute-value style. Attribute-value pairs are denoted as information objects. Depending on the actual data source, e.g., relational databases such as MySQL or simple JSON-encoded files, individual Data Adapters are required to be implemented by the framework users to manage data acquisition and loading. These adapters are then integrated with the pseudonymization framework to support multiple data sources at once. Our current implementation of LiDSec uses a GUI wizard as shown in Figure 5.3, although the actual pseudonymization can also be consumed as a service.
After loading the intended dataset (screenshot 1 in Figure 5.3), the process of data structure identification is initiated. This process parses and recognizes entities and their sub-elements of the target dataset, based on the dataset's format. As shown in Figure 5.3 (screenshot 3), the identified fields are presented to the user via the GUI in a structured way. Data structure identification is executed on in a semi-automated way: First, the logic automatically goes through the dataset and searches for and extracts known information objects. Un-identified entities that are missed by the

logic can then be added manually Afterwards.



Please specify your dataset location, the format of your datafile and their data type. Be careful that all standard datatypes are not supported yet and you should check for updates.

Dataset Location:

Log Files Format: .txt

Data Type: JSON

Configurartion file is used to keep your defiened policy for your dataset records. Map file holds the relation of your information. If it is your first time using this application please specify the location of your configuration file and map file. Please keep these files secure and available for further usage. If it is not your first time click on the create buttons to create them.

Configuration File:

Map File:

Here you can configure what to do with your information objects which are extracted from your dataset. You have three choices (1) keep the object as it was, (2) remove the entire object (we do not suggest it) or (3) change it with your given text. You can also manually substitute a specific text with your given text. If there is a missing field in your dataset which has not been identified, please click on the "Add unidentified Field" button and add it. Also you can manually edit the configuration file by clicking on "Edit Manualy" button.

Location

Location	Latitude	Longtitude	Altitude
Pseudonymization Type: keep	Pseudonymization Type: keep	Pseudonymization Type: keep change remove value remove entety	Pseudonymization Type: keep
Additional Text:	Additional Text:		Additional Text:
	Replace:	Replace:	Replace:
	With:	With:	With:

Following shows the result of applying "mySimple Identifier" re-identifier plugin on your dataset.

Evaluators Results

```
3 syptom(s) with high severity has been identified:

1. SMS/body: "I have called Soheil, he is not available yet."
{"SMS":{"Address":"Address41","type":"4","date":"Jan 8 2010 11:24:52 PM","body":"Congradulation D
2 "Soheil"(s) have beed idenitified from your contact list.

2. SMS/body: "Lunch with Nassir?"
{"SMS":{"Address":"Address15","type":"4","date":"Jan 14 2010 16:31:40 PM","body":"Lunch with Nass
1 "Nassir"(s) have beed idenitified from your contact list.
```

The "mySimple Identifier" plugin suggests you to redo the pseudonymization process. If you agree please click the Redo Process button.

Figure 5.3: Screenshots from the LiDSec GUI

After the dataset's structure has been identified, the user needs to create pseudonymization rules that define how to handle each entity (Figure 5.3, screenshot 3). Basically there are four options:

- *Keep* the information object as it is.
- *Remove* or generalize the *value* of a particular information object, i.e., anonymize.
- *Remove* the complete *entity*, i.e., suppress the information object.
- *Change* or substitute the particular information object with a pseudonymized one.

Anonymization usually alters the overall semantics and accuracy of the original data and thus should be applied only when necessary. Similarly, suppressing entire information objects may be necessary on highly sensitive information objects where their plain existence may result in a privacy compromise. In general, pseudonymization is the better choice for preserving data expressiveness, but it still lies in the user's hands to find an acceptable trade-off between usability of the pseudonymized dataset for the data consumer and the user's own privacy requirements.

Creating these rules is a simple process which is supported by the GUI: The user is presented with a view of all identified entities and can select the appropriate option for each of these entities (or for groups of entities). The choices are then converted and collected in the *Rule Settings* component of LiDSec which stores these choices as pseudonymization policies and applies them to the dataset during the pseudonymization process. In our current implementation of LiDSec, we store these policies in an external configuration file. These policies can be easily replicated for different applications and forwarded or modified to fit the needs of both the user and the data consumer.

After the pseudonymization policies have been clarified and persisted in the configuration file, the *Converter Engine* initiates the actual pseudonymization process, scans the dataset for the information objects stated in the pseudonymization policies, and converts or removes information objects according to the rules to produce a sanitize dataset. All replaced values, i.e., the mapping between original information and pseudonyms (e.g., "John Smith" converted to "Person12"), are maintained in a map file. This map file can be reused for different datasets if the user chooses to (as shown in Figure 4.9, screenshot 2) resulting in the same pseudonyms for the same values in these datasets. Alternatively, the user may choose to create a new file with new pseudonyms to semantically unlink the current pseudonymized dataset from any previous ones containing the same values. Thereby, the user is able to carefully control the information flow concerning the linkage between multiple datasets by creating multiple virtual identities (using different pseudonyms) with individual sets of information.

Following the pseudonymization process, the outcome's quality is checked by the *Validator*. We designed this Validator module to be easily extensible by external plug-ins to make use of existing re-identification approaches. In this context, we de-

fine re-identification as the process of successfully identifying the particular dataset's individual by analyzing the pseudonymized dataset and residual personal information that should have been removed by de-personalization or de-identification [216]. The *Reporting Module* is responsible for presenting the results to the user.
Our LiDSec prototype has been implemented in Java 6 with an optional GUI layer to assist users in working with the provided services; it can be used as application with the GUI wizard or as a service. The prototype has been tested on a Mac Book Pro (MacOS 10.6) with a 2.4GHz CPU and 2GB memory, as well as on a Lenovo Thinkpad (Windows 7) with a 2.5GHz CPU and 4GB memory. No platform-specific library is required for the execution of the prototype. Configuration and map files are both encoded as JSON files which need to be kept safe by the user to prevent information leakage [79, 170].
Following shows an example of a record (log) to be pseudonymized, containing a particular text message (SMS).

```
{"SMS":{"Address":"06802368296",
 "type":"1", "datetime":"Jan 14 2010 3:39:21 PM",
 "Body":"plz call me to schedule the plan",
 "metadata":{"name":"John"}}}
```

In this example, the user intends to publish this information to a data consumer who utilizes the data to analyze the frequency of SMS sent by the user. The user considers the phone number and content as privacy sensitive and thus creates a pseudonymization policy that hides those entries. Furthermore, the metadata entry needs to be generalized (anonymized). The resulting record is produced as follows:

```
{"SMS":{"Address":"address7",
 "type":"1", "datetime":"Jan 14 2010 3:39:21 PM",
 "Body":"body38","metadata":{"name":"_name"}}}
```

The address and body values are replaced with individual pseudonyms, while the metadata name's value is generalized to the fixed value of “name”; the time stamp and provider entries are left as they were.

Evaluation

We evaluated LiDSec from three different perspectives: pseudonymization effectiveness and quality, usability, and validity. Effectiveness is proven by testing the tool with real world data, while usability is evaluated by a user survey based on Nielsen's usability heuristics [158]. Finally, validity is proven by revisiting the basic requirements stated before.

As our test case, we pseudonymized the UbiqLog dataset, which composed of all activities a user performs with her mobile phone. The dataset consists of a set of log files - one log file per each day - encoded in JSON format. To evaluate the quality of the pseudonymized dataset, a simple re-identifier has been implemented as Validator component. Having access to the map file, this re-identifier scans the resulting pseudonymized dataset for any values stored in the map file, i.e., it checks whether a particular value of a specific attribute being pseudonymized according to a pseudonymization rule is still present within another non-pseudonymized attribute. Figure 5.3 (screenshot 4) shows a report where three re-identification symptoms are detected, all of them appearing in the text message content. In this example, the user pseudonymized the sender and receiver names, but not the (complete) text message content. Therefore, the user still needs to alter the pseudonymization policies and/or manually modify the record and remove these symptoms. In an iterative process, the policies are refined until no symptoms are identified any more.

The user interface's usability is evaluated by five users of (relatively) similar age (between 23 and 31 years of age), but different gender (two women, three men) and computer expertise (four stated having above average computer knowledge and one having only basic knowledge). The representative task was to pseudonymize a single life-log dataset due to its simplicity and expressiveness containing human readable data. All participants received an introduction about the tool's purpose, but no hints on how to actually use the tool. The results of the survey based on Nielsen's usability heuristics indicated the highest score (5 of 5) in “visibility of the system status”, and the lowest score (2 of 5) in “help and documentation” as well as “recognition rather than recall”. Other heuristic factors received above average scores.

Finally, LiDSec meets the requirements of privacy-preserving data sharing as follows:

User Control: Our approach revolves around the user to specify an individual pseudonymization policy set that can be either derived from a default set or created from scratch. This ensures the development of fine-grained and document type-specific privacy policies, including the option of replacing certain values with selected pseudonyms, removing these values, or even removing complete information objects to limit unwanted information leakage when publishing privacy-sensitive data.

Efficiency: We developed a lightweight solution with the aim of being highly efficient. As it completely forgoes computationally-expensive cryptographic algorithms or other complex operations, it is suitable to be deployed on devices with limited computational capabilities (e.g., mobile devices). Our approach does not rely on specific third party solutions and is thus largely independent from software technologies. The actual pseudonymization process requires only simple string replacement operations and thus can be executed within a very small amount of time.

Adaptability: The introduction of document type-specific data adapters allows to adapt our solution to a whole range of document domains and potential data sources. Without a specific application area in mind, LiDSec is able to handle any text-based documents, as long as its structure is known. Potential data sources include simple text files, relational databases, or other data-providing services (e.g., web services). The prototype can be used as standalone tool with the GUI wizard as well as its functionality consumed as an intermediate pseudonymization service.

High Level of Automatization: Data structure identification is handled in a semi-automated way, i.e., the document type and the involved entities are automatically determined by analyzing the document content; manual work is only necessary when unknown entities are detected. Pseudonymization policies need to be created once by the data owner and are then re-used for all entities matching the rule sets, reducing the required human workload for privacy-enabled data sharing.

While our solution is in general highly adaptable to any text-based information, we identified some limitations due to our design decisions. First, as the targeted use is to publish machine-processable documents in a privacy-preserving manner, we focus on structured data elements only; free-text documents is not supported by LiDSec. Therefore, the approach is more suited to "tokenized" data such as dates, addresses, names, basically any number or short descriptive term that can be easily replaced. The pseudonymization of records including longer content (e.g., blog

entries) usually comes with a considerable loss of expressiveness (when removed) or a higher risk of re-identification (when left untouched). Second, this solution is aimed at pseudonymizing information upon sharing with other persons, thus requires a trusted local storage to keep the original datasets[2]. Another issue may be with using the same pseudonyms for the same value occurrences in multiple datasets, which may be beneficial for the data consumer but also increases the chances of successful statistical and inference attacks. But this can be easily controlled by the data provider by simply using different pseudonyms. In general, the data provider needs to find a balance between personal privacy and data usability.
In summary this approach 1) increases the data owners' awareness of what information they are sharing, thus rendering data publishing is going to be more transparent, 2) gives owners the ability to control the access over their information.

[2] While pseudonymization of information objects is reversible via the map file, reversing anonymization and suppression of attributes requires the original datasets

5.3 Ethics for Sharing Life-log Information

Since 1960 democratic countries developed privacy laws to protect individuals, but development of privacy laws is not consistent with technological development. American and european privacy legislation is based on a regulatory regime called Fair Information Practices (FIPs) [224] and FIPs doctrine recognized individuals interest in their personal information even though these information are created by third parties.

Life logging and sharing life-log information is a new approach in social computing and computer sciences. Moor [154] described that developing technologies suffer from a policy vacuum and thus ethical problems. He noted that because of the limited human cognitive system our ethical understanding of a developing technology will never be complete and the number of ethical problems will increase as technological development progresses. Meaning that ethical considerations that I list in the following might not be complete and it is possible that some points are missing from this list and during real system usage in a business environment more ethical considerations will be discovered.

I use operational SNSs and scientific efforts in order to identify obligations and rights such as what Collins et al.[57] proposed. In other words, I try to explicitly define rights, responsibilities and obligations for service providers and users based on the available sharing models and identified ethics. Service providers and service consumers (users) both need to apply ethical considerations while dealing with these features. Allen [24] realized that existing privacy laws are not sufficient to prevent unwanted usage of life-log data. Based on her notes and our interpretation, I identify ethics [177], which are related to the life-log data, and list them here explicitly. Here the ethical considerations focus only on the life-log data related ethics, thus I do not describe general ethics such as service quality and legal issues. User's rights, responsibilities and obligations regarding the life-log data could be listed as follows:

- The user is not mandated to keep his life-log data; he should be able to delete, edit or add content to his life-log dataset. In other words, the user should be able to change his data any time he likes without any requirement to provide a reason to the service provider.

- In the case of the misuse of a third party's information, which has been extracted from the user's dataset, the user is also legally responsible.
- Any illegal misuse of information by the user is the responsibility of the user and not the service provider.
- The user must be aware of any data manipulation policy. For instance the service provider must clarify whether the deleted data will remain in the server backups or not if the user deletes his data.
- The user should be able to enable or disable a sensor. It means that the service provider must provide enough features to let users configure easily sensors based on their preferences. For instance, the user decides to log his location and not his video on day A. The next day (day B), he decides to log both.

Based on the identified risks and studying a similar approach, It has been recommended that service providers should have the following obligations:

- Life-log data are the property of the person who creates it. It means the user is one of the owners of these data and service providers, which are bound by ethical issues, should give users the right to manipulate and manage their data. Additionally ownership of each user's data component must be clarified. Either the user is the sole owner of the data or the service provider and user both are owners. The data component can be an information block such as the user's email address or it can be sensitive information about the user such as the user's biological information.
- The user's data cannot be manipulated by the service provider without the user's consent. Except in the case where the user intends to commit fraud, misconduct or any other activities which break the agreement between the user and the service provider.
- The service provider should clarify its rights in using the user's data, and the user can accept or reject them during the service registration process. I suggest that the service provider should not be allowed to copy or transfer

the user's private data to a third party without the user's permission. If the service provider intends to keep the right to give users' data to third parties, I suggest that the service provider informs the user during the service registration process.

- The service provider must provide counter technologies such as encrypting user's information, providing authorization and authentication, etc. to block unauthorized surveillance of the user. Collins et al. [57] suggested that the service provider should perform Publicity Tests which are tests designed to reduce the risks and check the desired functionality of the system before releasing it to the public.
- The service provider must define its responsibilities in the case of data corruption, eavesdropping or any form of data loss or data misuse.
- It would be better if the service provider informed users about third parties who intend to use their data such as scientific institutions, security agencies, advertising firms, etc. User awareness is not required in every case, e.g. observing visitors in a street, since it does not intrude on any individual's privacy, is not a problem. Furthermore the service provider should clarify which information class is intended to be shared with third parties, e.g. raw information from users' life-logs, the analysis result of users' life-logs, users' registration information etc.
- The service provider should clarify whether or not, after the user's death, his information can be inherited or not. If yes, by whom? If not, what is the service provider's policy on the user data?

Kietzmann and Angell [116] described the panopticon in our society and from the first generation we are moving toward the second generation panopticon. First generation include surveillance camera, speed cameras, etc. which focused on related set of data and precise tasks. However second generation panopticons include computer supported surveillance technologies which are weaved into our daily lives. Applying these ethics or similar ethics might assist users in not realizing these digital services as panopticon.

In social communities some issues are not clear, such as data ownership after the user's death. To my knowledge Legacy Locker [3], FaceBook's If I Die [4] and DocuBank [5] are efforts to manage the problem of online data after the death of the owner. Some social networking systems have a solution for the deceased user, e.g. Facebook follows a family's wishes to take down a deceased user's profile or keep it in a memorial state' [7]. Another example to consider is a user who shares information objects that include the appearance of others (Scheiner named this information incidental data' in SNS), e.g. a photo or video of a social event. The question is: Are the others who appear in that information object owners of that information object or not? What is the service provider's policy regarding these objects? Facebook lets others remove their annotation in a photo. This means that the annotation belongs to the owner of the photo and those who appear in a photo. But the photo belongs to the owner, which sounds reasonable but it is still controversial. How harmful would a stolen life-log dataset be for the user ? What is the impact of a life-log on a society? These are some open questions which will be answered when a commercial system is available and operational.

[3] http://legacylocker.com

[4] http://www.facebook.com/IFiDieApp

[5] http://www.docubank.com

Chapter 6

Conclusion and Future Work

Recording personal or community experiences has a longstanding history in humankind. Our ancestors tried to increase their memory capacity by employing external storage devices and recording their experiences. For instance the use of external storage began with painting on stones, then writing was developed, next paper, and now we are in the digital era. In particular using these external devices assists humans in 1) preserving knowledge and 2) sharing knowledge, which is the most important requirement of human development.

The paradigm of life-logs employs external digital devices to assist humans in augmenting their memory. The memory augmentation will be achieved through sensing and recording users' contextual information continuously, which creates an electronic memory. Furthermore, as has been described before, sharing memory has many advantages for both user and society.

This dissertation introduced a novel, holistic and multipurpose framework for life logging. Terms holistic and multipurpose have been used to emphasize that the design was not based on specific use cases e.g. health monitoring. The focus is on the software engineering aspects of life logging and methods of tackling security and privacy related issues. Moreover this framework tries to resolve two necessary characteristics of our memory (sharing and forgetting) in digital memory.

The framework contains a conceptual definition and a technical definition. The conceptual definition includes a sensor classification, a flexible data model which supports sharing and forgetting and defines a fine-grained access limitation on each data object. Technical architecture starts by defining design challenges and considerations during the implementation of such a life-log. These design challenges should be considered in any life-log system regardless of their use. Next the client-server architecture has been defined and each of their components have been explained in detail. At the end I have proposed two implemented prototypes which are based on the proposed framework and they were used as a "proof of concept" for the framework.

Since life-logs are very sensitive from a privacy perspective, I have tried to resolve the privacy and security issues by 1) considering the security during the software architecture design, 2) enabling users to have control over their private information and 3) proposing ethics for life-log service providers and users as service consumers. Three different approaches are worth further investigation and they can be assumed as future work for this research. I list them as follows:

- *Sharing Personal Data Streams:*
  As has been described before a life-log dataset is a data stream. Its structure is different from a usual database and unlike those databases their records are not static [97]. Timestamp plays an important role in data streams and their records usually contain a timestamp field or a date related field. However as has been described before available SNSs are not capable hosting data streams. To my knowledge there is no research for a generic and holistic model for sharing data streams and existing sharing models are designed for specific types of data e.g. patchube [1], which is designed to share sensor network data and focus on the "Internet of Things". Some of our personal information could be interpreted as data streams such biological information and there are many tools available on the market that enable users to quantify those information objects such as FitBit and Zeo [2]. However there is no generic platform that enables users to share and combine their information from different sources. There is a potential research topic to investigate and identify requirements and challenges of such a system. This system hosts data streams from different information sources and enables users to share information. It is notable that

[1] https://pachube.com

[2] http://www.myzeo.com

such a system should not be restricted to personal information and it should be capable of hosting any data stream.

Such a system is a significant step toward open data, with respect to privacy, and more personal information will be shared than before. Furthermore it could benefit users and society in sustainability and health related issues. For example in order to augment users transportation, a system could analyze users' location changes and their daily commute. The result of the analysis could reduce transportation overheads (sustainability). Another example might be tracking and analyzing users' biological information, and thus identifying patterns of epidemic distribution.

- *Behavior Identification Through Mobile Sensing:*

  The result of this research is a set of life-log tools (a context sensing tool and a server for hosting information). Another research potential is to concentrate on studying reflection methods on the life-log dataset. As has been stated before pervasive devices e.g. mobile phone are capable of collecting valuable information about users' surrounding context. There are several initial efforts that concentrate on behavior learning [147] or mobile health [10, 1] by using life logging and mobile sensing.

  A rule engine can be designed to enable users to define simple rules and notify or report them about their activities and behaviors. Combining the rule engine and encouragement methods could persuade users to change their behavior. For instance a game, based on a life-log, could be designed to monitor users sporting activities and motivate them to compete in a team (in a social network). The more a user exercises, the more she gains rewards in the system. Another example might be a game, which is based on the rule engine, and can be used to persuade users to attend the library more and go to the pub less (via location sensor).

  Another interesting research area is using life-logs for recommendation systems and employing life-log data for online marketing. As has been described before, the concept of online marketing is based on analyzing past user activities and providing them with related advertisements. Recent online marketing has been based on desktop activities and online traces of users. However life-logs can

extend this feature to contextual information and daily life information. As a result a more precise advertisements could be delivered to users.
Moreover new behavioral patterns can be identified through spatial or temporal analysis of the life-log data. For instance the system can check if visiting specific places or persons has any influence on users' heart beat rate or not.

- *Implementation of Real Memory Functions in Electronic Memory:*
  The paradigm of life logging promises an electronic memory which works in synergy with human memory and assists us. Our memory operates based on some principles such as forgetting, generalizing, emotions, etc. Ideally an electronic memory should be able to support real memory functions. For instance the Gestalt theory [118] states that "the whole is greater than some of the parts" and human visual perception can be described with Gestalt principles. It is possible to employ such theories for resolving life-log challenges. In this scenario Gestalt can be used to challenge sensor data loss. In a more technical sense, in some situations sensors may not be available and can not read the context. Then we can use gestalt patterns to estimate the lost information object and add it to the dataset.
  For instance consider a scenario in which a user forgets to turn on her location sensor. Then she realizes that information has been lost and she would like to have this information back. The system can use a combination of Bayesian network [220] and Gestalt to examine the users' dataset and based on her previous activities estimate the lost information object. Or when a user enters in an indoor environment there is no GPS signal until she comes out from that location, then the GPS signal will be available again. During that time when there is no GPS signal, the system can estimate that the user is in the nearest indoor location and log that indoor location (with a flag that indicates it is not the original data from the sensor).

Appendix A

Benchmarking Mobile Application Resource Usage

As has been described in the proof of concept of the "Framework Description" chapter, we have developed a tool which is designed for benchmarking mobile application resource usage [175, 178]. It is not part of the framework and thus in this appendix I will explain it in detail.

According to studies, done by Compass Intelligence, companies in United States will have spent $11.6 billion on mobile applications by 2012 [45]. This indicates a significant increase in the number of mobile applications from both quantity and quality point of view. Many duplicate applications with similar functionalities and features end up on the market.

Mobile phones are subsets of pervasive devices and like other pervasive devices, they have finite energy sources [193]. Pervasive devices also suffer from client thickness [194]. This means that there will always be the challenge of willingness to increase the quality of an application while dealing with the shortage of available resources. Developers and researchers try to handle resource shortages of the pervasive devices in different ways, such as studying the context and adapting the device to the current context [23], optimizing energy usages with a CPU scheduler [240] and so forth.

This information shows that the resource usage of an application is an important factor for mobile devices which could affect the application's quality. Resource utilization is one of the performance metrics. Among other performance metrics

include application response time, throughput, reliability and availability [110].
For example consider a scenario where a user intends to purchase an audio player for his smart phone. There are many different choices on the market, "Music player X" is an audio player which plays user desired audio format like MP3 and gets some information about the current music track from the Internet. "Music player Y" is another audio player, which does not only play the desired audio format and gets the information from the Internet but also sends the audio track name to the micro-blog account (e.g. twitter) of the user. These features might be attractive for some users, but wireless network bandwidth is currently limited and expensive. A quality operator can study which application requires less network connection as a capability fact and decides upon choosing the appropriate application. Large scale industrial mobile device producers, who are interested in purchasing applications from third parties and embed them into their devices, can benefit from studying resource usage of the applications. In this scale, small amounts of disk space or memory allocation play an important role.
Here we focus on measuring resource usage of applications via a monitoring tool, and the benchmarking capability of the target application from the resource consumption point of view. Qualitative features like user interface design or application features are not within the scope of this research. We provide a monitoring tool that tracks resource usages of the device when the target application is running. It generates a trace from the resource usage. This trace can be used to study capabilities of the application or can be used for studying QoS issues. Our monitoring approach does not require any information about the target application. It resides on the same mobile device and monitors resources during the execution of the target application. In order to be flexible and scalable, we designed an approach that has no dependencies to the target application.

A.1 Resource Classification

As a first step, the resources of the mobile devices which are worth measuring and affect the capability of the application, must be identified. Each mobile application consumes *CPU*, *memory*, *battery* and may perform *disk* and *network* activities.

These are the base resources that should be measured.
It is notable that these resources can influence each other, for example CPU utilization affects battery consumption, but we intend to study them separately regardless of their interdependencies. Currently there is no standard model to prioritize mobile resources. According to [119] battery, which is responsible for the device power, is known as the most important resource for mobile phones.

A.2 Controversies and Restrictions

Monitoring can be used to dynamically manage QoS [51]. We made a resource utilization monitoring prototype to track the resource usage of mobile applications. A quality operator or application evaluator can set weights for prioritizing the importance of the measured resources, and subsequently evaluates application resource utilization via a *utility function.* A utility function enables them to compare different applications with the same functionality via profiling the resource usages for each application. We provide a utility function which helps quality operators to find out how different mobile applications with similar functionalities differ in their resource usages. This will be explained in detail in the next section.
Evaluating end user quality facts like usability or the execution time of the application is not in the scope of this research. Only application resource consumption as an important performance metric is measured here. Despite the fact that they are worth to be considered, this will not be supported by our monitoring approach because our tool resides separately from the application and will not have any interaction with the target application. Separation of the monitoring tool from the target application behavior or architecture makes our approach flexible and scalable but it also puts a restriction on measuring end user quality factors of the target application. Most end user quality metrics require quality evaluators to delve deep into the usage or the architecture of the target application.
We profile application resource usage via tracking the usage of resources of the target application or for the whole device. It could be argued that overall resource usage of the device should be considered during the target application monitoring, otherwise target application resource consumption is calculated with an overhead, which comes

from two sources, one for the general device resource usage overhead, and one for the monitoring application overhead, which introduces a systematic measurement error Δ_i for resource i. If the user intends to measure precisely the resource utilization of the target application A without this error, she can execute the monitoring tool without starting the target application, then record the consumption of resource i and compute an estimate for Δ_i, then execute both monitoring tool and target application and derive the total consumption $R_i(A)$ under the same condition. The true resource consumption $R_i^{true}(A)$ of the application can then be estimated by

$$R_i^{true}(A) = R_i(A) - \Delta_i. \tag{A.1}$$

It is important to note that there is no guarantee that the returned numbers describe precisely the resource utilization in that condition. In each of our experiments, numbers have slightly changed, hence in any calculation a percentage of error must be considered. Running the experiments for a couple of times and taking averages is thus mandatory. A special problem is given by the fact that due to random fluctuations, for a resource that is not used at all by some applications, $R_i^{true}(A)$ could actually be smaller than zero. We take care of this problem by using $\max\{R_i^{true}(A), 0\}$ instead of $R_i^{true}(A)$.

A.3 Benchmarking Applications via a Utility Function

In this work resource consumption is described by metric data and not categorical data. Furthermore, we ignore the relation between performance indicators, for example network activity is not related to the disk activity. To be able to calculate capabilities of applications based on the resource utilization, we need to analyze them together and make sure that both of these data have the same format. In order to fulfill this requirement, our benchmarking approach calculates the sum of the scores for each of the resources. When users intend to compare two applications, the conditions of the experiments should always be the same. The conditions are composed of different components like device state, experiment duration in time, number of inputs issued to the target application etc.

A.3.1 Utility Function

According to Walsh et al. [233], utility functions allow indicating the degree of desirability of a service, therefore we chose a utility function to represent the outcome of our benchmark. A utility function maps the resource usage of the application to a numerical degree of satisfaction. This utility function provides a multidimensional mapping from consumption measurements from n resources [208] to one single utility value. In this mapping, the importance of resource i is represented by a weighting factor W_i. We propose to set W_i to 0 in case the resource is not important. When the resource is considered low priority, W_i can be set to 1, and when it has high priority, it can be set to 2. A real example of using the utility function execution will be described in next sections. Other factors like "execution time of an operation" could also be considered using this utility function, but they are currently not supported by our monitoring tool. In the following, $0 \leq U_i(A) \leq 1$ defines the normalized utility of resource (UR) i under condition c, with systematic error removed due to (A.1). Then the utility $U(A)$ of application A under condition c is defined by

$$U(A) = \frac{\sum\limits_{i=1}^{n} W_i \times U_i(A)}{\sum\limits_{i=1}^{n} W_i}, \quad 0 \leq U(A) \leq 1. \tag{A.2}$$

The utility function thus results in a value between 0 and 1. The higher the value, the higher is the utility of the application.

Utility of a Resource

One problem we face is to make the consumption of different resource types comparable with each other, in order to use them in one formula. Therefore, we seek a $0 \leq U_i(A) \leq 1$, i.e., the normalized utility of resource i for application A, which should be a function of the respective consumption of resource i. We start by mapping the resource consumptions onto the interval $[0, 1]$. The resources battery and

CPU consumptions are represented via percentages. These can be mapped to the interval $[0, 1]$ in a natural way. The size of memory is always limited. For example Android emulator SDK 1.1 shows about 94572 KB total memory, with out running the target application it uses about 61296 KB of memory. Normalizing the usage of memory is thus done by dividing the memory used by the application by the total amount of memory that could be used by the application which is about 23296 KB. This value will use as total amount of available memory in our calculation as shown in Table A.2 . Note that this is the total amount of memory minus the amount of memory used by the operating system, running system tasks, and the monitoring tool. For instance we assume that during the benchmark run, resource i is sampled K times for application A, resulting in the values $0 \leq R_{ij}(A) \leq 1,\ 1 \leq j \leq K$, and the respective arithmetic mean $\bar{R}_i(A)$. Furthermore let Δ_i be an estimate for the systematic error as described in (A.1), also being mapped to the interval $[0, 1]$. The normalized utility $U_i(A)$ of resource i for the battery is then defined by:

$$U_i(A) = 1 - \max\left\{\bar{R}_i(A) - \Delta_i, 0\right\}. \tag{A.3}$$

Resource usage of CPU and Memory can be measured disregard to the monitoring tool overhead. It means Δ is zero for CPU and memory.

On the other hand, the number of disk and network activities potentially could be very high and are only limited by the total network and internal bus bandwidth, and the bandwidth and access times of the disk. In practice, most applications will use only a small fraction of the possible maxima. In order to derive a mapping into the interval $[0, 1]$ we specify, that the highest utility, i.e., the value 1, should be achieved in case the resource is not used at all. Likewise, the higher the observed numbers are, the smaller the utility should be. For deriving a stable utility of application A and resource i (including disk and network), we thus propose to run the application several times, and observe the measurement values $R_{ij}(A)$ for each run j, with arithmetic mean $\bar{R}_i(A)$. Again, let Δ_i denote the systematic error, this time in the original value range (not normalized). Then the utility for resource i is defined by

$$U_i(A) = \left(\max\left\{\bar{R}_i(A) - \Delta_i, 0\right\} + 1\right)^{-\alpha_i}. \tag{A.4}$$

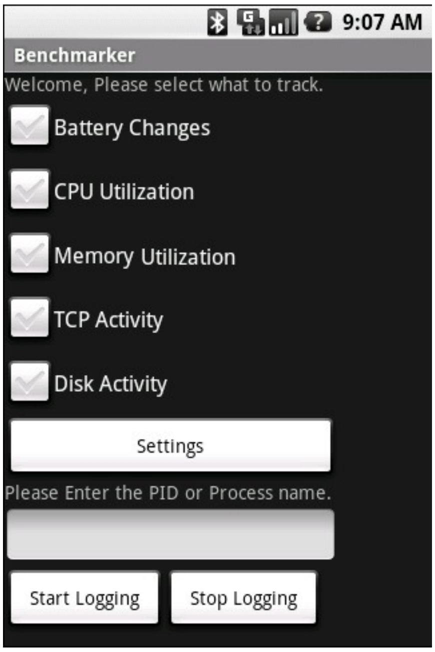
This definition depends on constants α_i reflecting how fast the utility approaches zero for resource i, and is influenced by the possible value range of observations. For instance, it reflects whether the networking activity is reported in bytes, KB, MB etc. This way, the benchmarking results of the applications are normalized, combine scores from different normalized resources, but on the other hand, the utility of one application does not depend on the measurement results of another application. What should be kept fixed, though, when comparing applications with each other, is the device that is used for measuring. One possible problem is given by the fact that (A.3) is a linear function, while (A.4) decreases with geometric speed. However, we think that if a resource is used in such a high degree that it is almost fully utilized, the resulting utility should be near zero anyway, which is achieved by both functions. The influence of this highly utilized resource then in (A.2) disappears, and (A.2) more or less reflects the utility of the other resources only.

Weights

Applications vary from their resource usages based on their functionality. For example a music player does not require a network connection or it has minimal network activity but it consumes a significant amount of battery usage. The reason is that listening to music generally runs significantly longer than for instance checking emails. On the other hand a social-networking application requires heavy network activities but consumes less battery than the music player. These applications are totally different from their resource usage perspective. Assigning priority or weight to the resources can define the level of importance in a numerical value, e.g., in the case of music player, battery gets high priority and network activity gets zero or low priority, while weights must be assigned by the user (quality operator) based on the application functionality, for example in a music player application battery has more weight than the network or in a social-networking client's battery is not important while network bandwidth usage is considered important. Users must classify applications based on their type (music player, social-networking, games, etc.) and assign to a weight.

A.3.2 Resource Monitoring Methods

Android has been chosen as the implementation platform. It is an open source mobile operating system, which is based on the Linux kernel. We have developed a resource-monitoring tool which tracks and logs the resource consumption of a given process. This monitoring tool will run in the background parallel to the target application via using Android services, It uses a GUI to let the user start and stop monitoring processes (shown in Figure A.1). A user can also select which resource(s) to track.



Welcome, Please select what to track.

☑ Battery Changes

☑ CPU Utilization

☑ Memory Utilization

☑ TCP Activity

☑ Disk Activity

Please Enter the PID or Process name.

Figure A.1: Resource monitoring tool

The user can set sampling interval via the Settings button. The sampling interval can be set for the battery, the CPU and the memory. Network and disk activities are sampled only twice, once at the start, while the start button is pressed and another time when the stop button is pressed. Unlike the CPU and the memory, they are not measured per process. Battery usage is mainly the hidden resource and it is difficult to be measured. Fortunately in Android it is possible to measure battery by creating a broadcast receiver and call it via an intent, which contains

ACTION_BATTERY_CHANGED as the action. Other resources have been measured via reading */proc* subfolders.

Sampling is done based on the predefined intervals. The sampling interval depends on the resource type, e.g. sampling interval for battery will be done each 10 minutes but CPU utilization could be sampled each second. As it has been explained before, monitoring an application itself consumes resources. This indicates that the sampling result is a mixture of monitoring application and the target application. To increase quality of the monitoring result, the user can monitor the target application with same condition more than once and calculate the arithmetic mean of the observations.

The monitoring tool runs in the background as a set of services (Android services). Each service is responsible to log one resource. The result of the monitoring is a trace file and below is an example of it:

```
[29 Mar 2009 16:17:30 GMT][GUI] START LOGGING RESOURCES
[29 Mar 2009 16:17:30 GMT][NET][Receive:487, Send:405]
[29 Mar 2009 16:17:30 GMT][DSK][Read issued:9, Write Completed:1]
[29 Mar 2009 16:17:33 GMT][CPU][PID:217, CPU:1%, #Thread:6]
[29 Mar 2009 16:17:33 GMT][MEM][PID:217,VmSize: 115084kB, VmRSS:22760kB]
[29 Mar 2009 16:17:37 GMT][CPU][PID:217, CPU:2 %, #Thread:6]
[29 Mar 2009 16:17:37 GMT][MEM][PID:217,VmSize: 115124kB, VmRSS:22852kB]
[29 Mar 2009 16:24:15 GMT][MEM][PID:217,VmSize: 118696kB, VmRSS:24992kB]
[29 Mar 2009 16:24:18 GMT][BATT][The phone's battery is charging,  %50]
[29 Mar 2009 16:24:22 GMT][CPU][PID:217, CPU:25 %, #Thread:9]
[29 Mar 2009 16:27:34 GMT][MEM][PID:217,VmSize: 117360kB, VmRSS:25172kB]
[29 Mar 2009 16:27:36 GMT][NET][Receive:25088, Send:21161]
[29 Mar 2009 16:27:36 GMT][DSK][Read issued:10, Write Completed:86]
[29 Mar 2009 16:27:36 GMT][GUI] LOGGING RESOURCES HAVE BEEN STOPED
```

A.4 Evaluation and Verification

To evaluate this model, we monitored resource utilization in the same condition for two Sudoku games [1] via the proposed monitoring tool. Our test environment is an Apple Mac Book Pro with 2.4_GHz CPU which contains Android emulator SDK 1.1. We repeated the test four times. At the end we chose the largest dataset for the analysis. The arithmetic means of the measurements in each observation do not

[1] two freeware Suduko games have been downloaded from the Internet

show significant differences. We pruned the dataset to remove near zero CPU and memory usages from before and after Monkey user simulation start. Monkey [8] is a user simulator feature of the Android platform, that generates pseudo-random streams of the user events like click, tap the screen, etc.

To evaluate if we have enough samples, we found and removed outliers. Afterwards we test the normality of data with Quantile-Quantile plot [41] from each dataset as shown in Figure A.2. Shapiro-Wilk test [205] used to test the normality of data. These datasets passed the Shapiro-Wilk test for CPU and not for memory. The statistical error level for both the CPU and the memory of both applications had been calculated for 95% confidence interval and they were acceptable. As a stop criteria we used an estimate for the standard error of the mean. The result shows that the magnitude of standard errors is much smaller than the mean. The arithmetic mean for CPU and total memory allocated for application (vmsize) have been used to calculate the utility function (Table A.1).

The monitoring tool created an application trace from the CPU and memory usage. Both applications had been tested via the Monkey. We named these Sudoku games “Application A” and “Application B”. As it has been noted before, testing application in the same condition (same test duration, same device state, etc.) is very important. In order to adhere to the same condition for each test, we used the new android emulator instance with an empty 1 Gigabyte SD card for each experiment and the same experiment duration. First the Monkey simulator had been started. The result will be used by the utility function to evaluate these two applications and compare them. Figure A.3 shows the CPU usage for approximately the same number of samples (about 80) collected from the CPU traces of these applications. In this experiment CPU sampling has been done based on three seconds intervals. There are some zero or near zero CPU usages, at the beginning and at the end of the monitoring dataset. This is because the Monkey user simulator had been called after the monitoring process has been started and after the Monkey application has been stopped the monitoring application is still running.

To measure memory utilization, the monitoring tool logs “total memory size allocated for the target application” (vmsize) and the “resident set size” (vmrss) in a three second interval for each application. Vmrss and vmsize for each application have shown in Figure A.4. We use only vmsize for the utility function.

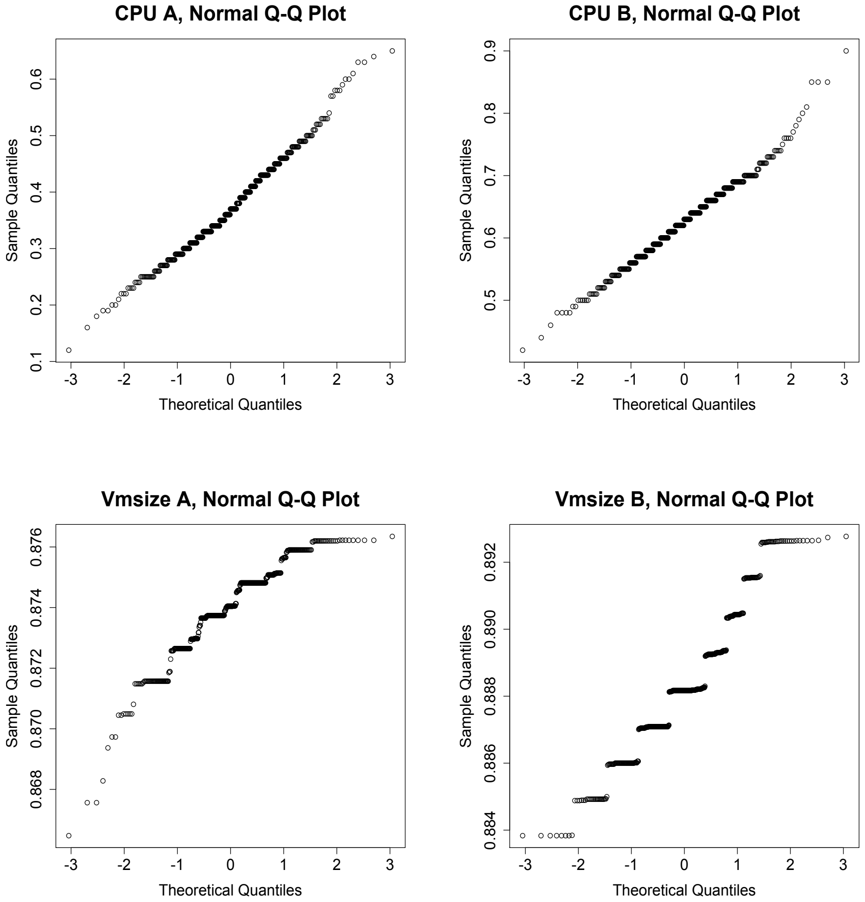


Figure A.2: Q-Q Plots for CPUs and VmSizes after removing outliers.

Table A.1 shows Utility of Resources, weights and their associated values. Table A.2 shows the result of the calculation and the utility function result for each application. Result of the utility function calculation like the result is shown in Figure A.3 and Figure A.4.

Results indicate that Application B consumes less CPU and memory than Application A. Application state could also affect utility function's error. In the Android platform components of the application have different states which these states represent application's life cycle [2]. For example if an application is visible to the user and it has the focus (start state), then more CPU could be assigned to it, or when another Android activity is resumed (pause state), then less CPU is assigned. The

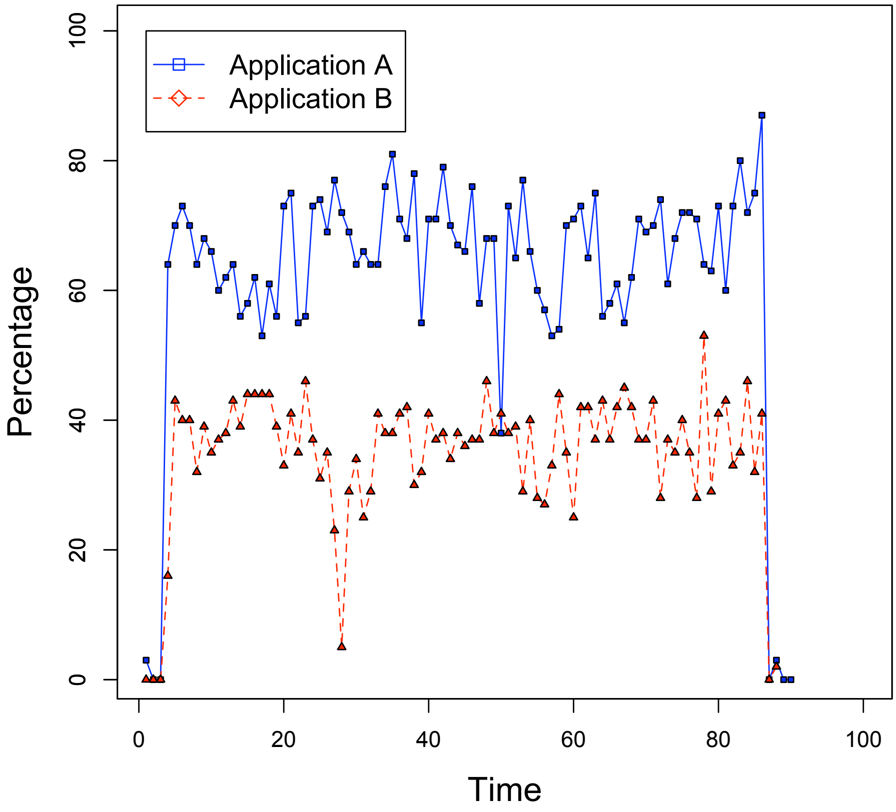


Figure A.3: CPU utilization trace.

same approach could be also done for the battery, the memory, the disk and the network. For example, if the application is visible to the user, then battery consumption will increase, if the application is suspended and another application get the GUI focus (pause), then battery consumption will decrease.

Results show that Application A uses more resources than Application B. To summarize this research a software monitoring approach to measure mobile resources and profile application's resource utilization has been proposed. It is a light software monitoring approach, which consumes few resources. It supports flexibility and modularity by separating resource monitoring from the target application. Implementation of a monitoring tool had been done on the Android platform. We have

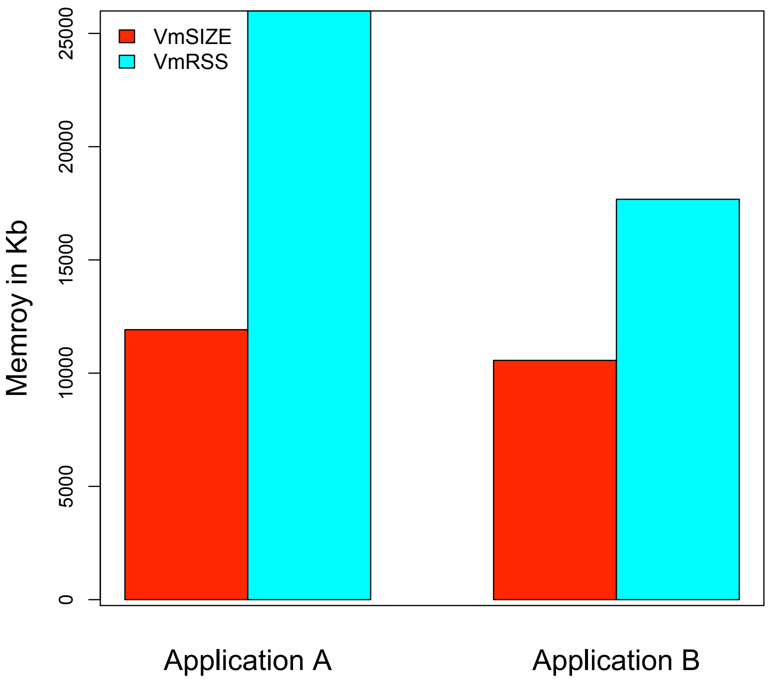


Figure A.4: Average memory usage

chosen Android because it provides access to the lower layers of operating systems such as battery usage (in easy manner) or “proc” sub folders of the Linux which can assist users to access current state of the Linux kernel.

To benchmark applications from their resource usage perspective, a utility function has been proposed. Its activity is based on the weight of the resource and average resource usage consumption. Resources data is a heterogeneous data and in order to be able to use them together, they have to be converted into a numerical data by average resource consumption and weight. Weight represents the level of the importance of each resource and will be set by the quality operator based on the application type. The UR is a normalized arithmetic mean of resource usage during the simulation.

Resource	Weight	Values for application A	Value for application B
CPU	2	62%	37%
Memory	1	vmSize=11919, vmRSS=25991	vmSize=10563, vmRSS=17657
Disk	0	-	-
Network	0	-	-
Battery	0	-	-

Table A.1: Trace values for use in utility function

	Application A		Application B	
Weight	2(CPU)	1(Memory)	2(CPU)	1(Memory)
Utility of Resource	62% => 1- 62/100	vmsize=11919 => 1-(11919/23296)	37% => 1- 37/100	vmsize=10563 => 1-(10563/23296)
Utility Function	$\frac{(2\times0.38)+(1\times0.4884)}{2+1}$		$\frac{(2\times0.63)+(1\times0.5466)}{2+1}$	
Result	0.4161		0.6022	

Table A.2: Utility function (as described before 23296 is the available amount of memory)

Appendix B

Abbreviations

A-GPS Assisted GPS

API Application Programming Interface

CAELA Communications Assistance for Law Enforcement

DAG Directed Acyclic Graph

e-Memory Electronic Memory

FIPs Fair Information Practices

FOAF Friend of a Friend (http://www.foaf-project.org)

GPS Global Positioning Service

GUI Graphical User Interface

HIPAA Health Insurance Portability and Accountability Act

IDE Integrated Development Environment

JSON Java Script Object Notation

LiDSec Light Weight Data Security

Memex Memory Extender

P2P peer-to-peer

PIM Personal Information Management

QoS Quality of Service

RDBMS Relational Database Management Systems

RDF Resource Description Framework

RSM Reliable Storage Media

SD Card Secure Digital Memory Card

SIFT Scale-invariant feature transform

SNS Social Networking Sites

SPARQL Protocol And RDF Query Language

UbiqLog Ubiquitous Logging

UR Utility of a Resource

vmSize Virtual Memory Size

vmRSS Virtual Memory Resident Set Size

WORN Write Once Read Never

Appendix C

Implementation Resources

A CD-ROM has been enclosed, which includes implementations of the framework components. The content of the CD-ROM is as follows:

- Server codes reside in the "`/Server/UbiqShare`" folder. The Server codes were written in Ruby 1.8 and Java scripts (JQuery). these codes should be executed in a Diaspora instance.
- UbiqLog codes reside in the "`/DataReader/UbiqLog`" folder. UbiqLog codes are written in Java for the Android 2 platform.
- Codes for the room tidiness metaphor reside in the "`/DataReader/RoomTidiness`" folder. Its codes were written by MATLAB Image Processing Toolbox release 2010b.
- Codes for the LiDSec reside in the "`/Security`" folder. All codes for LiDSec have been written with Java 6.
- Benchmarking Mobile Application codes reside in the "`/Benchmark`" folder. These codes were written in Java for Android 1,2 platform. The "`/R`" folder hosts files and commands for statistical analysis in R.

In order to execute the codes each Android applications has their associated .apk files and the Java project has its JAR file. The MATLAB execution has been described in its "`readme.txt`" file. The server codes should be deployed in a Diaspora instance.

Bibliography

[1] Cell Phone Intervention Trial for Young Adults (CITY). http://www.ccs.neu.edu/home/intille/positions/city.html. [Access 10 Feb 2012].

[2] Component Lifecycles, Application Fundamentals, Android Developer Guide. http://dev.android.com/guide/topics/fundamentals.html. [Access 15 April 2010].

[3] Fitbit. http://www.fitbit.com. [Accessed 30 Feb 2011].

[4] Nokia lifeblog. http://wayback.archive.org/web/*/http://www.nokia.com/lifeblog. [Accessed 10-July-2011].

[5] Q-sensor. http://www.affectiva.com/q-sensor. [Accessed 30 Feb 2011].

[6] Slife. http://www.slifeweb.com. [Accessed 25 Aug 2011].

[7] Tools for Managing Your Online Life After Death. http://www.time.com/time/specials/packages/completelist/0,29569,1920156,00.html. [Accessed 1-Aug-2011].

[8] UI/Application Exerciser Monkey, Android Developer Guide". http://dev.android.com/guide/developing/tools/monkey.html. [Access 25 April 2010].

[9] wakoopa. http://wakoopa.com. [Accessed 25 Aug 2011].

[10] Wockets project. http://www.ccs.neu.edu/home/intille/positions/wockets.html. [Access 10 Feb 2012].

[11] Zeitgeist. http://zeitgeist-project.com. [Accessed 25 Aug 2011].

[12] CALEA. http://www.fcc.gov/calea, 2005. [Accessed 1 June 2011].

[13] Directive 2006/24/EC of the European Parliament and of the Council. *Official Journal of European Union*, March 2006.

[14] The Web's New Gold Mine: Your Secrets. http://online.wsj.com/article/SB10001424052748703940904575395073512989404.html, 2010. [Access 30 July 2010].

[15] The Commission on Preservation and Access and the Research Libraries Group, Washington D.C. . *Preserving Digital Information. Report of the Task Force on Archiving of Digital Information.* 1996.

[16] Gregory D. Abowd and Elizabeth D. Mynatt. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):29–58, 2000.

[17] Eytan Adar, David Karger, and Lynn Andrea Stein. Haystack:Per-User Information Environments. In *Proceedings of the Eighth International Conference on Information and Knowledge Management (CIKM 1999)*, pages 413–422, 1999.

[18] Michael E. Addis and Kelly M. Carpenter. Why, why, why?: Reason-giving and Rumination as Predictors of Response to Activation- and Insight-Oriented Treatment Rationales. *Journal of Clinical Psychology*, 55(7):881–894, 1999.

[19] Mansoor Ahmed, Hanh Huu Hoang, Muhammad Shuaib Karim, Shah Khusro, Monika Lanzenberger, Khalid Latif, Elke Michlmayr, Khabib Mustofa, Tho Manh Nguyen, Andreas Rauber, and A Min Tjoa. Semanticlife:A Framework for Managing Information of a Human Lifetime. In *the 6th International Conference on Information Integration and Web-based Applications and Services*, 2004.

[20] Kiyoharu Aizawa. Digitizing Personal Experiences: Capture and Retrieval of Life Log. In *Proceedings of the 11th International Multi-Media Modelling Conference (MMM 2005)*, pages 10–15, 2005.

[21] Neta Aizenbud-Reshef, Eran Belinsky, Michal Jacovi, David Laufer, and Vladimir Soroka. Pensieve: Augmenting human memory. In *Extended Abstracts in Proceedings of the 26th International Conference on Human Factors in Computing Systems (CHI 2008)*, pages 3231–3236, 2008.

[22] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, and M. Dennis Mickunas. Cerberus: A Context-Aware Security Scheme for Smart Spaces. In *Pervasive Computing and Communications, IEEE International Conference on*, pages 489–496, 2003.

[23] Mourad Alia, Viktor S. Wold Eide, Nearchos Paspallis, Frank Eliassen, Svein O. Hallsteinsen, and George A. Papadopoulos. A Utility-Based Adaptivity Model for Mobile Applications. *International Conference on Advanced Information Networking and Applications Workshops*, 2:556–563, 2007.

[24] Anita L. Allen. Dredging up the Past: Lifelogging, Memory, and Surveillance. *The University of Chicago Law Review*, 75(1):47–74, 2007.

[25] Michael C. Anderson and Barbara A. Spellman. On the Status of Inhibitory Mechanisms in Cognition: Memory Retrieval as a Model Case. *Psychological Review*, 102(1):68–100, 1995.

[26] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jeniffer Widom. Models and Issues in Data Stream Systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS 2002)*, pages 1–16, 2002.

[27] Alan D. Baddeley. *Human Memory: Theory and Practice*. Psychology Press Ltd, 1997.

[28] Mathias Baldauf, Schahram Dustdar, and Florian Rosenberg. A Survey on Context-Aware Systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[29] Daniel Baloi and Andrew D.F. Price. Modelling Global Risk Factors Affecting Construction Cost Performance. *International Journal of Project Management*, 21(4):261–269, 2003.

[30] Barsalou, Lawrence W. The Content and Organization of Autobiographical Memory. *Theoretical Issues in Conceptual Information Processing*, May 1985.

[31] Petros Belimpasakis, Kimmo Roimela, and Yu You. Experience Explorer: A Life-Logging Platform Based on Mobile Context Collection. In *3rd International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2009)*, pages 77–82, 2009.

[32] Gordon Bell. A Personal Digital Store. *Communication of the ACM*, 44(1):86–91, 2001.

[33] Gordon Bell and Jim Gemmell. *Total recall: How the E-memory Revolution will Change Everything*. Dutton, 2009.

[34] Gordon Bell, Jim Gemmell, and Roger Lueder. Challenges in Using Lifetime Personal Information Stores. In *Proceedings of the 27th annual international ACM conference on Research and development in information retrieval (SIGIR 2004)*, pages 1–1, 2004.

[35] Gordon Bell and Jim Gray. Digital Immorality. Technical Report MSR-TR-2000-101, Microsoft Research, October 2000.

[36] Emma Berry, Narinder Kapur, Lyndsay Williams, Steve Hodges, Peter Watson, Gavin Smyth, James Srinivasan, Reg Smith, Barbara Wilson, and Ken Wood. The use of a wearable camera, SenseCam, as a pictorial diary to improve autobiographical memory in a patient with limbic encephalitis: A preliminary report. *Neuropsychological Rehabilitation*, 17(4-5):582–601, August 2007.

[37] George Esdras Bevans. How Working Men Spend Their Spare Times. *New York: Columbia University Press*, 1913.

[38] Robert A. Bjork and Alan Richardson-Klavehn. On the puzzling relationship between environmental context and human memory. *Experimental Psychology*, 41(3), 1989.

[39] Aron F. Bobick, Stephen S. Intille, James W. Davis, Freedom Baird, Claudio S. Pinhanez, Lee W. Campbell, Yuri A. Ivanov, Arjan Schütte, and Anrew Wilson. The KidsRoom: A Perceptually-based Interactive and Immersive Story Environment. *Presence*, 8(4):369–393, 1999.

[40] Joseph Bonneau and Sren Preibusch. The Privacy Jungle: On the Market for Data Protection in Social Networks. In *Economics of Information Security and Privacy*, pages 121–167. 2010.

[41] Sarah Boslaugh and Paul Watters. *Statistics in a Nutshell: A Desktop Quick Reference*, pages 118–119. O'Reilly Media, Inc., 2008.

[42] Jack W. Brehm. *A Theory of Psychological Reactance*. Academic Press New York, 1966.

[43] John Breslin and Stefan Decker. The Future of Social Networks on the Internet: the Need for Semantics. *IEEE Internet Computing*, 11(6):86–90, 2007.

[44] Sonja Buchegger, Doris Schöberg, Le-Hung Vu, and Anwitaman Datta. PeerSoN: P2P Social Networking: Early Experiences and Insights. In *Proceedings of the Second ACM Workshop on Social Network Systems Social Network Systems 2009, co-located with Eurosys 2009*, pages 46–52, 2009.

[45] Kneko Burney. New Year's Bustle? Vertical Market Expectations for 2009 ICT Spending in the US- From Crisis to Slow, Long-term Recovery. *Compass Intelligence*, January 2009.

[46] Vannevar Bush. As We May Think. *The Atlantic*, 1945.

[47] Daragh Byrne, Barry Lavelle, Aiden R. Doherty, Gareth J.F. Jones, and Alan F. Smeaton. Using Bluetooth and GPS Metadata to Measure Event Similarity in Sensecam Images. In *World Scientific Publishing*, 2007.

[48] Yuhan Cai, Xin Luna Dong, Alon Halevy, Jing Michelle Liu, and Jayant Madhavan. Personal information management with SEMEX. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD 2005)*, pages 921–923, 2005.

[49] Scott Carlson. On the Record, All the Time. *Chronicle of Higher Education*, 53(23):1, 2007.

[50] Soumen Chakrabarti, Sunita Sarawagi, and Byron Dom. Mining Surprising Patterns Using Temporal Description Length. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 606–617, 1998.

[51] Daniel Chalmers and Morris Sloman. A Survey of Quality of Service in Mobile Computing Environments. *IEEE Communications Surveys*, 2(2), March 1999.

[52] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[53] Guanling Chen and David Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, 2000.

[54] Snehal Chennuru, Peng-Wen Chen, Jiang Zhu, and Ying Zhang. Mobile Lifelogger - recording, indexing, and understanding a mobile user's life. In *Proceedings of The Second International Conference on Mobile Computing, Applications, and Services (MobiCASE 2010)*, 2010.

[55] Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Haehnel, Beverly Harrison, Bruce Hemingway, Jeffrey Hightower, Predrag Klasnja, Karl Koscher, Anthony LaMarca, James A. Landay, Louis LeGrand, Jonathan Lester, Ali Rahimi, Adam Rea, and Danny Wyatt. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, 7(2):32–41, 2008.

[56] Brain Patrick Clarkson. *Life Patterns: Structures of wearable sensors.* PhD thesis, Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science, 2002.

[57] W. Robert Collins, Keith W. Miller, Bethany J. Spielman, and Phillipe Wherry. How Good is Good Enough?: an Ethical Analysis of Software Construction and Use. *Communications of the ACM*, 37(1):81–91, 1994.

[58] Sunny Consolvo, David W. McDonald, Tammy Toscos, Mike Y. Chen, Jon Froehlich, Beverly L. Harrison, Predrag V. Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, Ian E. Smith, and James A. Landay. Activity Sensing in the Wild: a Field Trial of Ubifit Garden. In *Proceeding of the 26th annual SIGCHI conference on Human factors in computing systems (CHI 2008)*, pages 1797–1806, 2008.

[59] Consultative Committee for Space Data Systems. CCSDS 650.0-B-1. Recommendation for Space Data System Standards. Reference Model for an Open Archival Information System (OAIS). Technical report, 2001.

[60] Michael J. Covington, Prahlad Fogla, Zhiyuan Zhan, and Mustaque Ahamad. A Context-Aware Security Architecture for Emerging Applications. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC 2002)*, pages 249–260, 2002.

[61] Stephan Davies. Still Building the Memex. *Communications of the ACM*, 54(2):80–88, 2011.

[62] Anind K. Dey and Gregory D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *In proceedings of international conference on Human*

factors in computing systems. Workshop on The What, Who, Where, When, and How of Context-Awareness (CHI 2000), volume 4, pages 1–6, 2000.

[63] Jens-Peter Dittrich. imemex: A platform for personal dataspace management. In *SIGIR PIM 2nd Invitational Workshop for Personal Information Management*, 2006.

[64] Jens-Peter Dittrich and Marcos Antonio Vaz Salles. idm: a unified and versatile data model for personal dataspace management. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006)*, pages 367–378, 2006.

[65] Jens-Peter Dittrich, Marcos Antonio Vaz Salles, Donald Kossmann, and Lukas Blunschi. iMeMex: Escapes from the Personal Information Jungle. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, pages 1306–1309, 2005.

[66] Martin Dodge and Rob Kitchin. 'Outlines of a world coming into existence': Pervasive computing and the ethics of forgetting. *Environment and Planning B: Planning and Design,*, 34(3):431–445, 2007.

[67] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. The PARCTAB mobile computing system. In *Fourth Workshop on Workstation Operating Systems*, pages 34–39, 1993.

[68] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. In *Proceedings of the 26th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2003)*, pages 72–79, 2003.

[69] Nathan Eagle. *Machine Perception and Learning Complex Social Systems.* PhD thesis, Massachusetts Institute of Technology, 2005.

[70] Nathan Eagle. Behavioral Inference across Cultures: Using Telephones as a Cultural Lens. *IEEE Intelligent Systems*, 23(4):62–64, 2008.

[71] Nathan Eagle and Alex Pentland. Social Serendipity: Mobilizing Social Software. *IEEE Pervasive Computing*, 4(2):28–34, 2005.

[72] Nathan Eagle and Alex(Sandy) Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, 2006.

[73] Nathan Eagle and Alex(Sandy) Pentland. Eigenbehaviors: Identifying Structure in Routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[74] Nathan Eagle, Alex(Sandy) Pentland, and David Lazer. Inferring Friendship Network Structure by Using Mobile Phone Data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[75] Hermann Ebbinghaus. *Memory: A Contribution to Experimental Psychology.* Teachers college, Columbia university, 1913.

[76] Douglas C. Engelbart. AUGMENTING HUMAN INTELLECT: A Conceptual Framework. Technical report, Stanford Research Institute, 1962.

[77] European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with regard to the Processing of Personal Data and on the Free Movement of such Data. *Official Journal of the European Communities*, L 281:31–50, 1995.

[78] Scott Fertig, Eric Freeman, and David Gelernter. Lifestreams: An Alternative to the Desktop Metaphor. In *Conference on Human Factors in Computing Systems Conference Companion (CHI 1996)*, pages 410–411, 1996.

[79] Ulrich Flegel. Pseudonymizing Unix log files. In *International Conference on Infrastructure Security (InfraSec 2002)*, pages 162–179, 2002.

[80] Brain J. Fogg. Persuasive Technology: Using Computers to Change what we Think and Do. *Ubiquity*, 3(44), 2002.

[81] Brain J. Fogg. Creating Persuasive Technologies: An Eight-Step Design Process. In *Proceedings of the 4th International Conference on Persuasive Technology (Persuasive 2009)*. ACM New York, NY, USA, 2009.

[82] Brian J. Fogg. *Persuasive Technology: Using Computers to Change What We Think and Do.*, page 32. Morgan Kaufmann, 2003.

[83] Michel Foucault. *Discipline and Punish: The Birth of the Prison.* Penguin Books New York, 1991.

[84] Helmar G. Frank. Überraschungswert und Auffälligkeit. *Progress in Brain Research*, 17:179–208, 1951.

[85] Michael Franklin, Alon Halevy, and David Maier. From Databases to Dataspaces: A New Abstraction for Information Management. *ACM SIGMOD Record*, 34(4):27–33, 2005.

[86] Sigmund Freud. Repression. *Standard Edition of the Complete Psychological Works of Sigmund Freud*, 14:143–158, 1915.

[87] Charles Fried. Privacy: A Moral Analysis. *Yale Law Journal*, 77, 1968.

[88] Jon Froehlich, Mike Y. Chen, Sunny Consolvo, Beverly Harrison, and James A. Landay. MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications, and Services (MobiSys 2007)*, pages 57–70, 2007.

[89] Jon. Froehlich, Tawanna Dillahunt, Predrag Klasnja, Jeniffer Mankoff, Sunny Consolvo, Beverly Harrison, and James A. Landay. UbiGreen: Investigating a Mobile Tool for Tracking and Supporting Green Transportation Habits. In *Proceedings of*

the 27th International Conference on Human Factors in Computing Systems (CHI 2009), pages 1043–1052, 2009.

[90] Hans W. Gellersen, Albercht Schmidt, and Michael Beigl. Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts. *Mobile Networks and Applications*, 7(5):341–351, 2002.

[91] Jim Gemmell, Gordon Bell, and Roger Lueder. MyLifeBits: A Personal Database for Everything. *Communication of ACM*, 49(1):88–95, 2006.

[92] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. MyLifeBits: fulfilling the Memex vision. In *Proceedings of the tenth ACM international conference on Multimedia (MULTIMEDIA 2002)*, pages 235–238, 2002.

[93] Jim Gemmell, Roger Lueder, and Gordon Bell. The mylifebits lifetime store. In *Proceedings of the ACM SIGMM workshop on Experiential telepresence (ETP 2003)*, pages 80–83, 2003.

[94] Jim Gemmell, Lyndsay Williams, Ken Wood, Roger Lueder, and Gordon Bell. Passive Capture and Ensuing Issues for a Personal Lifetime Store. In *Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences (CARPE 2004)*, pages 48–55, 2004.

[95] Vadim Gerasimov. *EVERY SIGN OF LIFE*. PhD thesis, Massachusetts Institute of Technology, 2003.

[96] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Data modeling of time-based media. In *Proceedings of the 1994 ACM international conference on Management of data (SIGMOD 1994)*, volume 23, pages 91–102, 1994.

[97] Lukasz Golab and Tamer M. ´Ozsu. Issues in Data Stream Management. *ACM Special Interest Group on Management of Data (SIGMOD) Record*, 32:5–14, 2003.

[98] Ralph Gross and Alessandro Acquisti. Information Revelation and Privacy in Online Social Networks. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[99] Cathal Gurrin, Alan F. Smeaton, Daragh Byrne, Neil O'Hare, Gareth J.F. Jones, and Noel E. O'Connor. An Examination of a Large Visual Lifelog. In *Proceedings of the 4th Asia information retrieval conference on Information retrieval technology (AIRS 2008)*, pages 537–542, 2008.

[100] Joshua Hailpern, Nicholas Jitkoff, Andrew Warr, Karrie Karahalios, Robert Sesek, and Nikita Shkrob. YouPivot: Improving Recall with Contextual Search. In *Proceedings of the 2011 annual conference on Human factors in computing systems (CHI 2011)*, pages 1521–1530, 2011.

[101] Jiawei Han, Michelene Kamber, and Jian Pei. *Data mining: concepts and techniques.* Morgan Kaufmann, 2001.

[102] Jeniffer Healey and Rosalind W. Picard. StartleCam: a Cybernetic Wearable Camera. In *Second International Symposium on Wearable Computers*, pages 42–49, 1998.

[103] Steve Hodges, Lyndsay Williams, Emma Berry, Shahram Izadi, James Srinivasan, Alex Butler, Gavin Smyth, Narinder Kapur, and Ken Wood. SenseCam: A Retrospective Memory Aid. In *Proceedings of the 8th International Conference of Ubiquitous Computing (UbiComp 2006)*, pages 177–193, 2006.

[104] Norman J. Holter and J.A. Generelli. Remote Recording of Physiological Data by Radio. *Rocky Mountain Medical Journal*, 46(9):747, 1949.

[105] Eric Horvitz. Machine Learning, Reasoning, and Intelligence in Daily Life: Directions and Challenges. In *Proceedings of the Workshop on AI Techniques for Ambient Intelligence (AITAmI)*, 2007.

[106] Jane Hunter. Collaborative Semantic Tagging and Annotation Systems. *Annual Review of Information Science and Technology, American Society for Information Science*, 43(1):187–239, 2009.

[107] Oguz İçoğlu and Ardeshir Mahdavi. VIOLAS: A Vision-based Sensing System for Sentient Building Models. *Automation in Construction*, 16(5):685–712, 2007.

[108] Jadwiga Indulska and Peter Sutton. Location Management in Pervasive Systems. In *ACSW Frontiers*, pages 143–151, 2003.

[109] Laurent Itti and Pierre Baldi. Bayesian Surprise Attracts Human Attention. *Advances in Neural Information Processing Systems*, 18:547–554, 2006.

[110] Raj K. Jain. *The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* John Wiley & Sons, 1991.

[111] Anthony Jameson, Michael Schneider, Carolin Plate, Alexander Kröner, Vania Dimitrova, Nathalie Basselin, and Stephan Baldes. Recomindation: New Functions for Augmented Memories. In *Fourth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 141–150, 2006.

[112] William Jones. Personal Information Management. *Annual Review of Information Science and Technology*, 41(1):110–111, 2007.

[113] Karin Kappel and Thomas Grechenig. ”show-me”: Water Consumption at a glance to promote Water Conservation in the Shower. In *Proceedings of the 4th International Conference on Persuasive Technology (Persuasive 2009)*, 2009.

[114] Fahim Kawsar, Kaori Fujinami, and Tatsuo Nakajima. Augmenting Everyday Life with Sentient Artefacts. In *Proceedings of the 2005 joint conference on Smart Objects and Ambient Intelligence (sOc-EUSAI 2005)*, pages 141–146, 2005.

[115] Nicky Kern, Bernt Schiele, Holger Junker, Paul Lukowicz, Gerhard Tröster, and Albrecht Schmidt. Context Annotation for a Live Life Recording. In *Workshop on Memory and Sharing of Experiences(Pervasive 2004)*, 2004.

[116] Jan Kietzmann and Ian Angell. Panopticon Revisited. *Communications of the ACM*, 53(6):135–138, 2010.

[117] Ig-Jae Kim, Sang Chul Ahn, and Hyoung-Gon Kim. Personalized Life Log Media System in Ubiquitous Environment. In *Proceedings of the first International Conference Ubiquitous Convergence Technology (ICUCT 2006)*, pages 20–29, 2006.

[118] Kurt Koffka. *Principles of Gestalt Psychology.* 1935.

[119] Robin Kravets and P. Krishnan. Power Management Techniques for Mobile Communication. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom 1998)*, pages 157–168, 1998.

[120] Alexander Kröner, Michael Schneider, and Junichiro Mori. A framework for ubiquitous content sharing. *IEEE Pervasive Computing*, 8(4):58–65, 2009.

[121] Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[122] Wheeler Ladd and Harry T. Reis. Self-Recording of Everyday Life Events: Origins, Types, and Uses. *Journal of Personality*, pages 339–354, 1991.

[123] Michael Lambert. Visualizing and Analyzing Human-centered Data Streams. Master's thesis, Massachusetts Institute of Technology. Department of Electrical Engineering and Computer Science, 2005.

[124] Mik Lamming and Mike Flynn. Forget-me-not: intimate computing in support of human memory. In *Proceedings of FRIEND21 Symposium on Next Generation Human Interfaces*, 1994.

[125] Nicolas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[126] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax. *World Wide Web Consortium, http://www.w3.org/TR/PR-rdf-syntax*, 1999.

[127] Kenneth C. Laudon. Markets and Privacy. *Communication of ACM*, 39:92–104, 1996.

[128] Hyowon Lee, Alan F. Smeaton, Noel E. OConnor, Gareth Jones, Michael Blighe, David Byrne, Aiden Doherty, and Cathal Gurrin. Constructing a SenseCam Visual Diary as a Media Process. *Multimedia Systems*, 14(6):341–349, 2008.

[129] KwangHee Lee, KyoungJu Noh, and Changseok Bae. Personal Life Logger and Belonging Monitor Using Reliable ZigBee Networks. In *Human-Computer Interaction. HCI Applications and Services*, pages 961–970. 2007.

[130] Seunghwan Lee, Geehyuk Lee, and Hojin Kim. Ubigraphy: A third-person viewpoint life log. In *extended abstracts in Proceeding of the 26th annual SIGCHI conference on Human factors in computing systems (CHI 2008)*, pages 3531–3536, 2008.

[131] Paul Lehrer, Evgeny Vaschillo, and Bronya Vaschillo. Resonant Frequency Biofeedback Training to Increase Cardiac Variability: Rationale and Manual for Training. *Applied Psychophysiology and Biofeedback*, pages 177–191, 2000.

[132] Paul Lehrer, Evgeny Vaschillo, Bronya Vaschillo, Shou-En Lu, Anthony Scardella, Mahmood Siddique, and Robert H. Habib. Biofeedback as a treatment for asthma. *Chest*, 126(2):352–361, 2004.

[133] Ian Li, Anind Dey, and Judith Forlizzi. A Stage-Based Model of Personal Informatics Systems. In *ACM 28th International Conference on Human Factors in Computing Systems (CHI 2010)*, pages 557–566, 2010.

[134] Alfred Lief. *The Commonsense Psychiatry of Dr. Adolf Meyer*. McGraw-Hill, 1948.

[135] Elizabeth F. Loftus and Wesley Marburger. Since the eruption of Mt. St. Helens, has anyone beaten you up? Improving the accuracy of retrospective reports with landmark events. *Memory & Cognition*, pages 114–120, 1983.

[136] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *IEEE International Conference on Computer Vision*, pages 1150–1159, 1999.

[137] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones. In *Proceedings of the 7th international Conference on Mobile Systems, Applications, and Services*, pages 165–178, 2009.

[138] Hong Lu, Jun Yang, Zhigang Liu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. The Jigsaw Continuous Sensing Engine for Mobile Phone Applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SensSys 2010)*, pages 71–84, 2010.

[139] Artur Lugmayr, Teemu Saarinen, and Jean-Philippe Tournut. The Digital Aura - Ambient Mobile Computer Systems. In *14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2006)*, pages 348–354, 2006.

[140] David Lyon. *Surveillance after September 11*. Polity Press, 2003.

[141] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In *Selected Areas in Cryptography*, pages 184–199, 2000.

[142] Carolyn MacCann, Angela L. Duckworth, and Richard D. Roberts. Empirical Identification of the Major Facets of Conscientiousness. *Learning and Individual Differences*, 19(4):451–458, 2009.

[143] James B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, page 14, 1967.

[144] Steve Mann. Continuous Lifelong Capture of Personal Experience with EyeTap. In *Proceedings of the the 1st ACM workshop on Continuous archival and retrieval of personal experiences, (CARPE'04)*, pages 1–21, 2004.

[145] Steve Mann, James Fung, Chris Aimone, Anurag Sehgal, and Anurag Sehgal. Designing EyeTap Digital Eyeglasses for Continuous Lifelong Capture and Sharing of Personal Experiences. In *Proceedings of the 23th international conference extended abstracts on Human factors in computing systems, (CHI 2005)*, pages 2–7, 2005.

[146] Steve Mann, Jason Nolan, and Barry Wellman. Sousveillance: Inventing and Using Wearable Computing Devices for Data Collection in Surveillance Environments. *Surveillance and Society*, 1(3):331–355, 2003.

[147] M. Mantyla. The OtaSizzle Project: Large-Scale Service Experimentation Testbed. In *Eighth International Workshop on Applications and Services in Wireless Networks, (ASWN'08)*, pages 8–8, 2008.

[148] Victor Mayer-Schoenberger. *delete: The Virtue of Forgetting in the Digital Age.* Princeton University Press Princeton, 2009.

[149] Claude H. Miller, Lindsay T. Lane, Leslie M. Deatrick, Alice M. Young, and Kimberly A. Potts. Psychological Reactance and Promotional Health Messages: The Effects of Controlling Language, Lexical Concreteness, and the Restoration of Freedom. *Human Communication Research*, 33(2):219–240, 2007.

[150] Jim Miller. Who Are You? The Trade-Off between Information Utility and Privacy. *IEEE Internet Computing*, 12(4):93–96, 2008.

[151] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys'08)*, pages 337–350, 2008.

[152] Tom M. Mitchell. Mining our Reality. *Science*, 326(5960):1644–1645, 2009.

[153] Tom Michael Mitchell. *Machine Learning.* WCB/McGraw-Hill, 1997.

[154] James H. Moor. What is computer ethics ? *Metaphilosophy*, 16(4):266–275, 1985.

[155] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services (MobiSys 2009)*, pages 55–68, 2009.

[156] Tatsu Nakajima, Vili Lehdonvirta, Eiji Tokunaga, and Hiroaki Kimura. Reflecting human behavior to motivate desirable lifestyle. In *Proceedings of the 7th ACM conference on Designing interactive systems*, pages 405–414, 2008.

[157] Jason Nawyn, Stephen S. Intille, and Kent Larson. Embedding Behavior Modification Strategies into a Consumer Electronic Device: A Case Study. In *8th International Conference on Ubiquitous Computing (UbiComp 2006)*, pages 297–314, 2006.

[158] Jakob Nielsen. Ten Usability Heuristics. `http://www.useit.com/papers/heuristic/heuristic_list.html`, 1990.

[159] Oded Nov and Sunil Wattal. Social computing privacy concerns: Antecedents and effects. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 333–336. ACM, 2009.

[160] Oded Nov and Sunil Wattal. Social Computing Privacy Concerns: Antecedents and Effects. In *Proceedings of the 27th international conference on Human factors in computing systems (CHI 2009)*, pages 333–336, 2009.

[161] Hiroko Nozawa, Takuji Narumi, Kunihiro Nishimura, and Michitaka Hirose. Beat story: life-log system of subjective time using heart beat rate. In *the 35th International Conference on Exhibition on Computer Graphics and Interactive Techniques, posters (SIGGRAPH 2008)*, pages 1–1, 2008.

[162] Ted O'Donoghue and Matthew Rabin. Self-Awareness and Self-Control. *Time and Decision: Economic and Psychological Perspectives on Intertemporal Choice*, pages 217–243, 2003.

[163] Kieron O'Hara, Mischa Tuffield, and Nigel Shadbolt. Lifelogging: Privacy and Empowerment with Memories for Life. *Identity in the Information Society*, 1(2), 2009.

[164] Jon Orwant. Heterogeneous Learning in the Doppelgänger User Modeling System. *User Modeling and User-Adapted Interaction*, pages 107–130, 1994.

[165] Manoj Parameswaran and Andrew B. Whinston. Social Computing: An Overview. *Communications of the Association for Information Systems*, 19(1):762–780, 2007.

[166] S.Tejaswi Peesapati, Victoria Schwanda, Johnathon Schultz, Matt Lepage, So-yae Jeong, and Dan Cosley. Pensieve: Supporting Everyday Reminiscence. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 2027–2036, 2010.

[167] Alex (Sandy) Pentland. Automatic mapping and modeling of human networks. *Physica A: Statistical Mechanics and its Applications*, 378(1):59–67, 2007.

[168] Alex (Sandy) Pentland. *Honest Signals: How They Shape Our World.* The MIT Press, 2008.

[169] Per Persson and Younghee Jung. Nokia Sensor: from Research to Product. In *Proceedings of the 2005 conference on Designing for User eXperience (DUX 2005)*, pages 53–75, 2005.

[170] Klaus Pommerening. Medical Requirements for Data Protection. In *Proceedings of IFIP Congress*, volume 2, pages 533–540, 1994.

[171] Brian L. Quick and Jeniffer R. Considine. Examining the Use of Forceful Language When Designing Exercise Persuasive Messages for Adults: A Test of Conceptualizing Reactance Arousal as a Two-Step Process. *Health Communication*, 23(5):483–491, 2008.

[172] Richard J. Radke, Srinivas Andra, Omar Al-Kofahi, and Badrinath Roysam. Image Change Detection Algorithms: a Systematic Survey. *Image Processing, IEEE Transactions on*, 14(3):294–307, 2005.

[173] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005.

[174] Aza Raskin. Your Life Experiences, Brought to You by Budweiser. *IEEE Security & Privacy*, pages 86–88, 2011.

[175] Reza Rawassizadeh. Mobile Application Benchmarking based on the Resource Usage Monitoring. *International Journal of Mobile Computing and Multimedia Communications*, 1:64–75, 2009.

[176] Reza Rawassizadeh. A Holistic Multipurpose Life-log Framework. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing. Doctoral Colloquium (UbiComp 2010)*, pages 481–484, 2010.

[177] Reza Rawassizadeh. Toward Sharing Life-log Information with Society. *Behaviour and Information Technology (to appear)*, 2012. `http://www.tandfonline.com/doi/abs/10.1080/0144929X.2010.510208`.

[178] Reza Rawassizadeh, Amin Anjomshoaa, and A Min Tjoa. A Qualitative Resource Utilization Benchmarking for Mobile Applications. In *Innovations in Mobile Multimedia Communications and Applications: New Technologies*, pages 149–160. IGI Global, 2011.

[179] Reza Rawassizadeh, Amin Anjomshoaa, and Martin Tomitsch. A Framework for Long-Term Archiving of Pervasive Device Information. In *The 9th International*

Conference on Advances in Mobile Computing & Multimedia (MoMM-2011)., pages 244–247, 2011.

[180] Reza Rawassizadeh, Johannes Heurix, Soheil Khosravipour, and A Min Tjoa. LiD-Sec: A Lightweight Pseudonymization Approach for Privacy-Preserving Publishing of Textual Personal Information. In *Sixth International Conference on Availability, Reliability and Security (ARES 2011). In proceedings of First International Workshop on Privacy by Design (PBD 2011)*, 2011.

[181] Reza Rawassizadeh, Soheil Khosravipour, and A Min Tjoa. A Persuasive Approach for Indoor Environment Tidiness. In *6th International Conference on Persuasive Technology (Persuasive 2011)*, 2011.

[182] Reza Rawassizadeh, Elaheh Momeni, Katarzyna Wac, and Martin Tomitsch. Supporting Forgetting and Semantic Enrichment of e-Memories through Annotation. In *ACM 29th International Conference on Human Factors in Computing Systems, In workshop on Bridging Practices, Theories, and Technologies to Support Reminiscence (CHI 2011)*, 2011.

[183] Reza Rawassizadeh and A Min Tjoa. Securing Shareable Life-logs. In *The Second IEEE International Conference on Social Computing/IEEE International Conference on Privacy, Security, Risk and Trust (PASAAT 2010). In proceedings of First Workshop on Privacy Aspects of Social Web and Cloud Computing (PASWeb 2010)*, pages 1105–1110, 2010.

[184] Reza Rawassizadeh and Martin Tomitsch. Toward Preserving Pervasive Device Information. In *ACM 28th International Conference on Human Factors in Computing Systems, In workshop on Know Thyself: Monitoring and Reflecting on Facets of One's Life (CHI 2010)*, 2010.

[185] Reza Rawassizadeh, Martin Tomitsch, Katarzyna Wac, and A Min Tjoa. UbiqLog: A Generic Mobile Phone based Life-Log Framework. *Personal and Ubiquitous Computing (to appear)*, 2012.

[186] John Rawls. *A Theory of Justice.* Harvard University Press, 1971.

[187] Bradley J. Rhodes. Context-Aware Computing (or, why context needs wearables and wearables need context) . `http://www.media.mit.edu/wearables/lizzy/context.html`.

[188] Bradley J. Rhodes. The Wearable Remembrance Agent: A System for Augmented Memory. In *Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology (PAAM 1996)*, pages 487–495, 1996.

[189] Meredith Ringel, Edward Cutrell, Susan Dumais, and Eric Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores.

In *IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003)*, pages 184–191, 2003.

[190] George Roussos, Marsh Andy J., and Stavroula Maglavera. Enabling Pervasive Computing with Smart Phones. *IEEE Pervasive Computing*, 4(2):20–27, 2005.

[191] Åsa Rudström, Martin Svensson, Rickard Cöster, and Kristina Höök. Mobitip: Using Bluetooth as a Mediator of Social Context. In *Proceedings of the 6th ACM International Conference on Ubiquitous Computing, Adjunct Proceedings (Ubicomp 2004)*, 2004.

[192] Dong-Wan Ryoo and Changseok Bae. Design of The Wearable Gadgets for Life-Log Services based on UTC. *IEEE Transactions on Consumer Electronics*, 53(4):1–6, 2007.

[193] Mahadev Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the 15th annual ACM symposium on Principles of distributed computing (PODC 1996)*, pages 1–7, 1996.

[194] Mahadev Satyanarayanan. Pervasive Computing: Vision and challenges. *IEEE Personal Communications*, 8(4):10–17, 2001.

[195] Daniel L. Schacter. The Seven Sins of Memory. *American Psychologist*, 54(3):182–203, 1999.

[196] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[197] Bill N. Schilit, Anthony LaMarca, Gaetano Borriello, William G. Griswold, David McDonald, Edward Lazowska, Anand Balachandran, Jason Hong, and Vaughn Iverson. Challenge: Ubiquitous Location-Aware Computing and the "Place Lab" Initiative. In *Proceedings of the 1st ACM international workshop on Wireless mobile applications and services on WLAN hotspots (WMASH 2003)*, pages 29–35, 2003.

[198] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, Kristof Van Laerhoven, and Walter Van de Velde. Advanced Interaction in Context. In *First International Symposium on Handheld and Ubiquitous Computing*, pages 89–101, 1999.

[199] Bruce Schneier. Taxonomy of Social Networking and Privacy. `http://cyberlaw.stanford.edu/node/6381`.

[200] Michael Scriven. Beyond Formative and Summative Evaluation. *Evaluation and Education at Quarter Century*, pages 19–64, 1991.

[201] Abigail J. Sellen, Andrew Fogg, Mike Aitken, Steve Hodges, Carsten Rother, and Ken Wood. Do Life-Logging Technologies Support Memory for the Past? An Experimental Study Using SenseCam. In *Proceedings of the 25th conference on Human Factors in Computing Systems (CHI 2007)*, pages 81–90, 2007.

[202] Abigail J. Sellen and Steve Whittaker. Beyond Total Capture: A Constructive Critique of Lifelogging. *Communications of the ACM*, 53(5):70–77, 2010.

[203] Noah Shachtman. A Spy Machine of DARPAs Dreams. `http://www.wired.com/print/techbiz/media/news/2003/05/58909`, 2003.

[204] Claude E. Shannon. Prediction and Entropy. *Bell Systems Technical Journal*, 30:50–64, 1951.

[205] S.S. Shapiro, M.B. Wilk, and H.J. Chen. A Comparative Study of Various Tests for Normality. *American Statistical Association*, 63(324):134–137, 1968.

[206] Sarah Sitton. The Messy Desk Effect: How Tidiness Affects the Perception of Others. *Journal of Psychology: Interdisciplinary and Applied*, 117(2):263–267, 1984.

[207] Burrhus F. Skinner. Are Theories of Learning Necessary?, 1950.

[208] James E. Smith. Characterizing Computer Performance with a Single Number. *Communication of ACM*, 31(10):1202–1206, 1988.

[209] Richard M. Soley. Memex is not Enough. In *Third International Conference on Fundamental Approaches to Software Engineering (FASE 2000)*, 2000.

[210] Daniel J. Solove. Conceptualizing Privacy. *California Law Review*, 90:1087–1155, 2002.

[211] Meesun Song, Wonkyu Lee, and Kim Junghwan. Extraction and Visualization of Implicit Social Relations on Social Networking Services. In *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 1425–1430, 2010.

[212] Anna Ståhl, Kristina H´o´ok, Martin Svensson, Alex S. Taylor, and Marco Combetto. Experiencing the Affective Diary. *Personal and Ubiquitous Computing*, 13(5):365–378, 2009.

[213] Michael Stonebraker. SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10–11, 2010.

[214] Lior J. Strahilevitz. A Social Networks Theory of Privacy. *The University of Chicago Law Review*, (72):919–988, 2005.

[215] Mark Stringer, Geraldine Fitzpatrick, Dan Chalmers, Eric Harris, Renan Krishna, and Markus Haarlander. Kuckuck–Exploring Ways of Sensing and Displaying Energy Consumption Information in the Home. In *Proceedings of the 9th ACM International Conference on Ubiquitous Computing, Workshop on Ubiquitous Sustainability: Technologies for Green Values (UbiComp 2007)*, 2007.

[216] Latanya Sweeney. k-anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.

[217] Josef H.M. Tah, Anthony Thorpe, and Ronald McCaffer. Contractor Project Risks Contingency Allocation Using Linguistic Approximation. *Computing Systems in Engineering*, 4(2-3):281–293, 1993.

[218] Emmanuel Munguia Tapia, Stephan S. Intille, and Kent Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. *Pervasive Computing*, pages 158–175, 2004.

[219] Michael Terry, Elizabeth D. Mynatt, Kathy Ryall, and Darren Leigh. Social Net: Using Patterns of Physical Proximity over Time to Infer Shared Interests. In *Extended Abstracts in 20th international conference extended abstracts on Human factors in computing systems (CHI 2002)*, pages 816–817, 2002.

[220] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[221] Endel Tulving. *Elements of Episodic Memory*. Oxford University Press, 1983.

[222] Yuya Uesugi, Tasuku Yamamoto, Jun Sawamoto, Norihisa Segawa, Eiji Sugino, Takuto Goto, and Hiroshi Yajima. A Proposal of an Application of Life Log to Remote Consultation. In *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 315–319, 2011.

[223] U.S. Congress. Health Insurance Portability and Accountability Act of 1996. *104th Congress*, 1996.

[224] U.S. Department of Health, Education & Welfare. Records, Computers and the Rights of Citizens. Technical report, 1973.

[225] Upkar Varshney. Pervasive Healthcare and Wireless Health Monitoring. *Mobile Networks and Applications*, 12(2-3):113–127, 2007.

[226] Sunil Vemuri. *Personal Long-term Memory Aids*. PhD thesis, Massachusetts Institute of Technology, 2004.

[227] Sunil Vemuri and Walter Bender. Next-Generation Personal Memory Aids. *BT Technology Journal*, 22(4):125–138, 2004.

[228] Sunil Vemuri, Chris Schmandt, and Walter Bender. iremember: a personal, long-term memory prosthesis. In *the 3rd ACM workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE 2006)*, pages 65–74, 2006.

[229] Sunil Vemuri, Chris Schmandt, Walter Bender, Stefanie Tellex, and Brad Lassey. An Audio-Based Personal Memory Aid. In *Proceedings of the 6th International Conference of Ubiquitous Computing (Ubicomp 2004)*, pages 400–417, 2004.

[230] Victor H. Vroom and Kenneth R. MacCrimmon. Toward a Stochastic Model of Managerial Careers. *Administrative Science Quarterly*, 13(1):26–46, 1968.

[231] Katarzyna Wac, Pravin Pawar, Tom Broens, Bert-Jan van Beijnum, and Aart van Halteren. Using SOC in Development of Context-Aware Systems: Domain-Model Approach. In *Enabling Context-Aware Web Services: Methods, Architectures, and Technologies*, pages 171–210. Chapman and Hall/CRC, 2010.

[232] James S. Walker. *A Primer on Wavelets and Their Scientific Applications*, pages 141–144. CRC press, 2008.

[233] William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart, and Rajarshi Das. Utility Functions in Autonomic Systems. In *International Conference on Autonomic Computing, 2004.*, pages 70–77, May 2004.

[234] Waugh, Andrew and Wilkinson, Ross and Hills, Brendan and Dell'oro, Jon. Preserving Digital Information Forever. In *Fifth ACM conference on Digital Libraries (DL 2000)*, pages 175–184, 2000.

[235] Mark Weiser. The World is not a Desktop. *Interactions*, 1(1):7–8, 1994.

[236] Timothy D. Wilson and Elizabeth W. Dunn. Self-knowledge: Its Limits, Value, and. Potential for Improvement. *Annual Review Psychology*, 55:493–518, 2004.

[237] Terry Winograd. Architecture for Context. *Human Computer Interaction*, 16(2):401–419, 2001.

[238] David Murakami Wood and Kirstie Ball. A Report on the Surveillance Society. *Information Commissioner Office*, 2006.

[239] Jude Yew, Faison Gibson, and Stephanie Teasley. Learning by Tagging: Group knowledge formation in a self-organizing learning community. In *Proceedings of the 7th International Conference on Learning Sciences (ICLS 2006)*, pages 1010–1011, 2006.

[240] Wanghong Yuan and Klara Nahrstedt. Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP 2003)*, pages 149–163, 2003.

[241] Robert B. Zajonc. Social Facilitation. *Science*, 149(3681):269–274, 1965.

[242] Philip G. Zimbardo and Michael R. Leippe. *The Psychology of Attitude Change and Social Influence.* McGraw-Hill New York, 1991.