33 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

New Concepts for Presence and Availability in Ubiquitous and Mobile Computing

Enabling Selective Availability through Stream-Based Active Learning

Mirko Fetter

University of Bamberg Press

33 Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Schriften aus der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg

Contributions of the Faculty Information Systems and Applied Computer Sciences of the Otto-Friedrich-University Bamberg

Band 33

University of Bamberg Press
2018

New Concepts for Presence and Availability in Ubiquitous and Mobile Computing

Enabling Selective Availability through Stream-Based Active Learning

Mirko Fetter

University of Bamberg Press
2018

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Informationen sind im Internet über http://dnb.d-nb.de/ abrufbar.

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg als Dissertation vorgelegen.
1. Gutachter: Prof. Dr. Tom Gross
2. Gutachter: Prof. Dr. Andreas Henrich
Tag der mündlichen Prüfung: 25.10.2017

Herstellung und Druck: docupoint, Magdeburg
Umschlaggestaltung: University of Bamberg Press, Larissa Günther
Umschlagbild: „Vernetzt“ © Maria Fetter

ISSN: 1867-7401
ISBN: 978-3-86309-623-6 (Druckausgabe)
eISBN: 978-3-86309-624-3 (Online-Ausgabe)
URN: urn:nbn:de:bvb:473-opus4-531619
DOI: http://dx.doi.org/10.20378/irbo-53161

For Bruno & Agnes

Acknowledgements

First and foremost, I would like to thank my family, a source of never-ending support and inspiration. I thank my wife Stephani and my children Bruno and Agnes for their love and understanding. I also thank my parents for their generous and self-forgetting sacrifice for their kids—without you this would not have been possible. I thank my sister and her family, for bringing me the joy of becoming an uncle twice during this endeavour of writing this thesis.

I thank my supervisor Prof. Dr. Tom Gross, for mentoring me in this process. My thanks goes to him, as well as to all my current and former colleagues at the Cooperative Media Lab in Bamberg and Weimar, with whom many of the concepts presented here were discussed and sometimes even conceived.

I worked with many talented students during the time of writing this thesis—some of which directly contributed to the research presented here. I want to thank all of them, but a few of them standout, as they worked together with me on some of the projects and prototypes that became part of this thesis. For his contribution to Disclosure Templates I want to thank Benjamin Zeller. For adding to the implementation of the Advanced Sensor Suite I thank Steffen Hippeli, Jakob Gomol, and Julian Seifert. For conducting research with me on the predictability of selective availability I thank Julian Seifert. I further thank David Wiesner and Jonas Pencke for their contribution to the LiLoLe implementation.

Table of Contents

Table of Figures

Table of Tables

Table of Listings

1 Introduction

The rapid speed with which modern information and communication technologies (ICT) is introduced in our daily work and private life has significantly transformed it in the last two decades. From being tied to one location with desktop computers and landline phones, now laptop computers and mobile phones allow us to work and communicate from everywhere. Thus, work and private life more and more intermix as former boundaries between those two aspects of our lives start to vanish (cf. Figure 1). Today, for example the location, does not allow estimating if a person is currently working or not. An email send to a smartphone might also reach us on our holidays far away from the office, and our significant other can send us a quick text message while we are in a business meeting. While we immediately see the problematic aspects of these two examples, I want to remind you of the chance, that the content of the communication in these two occasions has valuable information that outweighs the cost of the disruption. To say it in the words of Kranzberg's first law of technology: “Technology is neither good nor bad; nor is it neutral” [Kranzberg 1986, p. 545].

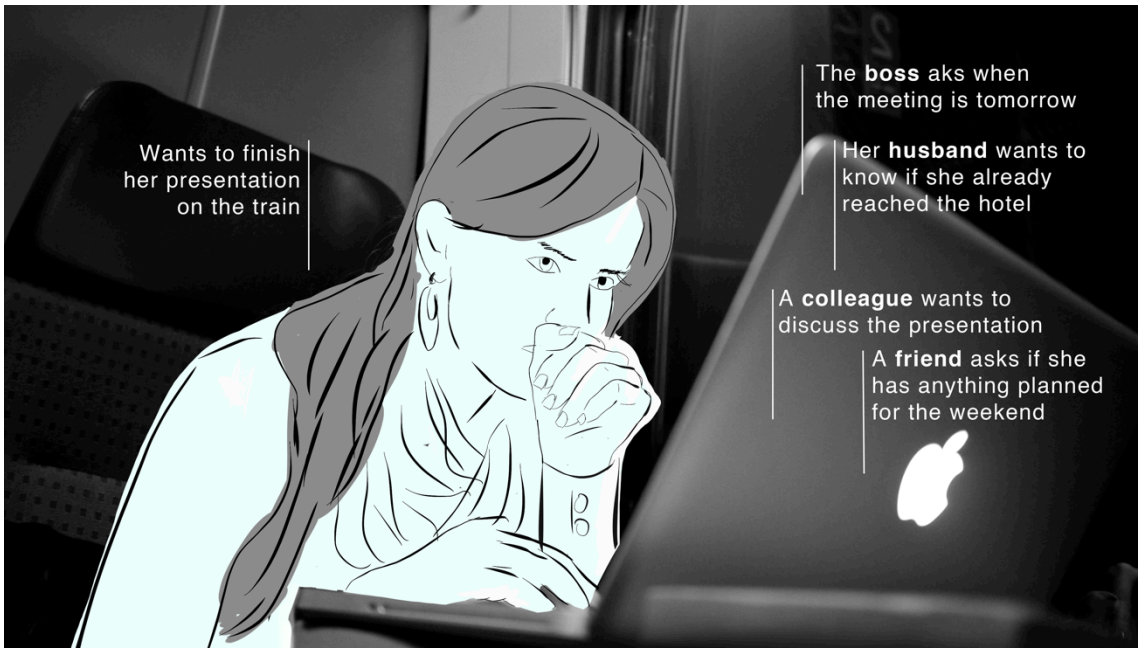

Figure 1. An illustrative scenario: A person working on the train on her computer while different persons with different concerns are trying to contact her via computer-mediated communication.

We see, constantly being online not only comes with risks, but also with chances. And those chances are the reason we let the technology into our lives in the first place. What was overlooked—or ignored—when developing these

technologies, was how to design them in a human-centred way. And now, as we have to deal with their side effects in form of communication or information overload, there is a chance that redesigning that very technology in a more human-centred way, can help us to conserve its benefits while reducing its disadvantages.

This thesis particularly looks at how people are able to better manage their availability and presence with others through technology in a more human-centred way and thus reduce the disruptive side effects of communication technologies without needing to let go their advantages. The aim of this work is to contribute to a better understanding of presence and availability management by suggesting a solution that can improve on the current state of the art.

1.1 Motivation

Modern forms of (*semi-*) synchronous communication like Instant Messaging (IM), SMS text messaging, but also telephoning, only rudimentary support people in finding appropriate moments for communicating. While the last two essentially only allow the recipient of communication to ignore an incoming message or a phone call, when it arrives in an inappropriate moment, IM applications at least provide minimum means to negotiate availability for communication. In current IM applications the online status is designed to signal a user's presence and availability for communication. Users can *manually* select between roughly a half-dozen online states reflecting their presence and availability to *all of their contacts*. While the basic idea of actively signalling one's own availability to potential communication partners is an improvement over the situation found for telephoning or text messaging, we also see that current means to do so are both: restricted and cumbersome.

Table 1. Four brief scenarios illustrating the influence of the situation and the relationship with the contacting person on our availability.

		Situation	
		Work-Life	**Private-Life**
Relationship	**Work-Life**	A co-worker asks a question via IM, regarding a document you wrote while you are working remotely in your home-office.	A customer has an urgent support question and is calling on your smartphone while you are doing your groceries after leaving the office.
	Private-Life	A friend is planning activities for the weekend, and writes a text-message while you are at a meeting with your manager.	Your parents are video calling to your tablet computer to chat with their grandchildren.

The *when, where, how,* and for *whom* we are available to communication are results of a continuous privacy regulation process—a matter of complex and

dynamic coherences, rules, and decisions we—as humans—constantly evaluate. To demonstrate the limitations of today's technologies and motivate this work, I provide four rather simple scenarios in Table 1. Let—for the sake of simplification—divide the relationships we have into those belonging to a work context and those belonging to a private context. Let us further assume there are only situations that can be uniquely assigned to either being part of our work life or our private life.

Already this simple setting shows us the limitations of current solutions:

- With only one online state it is impossible to adequately express one's availability to people we enjoy different types of relationships with.
- As the online state has also to reflect the current situation, it is necessary to constantly question whether the current online state fits the own current availability and eventually needs to be adapted manually.

We learn presence and availability are selective and dynamic. To provide meaningful ways for people to manage their availability in Computer-Mediated Communication (CMC), new concepts are needed, that account for this. With current tools, we only can decide to be available or unavailable for all contacts in the same way: for the co-worker, the friend, the customer, and the parents.

1.2 Research Problem, Scope, and Approach

According to Oulasvirta and Hornbæk [2016, p. 4960] a "research problem in HCI is a stated lack of understanding about some phenomenon in human use of computing, or stated inability to construct interactive technology to address that phenomenon for desired ends." Departing from this perspective, the research problems addressed in this thesis belong to both categories. First, this thesis aims to further minimise our lack of understanding on how users manage their availability and presence for instant computer-mediated communication systems towards different contacts. Second—building on this understanding—it proposes novel concepts and approaches to overcome the identified challenges with current interactive technology for availability management. Both was driven by previous work, stating that existing technology for managing availability in IM is currently insufficient, as:

- The single online status of most IM systems does not meet the users' need of expressing different availability to different contacts depending on various factors as the relationship, situation, information needs, etc. Hence, the online status is most times a sheer compromise.
- The need for a continuous manual adaption of the online status to reflect the current personal availability is at best exhaustive and at worst

unrealistic. The adaptation is often forgotten, leading to a wrong online status that accordingly is ignored by others.

The scope of this work thereby focuses on the improvement of current solutions to advertise one's availability and presence in *Instant Messaging* (IM) systems. The opportunity space for this research is framed by the new possibilities of *ubiquitous* and *mobile computing* technology that allows taking into account contextual information of the users, to better support them with their task.

The approach can be broken down into three distinctive steps. First, I am founding the concepts of availability and presence on theoretical work from various disciplines, in the sense of a concept-*driven approach* [Stolterman & Wiberg 2010]. This approach starts from theoretical rather than an empirical point as foundation for a future interaction design. Second, I used an *empirical approach* [Oulasvirta & Hornbæk 2016], to develop a more fine-grained understanding of user needs. And in a final step, I used a *constructive approach* [Oulasvirta & Hornbæk 2016] by combining applied machine learning and software engineering to conceptualise and implement a Framework that has the potential to empower people to manage their availability better with less effort.

1.3 Contribution of the Thesis

Overall the aim of this thesis was to conceptualise, design, and implement a framework that enables people to better manage their availability towards others and helps us to further and better understand how such systems should be designed. The central contribution of this thesis is three fold:

- I helped to develop a better understanding of how people can be supported in managing their availability and information disclosure to different audiences [Fetter *et al.* 2008; Fetter *et al.* 2010a; Gross & Fetter 2010].
- I introduce the concept of *Selective Availability* and show how it is predictable from sensor data through machine learning for nomadic users [Fetter *et al.* 2010b; Fetter *et al.* 2011b].
- I developed a concept for a *Framework* that is founding on *stream-based active learning* to build personal predictive models of *Selective Availability* and demonstrated through a proof-of-concept that my approach has the potential to automatically manage the users' availability [Fetter & Gross 2014].

Further, there are a number of additional publications that I co-authored during the completion of this thesis. A fraction of them indirectly contributed to the developed concepts here, but are not directly attributed to the three core contributions mentioned above. Foremost, these are the publications that are

concerned with IM in an applied context [Fetter & Gross 2008; Fetter & Gross 2009c; Gross *et al.* 2008; Gross *et al.* 2009], interesting IM usage patterns [Fetter & Gross 2009a; Gross & Fetter 2009], architectures for real-time applied machine learning in the context of availability and presence [Fetter & Gross 2009b; Gross & Fetter 2008], and novel tools for Experience Sampling [Fetter & Gross 2011a; Fetter & Gross 2011b]. A few of them are briefly discussed in the remainder of this work or referenced as examples.

In the Following, I give an overview on how the remainder of this thesis is structured.

1.4 Structure of the Thesis

In Chapter 2 we firstly develop a grasp of the concepts underlying this thesis and secondly reflect on the related work that was presented to the research community before and while this thesis evolved. I derivate the notions of presence and availability from the concept of interruptibility by showing how those concepts are rooted in the overarching theme of privacy.

In Chapter 3 I further foster our understanding of user needs in regard to privacy. I present patterns for *Selective Information Disclosure*, identified in a survey, and show how they can inform templates that help users to configure systems for availability management. I further report on an Experience Sampling Study, giving detailed insights in how users are selectively available towards different contacts. It further allows a precise view on the dynamic nature of availability.

In Chapter 4 I demonstrate the general feasibility of predicting *Selective Availability* from sensors with machine learning, based on the collected data from the previous study. I motivate the need for and discuss the implications of learning personalised predictive models. I also establish the requirements and discuss the design space of the context-sensing layer, which provides the groundwork for the approach proposed in the next chapter.

In Chapter 5 I present my concept for supporting selective availability through a life-long learning Framework (LiLoLe) relying on *Stream-Based Active Learning*. I further provide details on the proof-of-concept implementation of the Framework and discuss the respective components and building blocks of the implementation. And finally, I conclude with validating the feasibility of my concept based on a simulation with real user data.

The final Chapter 6 summarises and discusses the presented work and provides an outlook on future work.

2 Presence and Availability in HCI

To gain a common ground in terminology and concepts, as well as of the state of research in this field, this chapter aims to provide a sound understanding of the underlying concepts of this work and provides a detailed overview of the related work in respect to the core contribution of this thesis.

2.1 Underlying Concepts

The following section gives an introduction to the concepts of *presence* and *availability* in relevance to this thesis by narrowing down from broader concepts *privacy*, *self-disclosure*, and *impression management* and putting those in relation to the concept of *awareness*. Thereby this section aims to look at these concepts from a more general perspective rooted in fields like sociology, psychology or philosophy, and then applies a more specific human-computer interaction (HCI) perspective, to reflect the usage of this concepts in the field of HCI, and ultimately, in this work.

2.1.1 Privacy, Self-Disclosure, and Impression Management

First, I break down the broad and amorphous term *privacy*, by analysing definitions and conceptions from different fields. Subsequently, the aspects of privacy relevant in this work is highlighted and linked to the related concepts s*elf-disclosure* and *impression management* that are vital in this work. Further, this section reflects on the usage of these three concepts in the fields of HCI, and specifically in CSCW and Ubiquitous Computing.

2.1.1.1 Privacy

Privacy is a concept that defies easy definition, and can be reflected from many different viewpoints [Newell 1995], whether they are of philosophical, sociological, political, psychological, legal or just practical nature [Solove 2002]. Depending on the perspective, privacy can be interpreted as a condition, a process or, a goal [Margulis 1977; Newell 1995]. Robert Post [2001, p. 2087] goes even as far as saying that privacy is a "value so complex, so entangled in competing and contradictory dimensions, so engorged with various and distinct meanings, that I [he] sometimes despair whether it can be usefully addressed at all". According to Solove [2002] a view Robert Post shared with many scholars studying the different notions of privacy. In the Following I therefore start with a broad look at privacy and the prevailing definitions and will gradually refine my definition of privacy for this work. Therefore we begin with a first definition of the word privacy taken from the dictionary Merriam-Webster [2012e]: "

1) a : the quality or state of being apart from company or observation : seclusion
   b : freedom from unauthorized intrusion <one's right to privacy>
2) archaic : a place of seclusion
3) a : secrecy
   b : a private matter: secret"

Besides the attribution of privacy to a place, we already see two different meanings of privacy in this definition: The notion of seclusion, or as a very early privacy definition states "the right to be let alone" [Warren & Brandeis 1890, p. 193], and the notion of secrecy, in the sense of concealment of information. The Encyclopaedia of Psychology defines privacy and privacy regulation as "the selective control of access to the self or group" [Werner *et al.* 2000, p. 308]—based on the work of Irvin Altman [1975], one of the entries' authors—which addresses both notions (i.e. seclusion and secrecy) as it reflects on how people "open and close themselves to interaction". Nippert-Eng [2007, p. 2] provides the following definition: "privacy is a condition of relative inaccessibility. It is one conceptual end of a continuum of (in-) accessibility. "

In order to disentangle privacy and to understand the concept in its entirety, we will now extend our perspective by looking at broader reflections of privacy, from different viewpoints. Solove [2002, pp. 1094] analysed the body of work conceptualising privacy from authors of different disciplines (philosophy, psychology, sociology, law, etc.) and subsumes these conceptualisations under six headings: (Sol1) "the right to be let alone"—which is consonant with the notion of Warren and Brandeis [1890, p. 193]; (Sol2) "limited access to the self", which is the idea of being able to protect oneself from the unwanted access of third parties; (Sol3) "secrecy", which is the ability to shield certain matters from others; (Sol4) "control over personal information", as the ability to be in charge about which data about oneself is revealed to others; (Sol5) "personhood", which is the prevention of harm to the personality and dignity of an individual; and finally (Sol6) "intimacy", which is concerned with controlling the access to the intimate facets of a person's life and relationships. Solove adds that these headings are not meant to be taxonomical, as some of these concepts overlap, but nevertheless are "distinctive perspectives" on privacy. He further differentiates between concepts that are more interested in the process of and the practices around privacy and those that are more concerned with the outcome of privacy.

In an earlier work on privacy William Prosser [1960, p. 389] is more interested in the goals and ends of privacy, when identifying four different torts[1] from a legal perspective. These are (Pro1) "intrusion" (i.e., the act of harming a

[1] i.e., a civil wrong; act of unfairly causing harm to a person

person's seclusion or solitude), (Pro2) "public disclosure of private facts" (i.e., the act of revealing embarrassing truths about a person), (Pro3) "false light in the public eye" (i.e., the act of generating publicity that is incorrectly attributing utterances or opinions to a person), and (Pro4) "appropriation" (i.e., the act of advantageously but unrightfully making use of an aspect of a persons' identity). Of course these concepts of privacy violations can be mapped to the positive formulated privacy concepts of Solove. Prosser's definition of intrusion (Pro1) can be related to a violation of the right to be let alone (Sol1) as well as a violation of the limited access to the self (Sol2) and intimacy (Sol6). The public disclosure of private facts (Pro2) can be seen as a violation of the `control over personal information (Sol4), as well as of secrecy (Sol3). The concepts false light in the public eye (Pro3) as well as appropriation (Pro4) can be seen as violation of Solove's privacy concept of personhood (Sol5).

Alan Westin [1967] provides one further view on the concepts of privacy in his authoritative work "Privacy and Freedom". His conceptualisation of privacy defines four states [Westin 1967, pp. 31-32] (solitude, intimacy, anonymity, reserve) and four functions [Westin 1967, pp. 32-39] of privacy (personal autonomy, emotional release, self-evaluation, and limited and confidential communication). The first state, "Solitude" (Wes1), which he describes as the most complete state of privacy achievable by an individual, is the state where a person is free from any observation (auditory and visual) of others. The second state—"Intimacy" (Wes2)—is related to the notion of *group privacy*, which I reflect on later. It is the state of collective seclusion of a group people as for example a couple, a family but also a small number of colleagues. "Anonymity" (Wes3), the third state, is described as the freedom of being able to act unidentified in public. And finally "Reserve" (Wes4), which Westin describes as a form of an implicit consent against disclosing personal details by means of social etiquette. These four states—which can be seen on a gradual opening scale from seclusion and secrecy towards social interaction and disclosure—are the basis for his discussion of the functions of privacy. These functions describe the motives and rationales beyond the desire of privacy and are the following: "Personal Autonomy" (Wes5), which is the striving for a self-determined life, which in most parts is free from the manipulation or domination by third parties; "Emotional Release" (Wes6), relates to the ability of an individual to temporarily retract oneself from society and the roles, rules, and norms it imposes, in order to relax; "Self-Evaluation" (Wes7) describes the function privacy has in generating periods of retreat, allowing individuals to assess and process their experiences in order to plan future actions and originate; and finally the function of "limited and protected communication" that allows to share selected, confidential information with trustworthy third parties (Wes8). Steeves [2009] reflects on the social value of privacy based on Westin's states of

privacy and identifies a gradual increase of openness and social interaction in those concepts. This conception of privacy as informational control is illustrated in Figure 2.

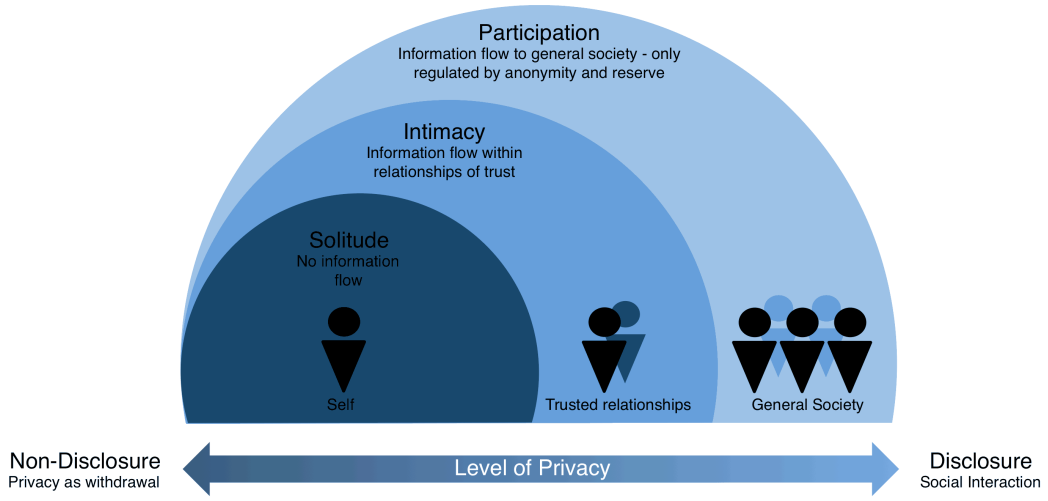

Figure 2. Privacy as informational control (based on [Steeves 2009, p. 201]).

The sociologist Goffman [1971, pp. 28-61] identifies eight "territories of the self', most of which can be seen as mechanisms related to the previously discussed privacy concepts. An intrusion into these territories by others can lead to a feeling of discontent for the person, lead to a withdrawal or even a conflict—and in most cases will be perceived as a violation of privacy. Most of the introduced territories are of spatial nature and can be linked to Hall's [1966] notion of proxemics. The first territory is "personal space" (Gof1), the territory surrounding oneself—varying in size depending on the situation (e.g., smaller in a overcrowded commuter train, elevator). "The stall" (Gof2) is a self-contained space that can be temporally allocated or occupied by a person and providing a boundary that is easily visible (e.g., telephone booth, an arm chair). "Use space" (Gof3) defines the territory surrounding a person that is claimed by this person, in order to carry out a specific task (e.g., the line of sight between a visitor of a museum and an artwork). "The turn" (Gof4) is not a spatial territory, but has a more temporal aspect, as it describes the formal or informal orderings structuring interactions with others, often relying on commonly accepted norms or official rules (e.g., to enqueue and not cut a line). "The sheath" (Gof5), describes the narrowest of the spatial territories, the body of a person, where the skin or clothes form the boundary. "Possessional territory" (Gof6), is made up of the objects surrounding a person that a person permanently (e.g., a rucksack or a lighter) or temporally (e.g., magazines in a waiting room) possesses. "Information preserve" (Gof7) relates to the access control to personal facts (e.g., answering to intrusive questions or expecting the content of a letter to be respected as private). Finally "conversational preserve" (Gof8) is concerned with

the ability to control with whom and when a person is talking, and who can join such a conversation. While most of these territories exist in different cultures, the characteristics of these territories are not fixed but come with a "social determined variability" [Goffman 1971, pp. 40]. All of the territories identified by Goffman again can be related to the two overarching privacy claims of *secrecy* and *solitude*. Accordingly the territorial behaviour shown by people can be interpreted as mechanisms for achieving privacy [Altman 1976]—either *secrecy* or *solitude*.

Altman's interest therefore strongly lays on privacy as a process, and not so much as a state or goal to achieve. His description of privacy as "a dynamic, changing boundary-regulation phenomenon" [1984, p. 102] takes privacy as an underlying on-going, process of regulation, which continuously determines our degree of social interaction with others. For Altman, privacy needs to be understood as a dialectic (in regards to openness and closeness) and dynamic (over time) boundary regulation process, where privacy is not static but "a selective control of access to the self" [Altman 1975, p. 18]. Altman's [1984, p. 78] notion of privacy therefore has two important propositions: "Privacy is a dynamic process whereby people vary in the degree to which they are accessible to others. Our use of the term, therefore, covers the whole gamut from extreme openness to extreme closeness". Altman [1984, p. 77] also says: "People control their accessibility to others, making privacy regulation a 'boundary control' process" with respect to the information that is outgoing but also with respect to the information that is incoming. The ultimate goal, according to Altman [1976] is to reach an optimum level of privacy by constantly evaluating the desired against the achieved privacy, and taking appropriate measures. Others share this view, that privacy is a constant process. For example, Westin [1967, p. 7] adheres to this idea that each "individual is continually engaged in a personal adjustment process". He goes on saying that everybody continually "balances the desire for privacy with the desire for disclosure and communication of himself [sic] to others, in light of the environmental conditions and social norms set by the society in which he [sic] lives".

One further aspect we can take away from the previous cited privacy definitions is, that conceptualising privacy as secrecy or solitude with a dyadic perspective is to narrow [Solove 2002], as such a definition ignores the notion of group privacy. Arnold Simmel [1971, p. 81] describes this idea the following way: "We become what we are not only by establishing boundaries around ourselves but also by a periodic opening of these boundaries to nourishment, to learning, and to intimacy. But the opening of a boundary of the self may require a boundary farther out, a boundary around the group to which we are opening ourselves". Westins' state of "intimacy" (Wes2) also identified by Solove (Sol6) describes this kind of boundary around a group. Also Goffman's "conversational

preserve" ultimately achieves a form of group privacy when it is defined as "the right of a set of individuals [...] to have their circle protected from entrance and overhearing by others" [Goffman 1971, p. 40]. Petronio [2002] uses the notion of personal and collective boundaries, when describing how people control the access to private information about the self as well as the access to private information that is co-owned with others. A working definition of privacy has to take this kind of polyadic interactions into account.

In order to achieve privacy, people of course rely on laws and legislation, but also, and even more on functions of social norms and practices. Nippert-Eng [2007] discusses different means for achieving privacy: Signals are shared, culturally understood behaviours indicating a desire for privacy. Space and time, are used to physically remove one self or temporally scheduling one's accessibility. Sensory and other information blocking techniques help to remove one self from sensory stimuli to mentally block off potential demands for attention (also with assistance of technology). And finally reserve, as a means to selectively regulate intimacy and information disclosure to restrict the accessibility of the own person.

So, when talking about privacy, two questions need to be answered: What is at stake (i.e., what should be kept private) and against which party this claim is invoked [Waldo *et al.* 2010]. What is making privacy so complex is the multifaceted intermingling of multiple interests resulting in difficult trade-offs.

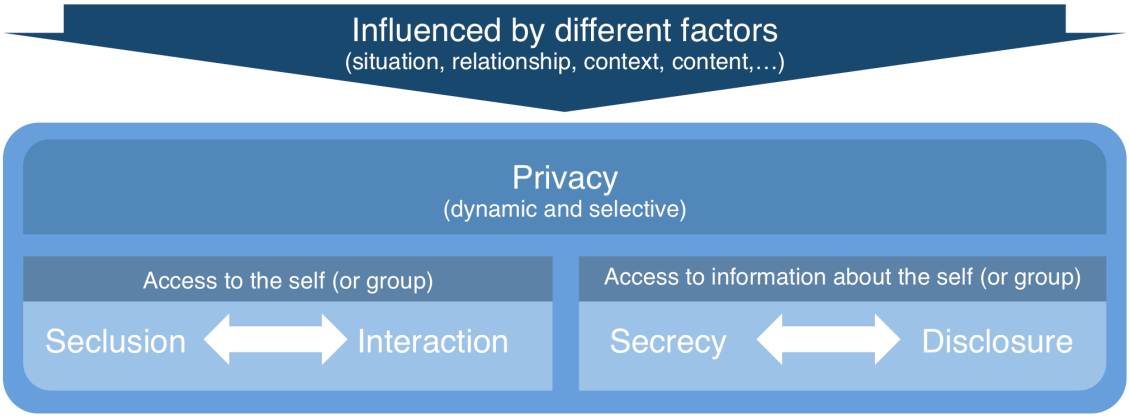

Figure 3. Model of the two main privacy functions regulating the access to the self (or group) and to information about the self (or group) as opposite pairs: Seclusion vs. Interaction and Secrecy vs. Disclosure.

In the remainder of this work, the term privacy is used, to refer to a continuous process (i.e., dynamic) of regulating the access to the self or a group (from *seclusion* over *intimacy* to *open interaction*) and to information about the self or the group (from *secrecy* to *information disclosure*) with different third parties (i.e., *selective*). This includes the aspect of variations in the degree of opening or closing oneself or a group to others. It also acknowledges the

multifarious functions and goals underlying this process, as well as the different mechanisms privacy relies on and factors that influence privacy. In Figure 3 this view on privacy is illustrated.

2.1.1.2 Self-Disclosure and Impression Management

In the context of this work the term privacy is strongly related with the ability to control access to personal information and the involved processes and mechanisms. Thereby our notion of privacy has a considerable connection to the concept of *self-disclosure*, which is the intentional or unintentional act of giving away information about oneself, including one's inner state (feelings, ambitions, thoughts, fears, etc.). According to a definition given in [Reis 2000] self-disclosure is "the process by which individuals reveal personal thoughts, feelings, and experiences to other people." In Altman's [1973] social penetration theory mutual self-disclosure becomes a fundamental process of building intimate relationships. By disclosing oneself to a third party, we are becoming vulnerable. This vulnerability—letting another person know about our feelings, motives, desires, etc.—allows a close relationship to develop. The communication privacy management theory developed by Sandra Petronio [2002]—building on Altman's social penetration theory—explains self-disclosure in form of boundaries separating private from public spheres. The permeability of those boundaries is continuously adapted, regulating the flow of information. Self-disclosure is the "the act of revealing personal information to others" [Jourard 1971, p.3], that they would not normally know or discover. Thereby breadth as well as depth of disclosure can be a measurement for its extent [Derlaga & Berg 1987]. In order to relate the concept of self-disclosure back to privacy, Westin [1967, p. 37] says that "deciding when and to what extent to disclose facts about himself [oneself]—and to put others in the position of receiving such confidences—is a matter of enormous concern in personal interaction, almost as important as whether to disclose at all."

Self-disclosure again is related to the concept of *impression management* (also self presentation [Leary & Kowalski 1990]), where a person seeks to convey a specific impression by controlling the information that is disclosed about oneself. That is, people try to influence the *impression formation* [Asch 1946] at the recipient—attempting to codetermine how they are perceived by shaping the information they disclose about themselves and thus constructing and crafting their public perception. Existing work in sociology (e.g., symbolic interactionism) and social psychology studied these phenomena. In social science the research on impression management analyses "how individuals present themselves and respond to the presentations of others" [Metts & Grohskopf 2003, p. 358]. Schlenker [2000, p. 236] gives the following definition: "Impression management is the goal-directed activity of regulating or

controlling information that is conveyed to audiences, thereby influencing the impressions formed by those audiences." Leary and Kowalski [1990, p. 34] acknowledge this purposive character when stating that people in daily encounters "try to convey images of themselves that promote their attainment of desired goals". Manning [2005, p. 398] says impression management characterises the "wide variety of strategies used by people to control the ideas others have about them" and that it is "concerned with the general ways in which people present themselves in public settings". In addition it should be mentioned, that while a lot of work is concerned with how people convey an image to an outer audience (i.e., third parties) some work also acknowledges an inner audience (i.e., the self) as the target of impression management [Greenwald & Breckler 1985]. Goffman [1955], who pioneered research on impression management with a qualitative approach, coined the term *face* as the „positive social value a person effectively claims for himself by the line others assume he has taken during a particular contact" (p. 213) and the term *face-work* to refer to the regulating process of impression management. Thereby self-disclosure—intentional or unintentional—can be seen as one major factor in influencing one's *face*. Goffman draws a distinction between information that is *given* in an intentional and controlled manner, and the information that is *given off*, that unintentionally is leaking through in an unmanaged way. As Goffman's work is mostly concerned with face-to-face interaction, this information is revealed through verbal and nonverbal expressive behaviour in interpersonal communication. Thereby the nonverbal behavioural acts in forms of facial expressions, emotional affects, or the vocal tone are usually more difficult to control, then the content of our verbal communication. In this light, it has to be noted that impression management can be seen as a continuous, unconsciously performed mechanical process, but that it also can become a central, and conscious task (e.g., in a job interview). In neither case impression management should be understood as pretense [Leary & Kowalski 1990]. In Leary and Kowalski's [1990, p. 36] two-component model, *impression motivation* and *impression construction* are identified as the two central processes of impression management. Thereby the process of *impression motivation* is a rather conscious process of evaluating the "goal-relevance of impressions" inline with the "value of desired goals" and the "discrepancy between desired and current image" in a given situation. Based on this motivation, the process of *impression construction* has to take into account the "self-concept" as well as "desired and undesired identity images","role constraints", the "target's values", and the "current or potential social image". A third component that can be found in-between the lines of Leary & Kowalski's model [1990] is the process of *impression monitoring*. This is entangled with two abilities: First, the ability to identify cues combined with the ability to draw conclusions from these cues, allowing to infer the

others' impression of oneself; and second, the ability to monitor and regulate the own behaviour. This ability is a construct also found in social psychology under the term self-monitoring, which is the "self-observation and self-control guided by situational cues to social appropriateness" [Snyder 1974, p. 526]. Of course, this again is related to the concepts of self-image, self-schema, and social identity, whose discussion is out of the scope this thesis.

Relating the concept of impression management back, to privacy we can understand Solove's [2002, p. 1106] interpretation of Richard Posner's privacy definition "as a form of self-interested economic behaviour, concealing true but harmful facts about oneself for one's own gain" as one practice of impression management. It goes without saying, that to form an impression—positive or negative—the behaviour in some form needs to be referable to a person. Accordingly, when people act anonymously impression management is less critical. Westin [1967, p. 31] reflects on this when referring to anonymity as the option to not be held "to the full rules of behaviour and role that would operate if he [or she, i.e., the anonymous person] were known to those observing him [her]". Naturally, to be able to act anonymously, individuals (as well as groups and institutions) need to be in control about "when, how, and to what extent information about them is communicated to others" [Westin 1967, p. 7]. A definition that is nicely interpreted as the "desire of people to choose freely under what circumstances and to what extent they will expose themselves, their attitude and their behaviour to others" [EPIC - Electronic Privacy Information Center 2001]. To reach this desire, people need the ability "to control the terms under which their personal information is acquired and used" [Culnan 2000, p. 20]. An ability, which to master becomes an increasing challenge in the digital age.

Concluding for the context of this thesis, we note that *self-disclosure* can be understood as a social mechanism, which is an enabler for interaction. By intentional disclosing ourselves, we open ourselves up to social interaction. *Impression management* thereby is a mechanism, which people apply to shape how others perceive them. Both are instruments in daily face-to-face interaction, and from there extends into our digital life, which is discussed in the following.

2.1.1.3 A HCI Perspective on Privacy, Self-Disclosure, and Impression Management

The introduction of new technology often comes with new privacy concerns. One of the earliest privacy definitions, the earlier mentioned "the right to be *let alone*", was formulated by Warren and Brandeis [1890, p. 208] as an answer to "recent inventions" like "instantaneous photographs" and "numerous mechanical devices", that now threatened to emotionally harm the individual. Another early example from the field of electronic communication was the

introduction of the function call display for landline wired phones. This functionality allowed the receiver of a phone call to see a caller's phone number, displayed on the phone. This functionality was seen as privacy invading, as it did not allow callers to stay anonymous until they revealed their identity on the phone [Lyon & Zureik 1996]. While call displays and photography today are considered ordinary, many steps in technological evolution were accompanied with privacy concerns. When staying within the example of the evolution of photography, some illustrative examples from the last two decades come to mind. So, in recent years privacy concerns were raised when: cameras were added to mobile phones, and hence made photography more ubiquitous; online services like flickr, instagram or facebook started to easily allow making photos available to a wide audience; Google Street View to a huge extent made pictures of peoples' homes and neighbourhoods available online; and very recently, the built-in camera of Google Glass that allow taking pictures relatively unnoticed. Technological advances in information and communications technology (ICT) repeatedly and continuously challenge our society to reflect on privacy. These technological changes come in hand with societal shifts as well as with singular events of national or even global impact (e.g., 9/11 or global surveillance disclosures in the last years), which challenge and transform our perception of privacy [Waldo *et al.* 2010].

The field of HCI is at the forefront of shaping how future information technology addresses human needs—in our case, whether it enables or hinders people to put individual privacy needs into effect. Consequently, privacy is a central aspect of HCI, as for example Karat et al. [2006], Ackerman and Mainwaring [2005], and Iachello and Hong [2007] show, when providing an overview of the influence, the understanding, and the consideration of privacy in past and current HCI research. In HCI, privacy is often discussed in context of the related terms trust—which is "a measure of how much users believe that the system will do what they ask it to do without causing any harm" [Karat *et al.* 2006, p. 640]—and security—which is the "concept associated with how well a system can protect information it contains" [Karat *et al.* 2006, p. 640]. Both aspects are important necessities for enabling privacy, but their discussion is out of the scope of this work. That is, we are more concerned with how the privacy needs of users can be supported and less focussed on if a system can factual (in respect to security) or perceptual (in respect to trust) really achieve this privacy. In the Following I primarily look at work that focuses on privacy and impression management requirements of users of ubiquitous and mobile computing, as well as frameworks and guidelines that inform the design of privacy-preserving software.

In the last years, mainly the advent of new paradigms like ubiquitous, mobile, or social computing introduced new challenges for the design of privacy

preserving systems [Ackerman & Mainwaring 2005; Dourish & Anderson 2006]. For example, in ubiquitous computing great amount of sensor data are collected upon the users, which can reveal a lot about an individual's whereabouts and whatabouts. While this data is often collected with a specific aim—that is, to directly or indirectly support end-users—the usage of such systems most times encompasses a form of privacy trade-off decision for its users [Waldo *et al.* 2010]. In the field of Ambient Assisted Living (AAL) for example, this includes the trade-off for the elderly between loosing privacy, by having all forms of medical data collected, and allowing sensors to monitor one's daily routines and social interactions for gaining the possibility of a more independent life. Also, according to Barkhuus, the increasing usage of mobile devices brings emerging issues for personal privacy now that "mobile applications share information that was previously unsharable" [Barkhuus 2012, p. 367]. For example, location sharing technologies on social media platforms such as foursquare [Foursquare Labs Inc. 2015], Google+ [Google+ 2015] or Facebook [Facebook Inc. 2015] now allow us to access personal information about people that formerly only was available to people in the vicinity. Thereby privacy problems arise in the tensions of today's behaviour and tomorrow's potential consequences according to Ackerman and Mainwaring [2005]. Adams [Adams 1999] identifies three factors influencing users' perceptions of privacy risks in multimedia communication: information sensitivity, information receiver, and information usage. The interaction of these factors influence how people share information with others in multimedia communication based on cost-benefit assessments of the potential privacy risks. Dourish and Anderson [2006] describe this view on privacy as being of *economic rationality*. In their analysis of privacy implications for HCI they argue for also taking the social and cultural contexts of privacy stronger into account when designing software. Besides the perspective of *economic rationality* they argue that the perspectives of *practical action* and *discursive practise* should have a stronger influence on the design of privacy mechanisms in HCI. Dourish and Anderson make a strong case that—like usability—also privacy cannot be retrofitted to an application or service ex-post.

But with the advent of Ubiquitous and Mobile Computing, HCI researchers and designers also have to ask if privacy is still a possibility at all. Already Weiser identified privacy as one of the key issues [Weiser 1991] when conceiving Ubiquitous Computing (UbiComp) and it continues to be a challenging aspect [Langheinrich 2009]. Camp and Connelly [2007] even call privacy versus UbiComp a Hobson's choice, meaning there is no choice, but merely the option of benefiting from UbiComp and risking privacy or not. As UbiComp—with the data collected on the user—is per se privacy invading, they

propose value-sensitive design as one approach to design systems that at least minimise the privacy violations.

We can see privacy is a central requirement when creating human-centred software and services. And HCI and CSCW research has a long history in identifying, detailing, and designing for such requirements. The whole field of CSCW is "grounded in the thoughtful analysis of the impact of technology and its design" [Boyle & Greenberg 2005, p. 331].

At first, this requires to generate a sound understanding of what privacy is and how it relates to the design of a system. Therefore, different guidelines, frameworks and models have been developed in these two fields, informed by countless studies and literature analyses from different fields. In the Following we glance at some of the most prominent examples. Palen and Dourish [2003] propose a conceptual framework for HCI practitioners to analyse privacy, grounded in the work of Irwin Altman. Based on this work by Palen and Dourish, Carew and Stapleton [2004] present a conceptual framework for interpreting privacy in the field of information system development. The *Privacy Grounding Model* [Romero *et al.* 2013] allows assessing how a mediated communication system fulfils people's need for the lightweight coordination of privacy. Boyle and Greenberg [2005] as well as Belllotti and Sellen [1993] transfer insights from video media spaces into guidelines for the design of CSCW systems. Barkhuus [2012] uses Nissenbaum's theory of *contextual integrity* as a framework to reflect on the concept of privacy in HCI. According to Nissenbaum's [2004] notion of *contextual integrity* privacy is strongly entangled with context. Nissenbaum sees two types of informational norms that influence how information is shared, both shaped by the context. First, the norms of appropriateness dictate what information "is appropriate or fitting in a particular context" [Nissenbaum 2004, p. 138]. Second, the norms of flow or distribution regulate the transfer of information between different parties based on relationships. Barkhuus relates this theory to HCI topics like location sharing and advocates for a more nuanced treatment of the notion of privacy.

Davis and Gutwin [2005] identify two central aspects of privacy in CSCW: *confidentiality* (i.e., control over data a person gives away), and *solitude* (i.e., control over data a person receives). Both aspects can be easily related to the two overarching themes in the conceptualisation of privacy: *secrecy* and *seclusion*. Bellotti [1998] distinguishes between two forms of privacy: *normative* and *operational*. While *normative* privacy (cf. [Reiman 1995]) refers to a set of norms that determine specific aspects of a person's nature and activity as inherently private, *operational* privacy is a view on the capabilities that empower people to control access to what is deemed private. Accordingly, *operational* privacy is of more interest for CSCW and HCI, as it can inform the design of access-control systems.

Besides full models and frameworks, many hands-on suggestions when designing for privacy can be found in the HCI literature. A few ideas that are central to this work are the following: Privacy-centred design has to take the individual differences [Dillon & Watson 1996] in the attitude to privacy into account [Ackerman & Mainwaring 2005], as privacy "does mean different things to different people" [Karat *et al.* 2006, p. 642]. Richards and Christensen add that applications "tend to show the same information, about a person, no matter who is viewing it" [Richards & Christensen 2004, p. 80]. They also reflect on the selective and dynamic aspects of privacy when stating: "What we're comfortable revealing depends a lot on to whom we're revealing it. This context is both nuanced and highly dynamic." [Richards & Christensen 2004, p. 82]. Another concern when designing for privacy is the technical firmness [Beckwirth 2003] of the users. People not always fully grasp the impact a certain technology has on their privacy nor do they fully understand the means they have to control privacy invading technology. And finally, when designing for information disclosure, it is also important to understand that information and privacy is not one-dimensional. In a statistical analysis of information disclosure datasets Knijnenburg *et al.* [2013] reveal the multidimensionality of information disclosure behaviour. They show that degree of disclosure for each kind of disclosed information cannot simply be summed up into a one-dimensional 'disclosure score' as there are complex relationships between the different dimensions.

Besides privacy, also *impression management* has been an aspect of work done in the field of HCI and CSCW. For example, Raento and Oulasvirta [2008] raise concerns that social awareness systems should support the adequate presentation of self (i.e., impression management). Boyle and Greenberg discuss aspects of self-appropriation and impression management in the context of media spaces [Boyle & Greenberg 2005]. Birnholtz *et al.* [2012a] studied impression management strategies in computer-mediated communication for co-located workers. Among other things Birnholtz *et al.* advocate in their work [2012b] to provide space for ambiguity, to allow for impression management. According to the proposal of the CHI 2004 workshop on forecasting presence and availability, one goal of the workshop was to "address techniques that allow users to understand and manage how forecasting affects the image that is projected of them" [Tullio *et al.* 2004, p. 1714]. Johri [2012] takes a look at how technology-mediated communication influences the way we form impressions of others, when analysing the *impression formation* process in distributed and virtual work environments. As a result, a model on the impression formation process in technology-mediated distributed work settings is presented to inform the design of technology. Stutzman and Hartzog [2012] looked at boundary regulation behaviours in social media and derived insights that can guide the

design of group context management systems to support users' impression management needs.

2.1.1.4 Conclusions

To conclude this analysis of privacy, self-disclosure, and impression management, we can see that while the concepts are complex in themselves, from the perspective of HCI and CSCW the interest in these concepts is mainly operational [Bellotti 1998]. Therefore, our main take aways are the two central functions of privacy (cf. Figure 3) we identified earlier. Accordingly, if we define those two functions as goal—the ideal state a person wants to achieve as an individual—the question is how to design systems that support users of computer-mediated communication tools to achieve this state. Accordingly, the aim is to develop lightweight tools that allow people to achieve two things:

1. An optimal balance between *seclusion* and *social interaction*, towards different social entities, at any given time, that provides room to retreat and focus and at the same time allows taking part in communication and social interaction.
2. An optimal balance between *secrecy* and *information disclosure,* towards different social entities, at any given time, that keeps the respective recipients of the information adequately informed and at the same time does not conflict with the conveyed impression or discloses otherwise sensitive information.

2.1.2 From Interruptibility to Availability and Presence

In the Following I build up on the previous conception of privacy to provide an understanding of the term *Interruptibility*[1]. This understanding helps us to define those two concepts that are central to this work, namely *Availability* and *Presence*. We start with a definition of the term *Interruptibility*.

2.1.2.1 Interruptibility

In the English language the suffix "-ibility" is used to form a noun from an adjective that ends with "-able" or "-ible"—often expressing some form of ability or suitability for a function or condition. The suffix "-ible" again, is used to form an adjective from a verb, with the meaning "able to be" [Cambridge Dictionaries Online 2014]. Accordingly the word *Interruptibility* refers to the *ability to be interrupted,* whereat the verb *interrupt* is defined by the Merriam-Webster [2013b] dictionary as: "

: to cause (something) to stop happening for a time

[1] In some papers relevant for this work interruptibility can also be found with a different spelling, i.e., interruptability. In this text the spelling interruptibility is predominantly used with the only exception in direct quotes.

: to cause (something) to not be even or continuous : to change or stop the sameness or smoothness of (something) [...]

1) to stop or hinder by breaking in [...]
2) to break the uniformity or continuity of [...] “

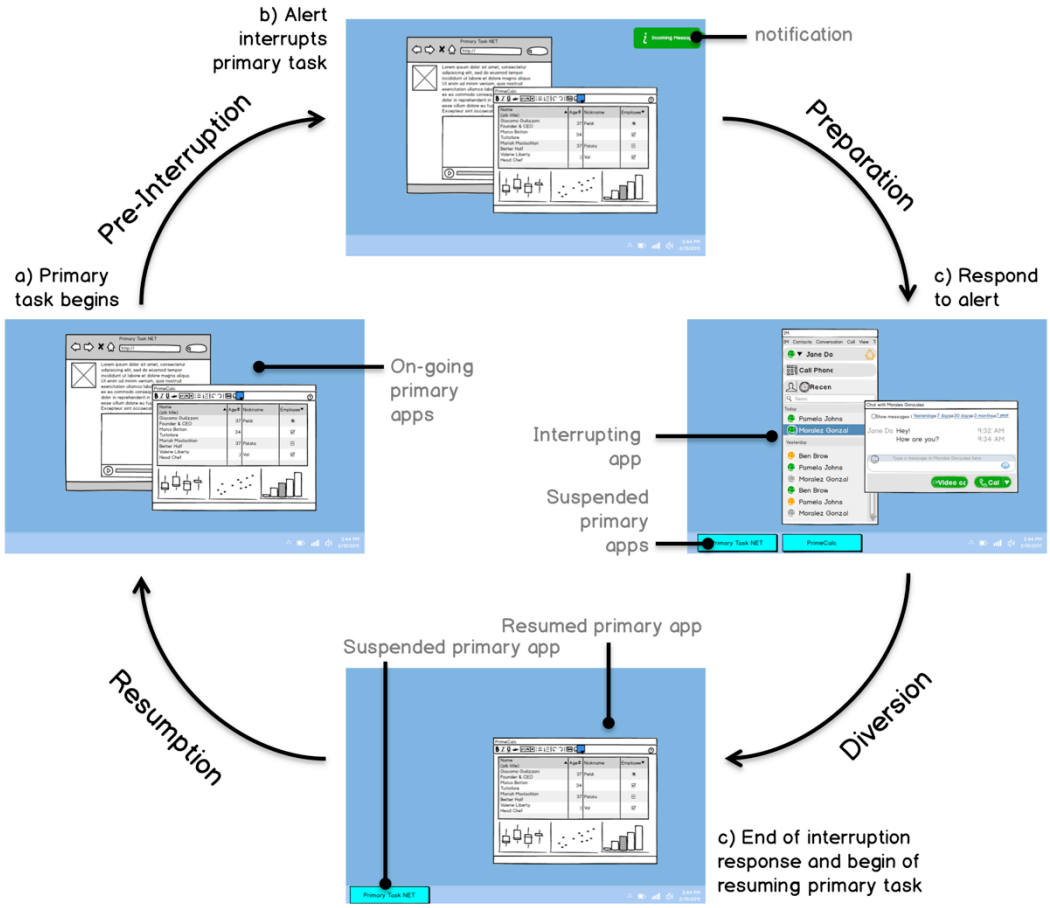

Figure 4. The four phases of the interruption lifecycle by Iqpal and Horvitz (adapted from [Iqbal & Horvitz 2007]).

Hence, *Interruptibility* is the state or condition of being able to deal with interruptions, even so such an interruption disturbs the continuity of a currently on-going process. In order to define *Interruptibility*, it seems to be necessary to first provide an understanding of what makes up an interruption in the context of this work. Based on this understanding, we then can define interruptibility as one’s general receptiveness to interruptions.

In the field of HCI various definitions for interruption exist. For example, O'Conaill and Frohlich [1995, pp. 262] define interruption as “a synchronous interaction which was not initiated by the subject, was unscheduled and resulted in the recipient discontinuing their current activity.” For this current activity usually the notion *primary task* (i.e., “normal daily tasks that users perform as their primary responsibility while in the computing environment” [Iqbal & Horvitz 2007, p. 678]) is used, which then is temporarily interrupted by a

secondary task (which is "much less important, generally not regarded as part of the user's main task, and is not under control of the user" [Ritter *et al.* 2014, p. 142]) and resumed later. Iqpal and Horvitz visualise this process in their interruption lifecycle, consisting of four phases (cf. Figure 4), which are pre-interruption, preparation, diversion, and resumption. In his article together with Latorella, McFarlane [2002, p. 18] cites his doctoral thesis, where he gives the following definition for *human interruption*: "the process of coordinating abrupt changes in people's activities." Another definition by Myata and Norman [1986, p. 268] characterises interruptions as introducing "new tasks on top of the ongoing activity, often unexpectedly" resulting in conflicts as a consequence of our limited cognitive capabilities. Such conflicts can be of different nature, as for example cognitive overload [Fussell *et al.* 1998], information overload [Farhoomand & Drury 2002], or communication overload [Harper 2010] to name a few. One reoccurring aspect of interruption research is how interruptions influence the execution of tasks in respect to the efficiency and effectiveness based on metrics such as task performance and error rate [Bailey & Konstan 2006]. One specific issue in this field is the interruption lag and the resumption lag (cf. Figure 5), which affects the overall performance [Trafton *et al.* 2003].

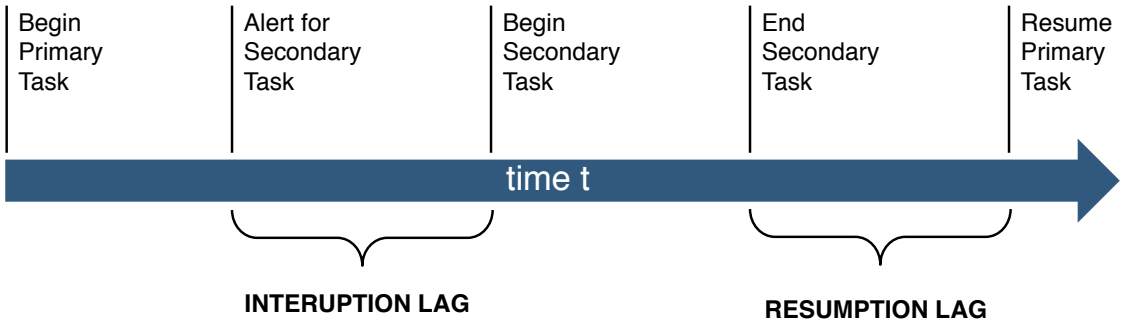

Figure 5. Visualisation of an interruption through a secondary task and the resumption of the primary task (adapted from [Trafton *et al.* 2003, p. 585]).

Further, different kinds of interruptions can be distinguished. To allow for a more detailed reflection on interruption McFarlane and Latorella [2002] came up with a *taxonomy of human interruption* with the following dimensions: source of interruption, individual characteristic of person receiving interruption, method of coordination, meaning of interruption, method of expression, channel of conveyance, human activity changed by interruption, and effect of interruption. To get a better understanding of the term interruption and narrow it down for this work, I firstly focus on the main distinctive feature: the source of an interruption. The two main categories in respect to the source are whether the interruptions are induced externally or internally [Mark *et al.* 2005]. Miyata and Norman define those two categories the following way [1986, p. 268]: "External interruptions result from events in the environment. Internal

interruptions come from our own thought processes—new ideas that draw attention from the current activity." Gonzales and Mark [2004] also make this differentiation. They describe internal interruptions as "self-initiated switching among working spheres" and external interruptions as "a condition in the environment that motivates switching " [2004, p. 118]. Also McFarlane and Latorella [2002, p.19] distinguish between the "Self [human]" as an internal source of interruption and "another person, computer, other animate object, inanimate object" as external sources of interruptions. While this widens our understanding of interruptions, I further break down the concept of interruption sources, to understand the term interruption for the scope of this work (cf. Figure 6).

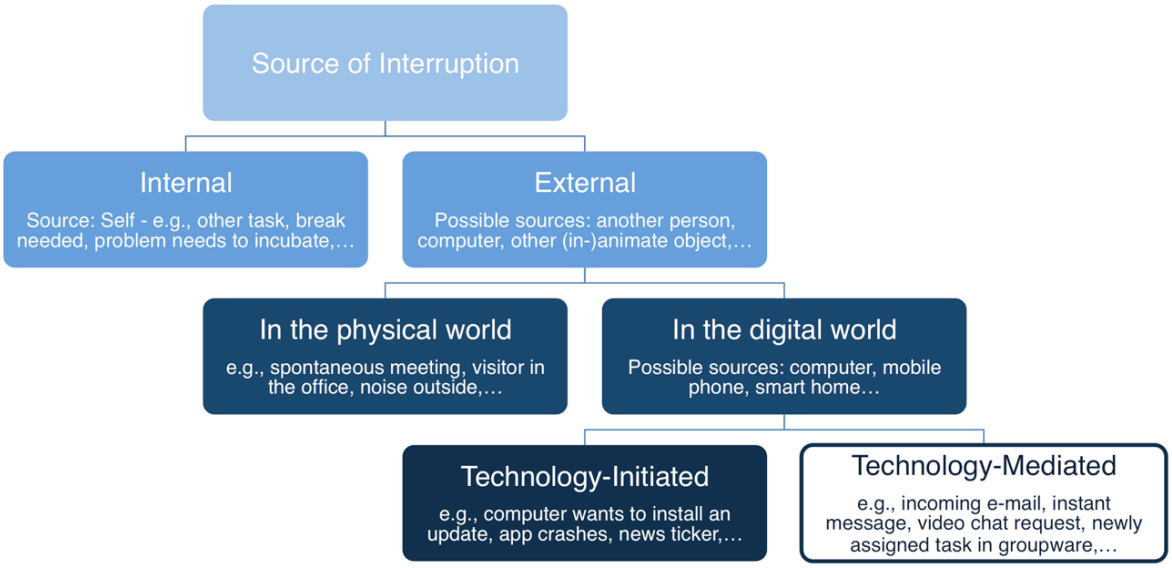

Figure 6. A hierarchical clustering of interruption sources.

In this work I am foremost interested in external interruptions. I further focus on interruptions that are primary in the digital realm and not in the physical realm, as an additional level of distinction. This brackets out interruptions due to conditions in the physical environment such as: somebody coming by the office for a short chat, noise outside the window that requires interrupting the work to close it, or being invited to a spontaneous meeting by a co-worker knocking on the office door. In the digital realm almost all interruptions can be accounted to what is called *notification systems*. The term *notifications system* [McCrickard *et al.* 2003b] subsumes hard- and software systems that deliver information to users, to satisfy their multitasking information demands. In this context of notifications systems McCrickard et al. [2003a, p. 551] define interruption "as an event within the notification system prompting transition of attention focus from a primary task to a notification".

Examples for such events can be found on classic desktop computers and mobile phones, which prompt users to react to incoming emails or instant

messages, provide information on news, stocks, and weather, or remind us of calendar events—but also on status updates of the devices themselves and their software (e.g., software updates or nearby WiFi networks). In the ubiquitous computing context in-vehicle information systems in our cars, ambient displays in our smart homes, or pervasive displays in the environment, notify us on various aspects—and thus interrupt us in our primary activity—whether it is that our car soon needs an inspection or that a parcel is delivered to our front door. What we already can see from these examples is that basically two different types of initiators are behind those messages. Emails, instant messages or the parcel delivered to our door usually are the results to social entities that initiate or continue some sort of social interaction with us. Others, like the weather update, the birthday reminder, or the information on nearby networks are initiated by devices and services, based on some scheduled events or information becoming available or being processed in the moment. Therefore, a last distinction I draw is whether the interruption is *technology-initiated* or *technology-mediated*. As this work is concerned with CMC, I am centrally interested in interruptions that are caused by people initiating or continuing communication through computing technology—hence computer-mediated interruptions.

We now can imagine that—due to interruptions—the work of information workers' becomes more and more fragmented [Mark *et al.* 2005]. Therefore, the influence of interruptions has been researched for different workplace environments from managers (e.g., [Mintzberg 1975]), over programmers (e.g., [Parnin & Rugaber 2011]), and nurses (e.g., [Kalisch & Aebersold 2010]) to pilots (e.g., [Latorella 1998]). Especially IM as a source of interruption in the workplace was repeatedly a central theme of studies [Cutrell *et al.* 2000; Czerwinski *et al.* 2000a; Garrett & Danzinger 2007]. But also other kinds of technology-induced interruptions, e.g., through notifications in mobile contexts have been researched [Fischer *et al.* 2011; Fischer *et al.* 2010; Pielot *et al.* 2014; Sahami Shirazi *et al.* 2014]. But, as new technology increasingly blurs the boundaries between work and home [Lindley *et al.* 2012] it challenges people to come up with new ways to negotiate those boundaries in everyday life [Nippert-Eng 1996]. Personal communication devices more and more lead to a state in which different spheres of daily life intertwine, thus requiring us to manage interactions "increasingly spontaneous" [Aoki & Woodruff 2005, p. 181] in different contexts, as for example work and home. Yet, research on interruption management in HCI and CSCW is foremost concerned with the influence of interruptions on individual tasks and task performance [Harr & Kaptelinin 2007] in working environments. Mostly it is concerned with the analysis of the benefits of the utility of provided information versus the cost associated with consuming the information in a specific context and manner [Roda & Thomas

2006]. So far, only few researchers looked at the effects of interruptions in private life [Tolmie *et al*. 2008], as for example in home-life [Nagel *et al*. 2004; Takemae *et al*. 2007].

Outside of the work context, other effects of interruptions—than the sole economical view on its influence on task performance—become relevant. Here, effects of interruptions on the general physical and mental wellbeing and social life of the people are central concerns. We learned in our discussion of privacy that people have a desire for intimacy, for undisturbed moments with people that are close to them. People have a basic need for "extended, isolated periods of reflection" [Jett & George 2003]. We learned about Westin's [1967] functions of privacy. Two of them—"emotional release" and "self-evaluation"—directly refer to peoples' need of times where the access to the self is limited. Interruptions in these phases of retreat can cause emotional impairments, like stress or frustration [Roda 2011]. Around the turn of the century—in a phase of increased urbanisation—Simmel identified a privacy function he called *reserve*. He describes it as a reaction to a steadily increasing number of contacts which sometimes is interpreted as a blasé attitude: "Wenn der fortwährenden äußeren Berührung mit unzähligen Menschen so viele innere Reaktionen antworten sollten, wie in der kleinen Stadt, in der man fast jeden Begegnenden kennt und zu jedem ein positives Verhältnis hat, so würde man sich innerlich völlig atomisieren und in eine ganz unausdenkbare seelische Verfassung geraten." [Simmel 1903, p. 195] (in engl.: "If the unceasing external contact of numbers of persons in the city should be met by the same number of inner reactions as in the small town, in which one knows almost every person he meets and to each of whom he has a positive relationship, one would be completely atomized internally and would fall into an unthinkable mental condition" [Simmel 2002, p. 15]). The same effects of excessive demands to interact and communicate can be seen today, as a by-product of today's information technologies, requiring people to get their minds off in an interruption-free environment. People need means for *reserve* in the information age.

Other researchers agree that limiting the focus of interruption research on task performance is to narrow. Harr [2007] identifies a social component of interruption, and arguments for widening the scope of interruption research in HCI and CSCW by taking into account the social dimension of interruption. He argues that research should not only take into account the interrupted individual and the single task, but also should take into account at the effects of the interruption on the interpersonal relationship of interrupter and interruptee, persons in the vicinity, and the communication and collaboration with third parties. Schiele and Kern [Kern *et al*. 2004; Schiele & Kern 2003] distinguish between *personal* and *social interruptibility* for the area of context-aware notification systems. While the first is concerned with the interruption of the

person itself (e.g., the person is concentrated on a task), the second is concerned with the environment of the person (e.g., the person is in a meeting). With this two-dimensional interruptibility space [Kern *et al.* 2004], they provide different examples for situations that affect different permutations of personal and social interruptibility. For example, attending a lecture results in low personal and social interruptibility while being in a bar typical results in high personal and social interruptibility. Driving a car is given as an example for low personal but high social interruptibility while sitting in a boring talk is given as an example for high personal but low social interruptibility.

Interruptions can lead to mistakes and stress [McFarlane & Latorella 2002], anxiety, and annoyance [Bailey *et al.* 2001], as well as other negative side effects, but also have the potential to be beneficial for the current work [Harr & Kaptelinin 2007], for example when they come along with timely and helpful information for the current task [Hudson *et al.* 2002; O'Conaill & Frohlich 1995]. The result of an interruption "can either be an unwanted distraction from an important task or attraction to valued information" [McCrickard *et al.* 2003a, p. 551]. Other positive effects of interruptions are that interrupted tasks are remembered better (based on the Zeigarnik Effect [van Bergen 1968]) or that they facilitate the performance on simple tasks [Speier *et al.* 1997]. Accordingly, managing interruptions is a constant cost/benefit trade-off [Hincapié-Ramos *et al.* 2011]. This trade-off can also be found in the *connectivity paradox* [Fonner & Roloff 2012; Leonardi *et al.* 2010], where teleworkers use computer-mediated communication technologies to achieve social presence, while at the same time minimising the beneficial effect of telework, the reduction of interruptions. Interruptions are seen as an unwelcome side effect when new computer or communication technology is introduced, as for example stated here: "interruption is an unavoidable cost of adding the telephone to" a workplace [McFarlane & Latorella 2002, p. 3]. Balancing between productive phases of quiet and interruption-free work and the openness to opportunistic and informal interactions is one of the challenges of modern workplaces [Lai *et al.* 2003]. The issue is to find "the balance between entertaining useful interruptions and avoiding distracting ones" [Hudson *et al.* 2002, p. 103].

To cope with this challenge, people developed multifarious interruption management techniques social practices, policies and tools to reduce interruptions [Liebowitz 2011], as well as strategies and coping mechanisms to resume after interruptions (e.g., [Parnin & Rugaber 2011]). However, especially in the field of notifications systems, there is still a vast potential to provide better tools to enable people to manage their interruptions.

To conclude, I showed that interruptibility—a person's receptivity to interruptions—is greatly depended on many factors including the current task, the source and the content of the interruption, the relationship to the

interrupter, and so forth. My analysis of the terms interruptions and interruptibility revealed parallels to the notions of privacy and seclusion—also interruptibility is very dynamic and largely dependent on the context and the involved persons. One conclusion we can make is that humans are not interruptible or non-interruptible per se, but selectively to different persons and their matters. In order to manage their interruptibility in the area of computer-mediated communication—which is central to this work—people need means to operationalise and express their interruptibility towards others. In the Following we finally look at the concepts of *availability* and *presence*, which allow persons expressing their receptivity to interruptions to others.

2.1.2.2 Availability and Presence

While the terms *availability* and *presence*[1] essentially describe different concepts, in the fields of HCI, CSCW, and CMC they are often used synonymously or at least as an idiomatic binomial pair. And indeed, it makes sense to regard *availability* and *presence* as a combined concept in the context of this work. In the Following I further explore these two concepts and relate them to our preceding analysis of *interruptibility*. But again I start with a first look at the denotation of these two words and their general linguistic usage. The dictionary Merriam-Webster [2012a] defines *availability* as: "

1) the quality or state of being available
2) an available person or thing"

As the two definitions of *availability* use the word *available* in the definition, we accordingly need to look at the definition of the adjective 'available' [Merriam-Webster.com 2012b]:"

: easy or possible to get or use
: present or ready for use
: present and able or willing to talk to someone

1) archaic : having a beneficial effect
2) valid —used of a legal plea or charge
3) present or ready for immediate use [...]
4) accessible, obtainable [...]
5) qualified or willing to do something or to assume a responsibility [...]
6) present in such chemical or physical form as to be usable (as by a plant) [...]"

---

[1] Please note that the term telepresence [Sheridan 1994] (also synthetic presence, virtual presence, ego presence and hence sometimes also just briefly presence [Draper et al. 1998, p. 354]), which is also common in the fields of HCI, CSCW and CMC, refers to a different concept than what is of matter in this work; telepresence is concerned with the illusion of a user in a synthetic environment that leads to the "sense of being physically present with virtual objects" [Sheridan 1992, p.6], for example at a remote site.

By combining aspects of these two definitions we can say that *availability* is the 'state of being present and able or willing to talk to someone'. We also see that according to the dictionary entry on *available*, which uses the word *present* several times, *presence* is an integral aspect of *availability*.

Accordingly we are looking at the definitions of the word *presence* [Merriam-Webster.com 2012c]:"

1) the fact or condition of being present [...]
2) a: the part of space within one's immediate vicinity
   b: the neighborhood of one of superior especially royal rank
3) archaic : company [...]
4) one that is present: as
   a: the actual person or thing that is present
   b: something present of a visible or concrete nature
5) a: the bearing, carriage, or air of a person; especially : stately or distinguished bearing
   b: a noteworthy quality of poise and effectiveness [...]
6) something (as a spirit) felt or believed to be present"

And at the definitions of the adjective *present [Merriam-Webster.com 2012d]: "*

: not past or future : existing or happening now
—used to say what someone or something is now
: at the particular place or event that is being referred to

1) now existing or in progress
2) a) being in view or at hand
   b) existing in something mentioned or under consideration
3) constituting the one actually involved, at hand, or being considered
4) of, relating to, or constituting a verb tense that is expressive of present time or the time of speaking
5) obsolete : attentive
6) archaic : instant, immediate"

Again, by combining both definitions, we can define *presence* as the "condition of being in view or at hand". From these two definitions, we can already see that *presence* to a certain extend affects *availability*. If a person we are trying to contact is generally "able or willing to talk" to us but not "in view or at hand", communication is impossible. Though, when we look at the two concepts from the perspective of interruptibility in the physical realm, the two concepts can be somehow told apart [Fogarty *et al*. 2004b]. For example, in an office environment, a colleague that is sitting in his office and momentarily involved in a conference call is present but currently unlikely to be available for a face-to-face meeting. Another colleague, who is currently not in the office, is not present. But with the office door left open and still being logged into the PC,

this colleague is likely to return soon and hence probably available for a meeting.

In the digital realm however, the two concepts are often intermingled, as they are often both represented in only one online state (e.g., the online state of an IM system). Harr and Wiberg [2008, p. 244] assert that "various kinds of technical support [...] often equals the notion of availability with the notion of presence", although they find this equalisation problematic. They differentiate between availability as "a state of mind (whether an individual is receptive for communication or not)" and presence as a concept that is concerned with "whether an individual is reachable via synchronous communication channels or not". [Harr & Wiberg 2008, p. 244]. Begole *et al.* [2004, p. 511] see *presence* as a part of *availability*. They identify two components that make up a person's general availability: presence and receptivity. They say a "recipient must both be *present* to receive the call as well as *receptive* to communication at that time"—a view that is congruent with my conclusions from the dictionary. They further differentiate between *device presence*, which can be seen as technical reachability (e.g., logged in to an IM network, mobile phone with network access), and *person presence*, which can be seen as actual reachability (i.e., a person is physically close to a communication device and is able to take notice of and react to incoming communication requests). Finally they define *receptivity* towards communication "as one's willingness to be interrupted." Hence, Begole et al. conceptualise *availability* as a product of *presence* and *interruptibility*. Lai et al. [2003] use the notion of *communication availability* referring to the receptivity towards what I previously called technology-mediated interruptions. Also Hudson *et al.* [2002] use the concept of interruption management to model personal *availability* as a function of one's *interruptibility*. Poikselkä and Mayer provide a more technological focussed definition of *presence* in their analysis of presence services for IP Multimedia Subsystems (IMS) and once more mix the concepts of *presence* and *availability*. They state that *presence* "can be seen as a user's status as perceived by others and others' status as perceived by the user" [Poikselkä & Mayer 2009, p. 139]. Thereby the status is dynamic and can contain different types of information including the status of the person or the person's devices, context- and location information, terminal capabilities, and preferred means for contacting the person (e.g., voice, video, instant messaging). In their definition of presence they explicitly include the aspect of a persons' availability and willingness to communicate. As well as the expressiveness, that allows a person to "to communicate to others when a person is able and willing to communicate as well as with whom and by what mean" [Poikselkä & Mayer 2009, p. 139].

What I found to be underrepresented in all these definitions is *availability* as a positive expression of a desire for communication. The perspective

represented in many of the definitions here, is that communication is an inevitable evil, and people only have the chance to manage when communication is least disruptive to them. On the other side many IM clients offer online states like "Skype Me" [Skype Limited 2011] or "Free for Chat" [ICQ LLC 2013], allowing to directly advertise a wish for social interaction. I think a characterisation of *availability* should consider this full spectrum.

To conclude, like before with *privacy* and *interruptibility*, we see several definitions and interpretations describing the same notions from different perspectives. They all highlight different aspects, whether that is the dynamic and fugacity of such concepts, or it is technical or social considerations, etc. Based on my derived definition of *availability* as the "state of being in view or at hand and able or willing to talk to someone" I see Begole *et al.'s* [2004] definition of availability in the field of technology-mediated communication as the most plausible. I also understand availability as the state of 'being present and receptive for technology-mediated communication'. This way *availability* can be seen as a positively connoted advertisement of ones' *interruptibility* in technology-mediated communication.

2.1.3 An Awareness Perspective on Presence and Availability

The term *awareness* arose in the early phase of CSCW research. It emerged in an endeavour to develop a better understanding of human work practices in order to support coordination in computer-supported cooperative work and since became a central topic. Various studies have been conducted, many models and frameworks were conceived, and various *awareness systems* [Markopoulos *et al.* 2009] were developed to provide adequate *awareness* to users—the aim was and is to support "effortless coordination" [Gross 2013]. Among these *awareness systems*, several were developed with the goal to provide *presence* and *availability* information to support interruption management. Therefore, a discussion of the concepts *presence* and *availability* cannot be complete without understanding the CSCW community's conception of *awareness*. Hence, in the Following I first provide a brief introduction into *awareness* in general. After that I continue with a focus on awareness for supporting *Presence* and *Availability* and I provide an introduction to the design space of awareness support systems such as *presence services* and *availability management systems*.

2.1.3.1 An Introduction to Awareness

Ethnographic workplace studies like the analyses of work in a London Underground control room [Heath & Luff 1992; Heath & Luff 1996] or an air traffic control [Harper & Hughes 1993; Harper *et al.* 1989] revealed that co-located workers elegantly and seamlessly align and arrange their work with their colleagues. Thereby acts of monitoring the activities of co-workers as well as

displaying aspects of the own work, which deem relevant for the colleagues, support this seamless interaction [Schmidt 2002]. The term *awareness* was conceived, to describe these coordinating practices. While in the physical realm "this information is ready at hand and group members can collect it naturally" [Gross *et al.* 2005, p. 324] in the digital realm—when cooperating over distance—this information gets lost and has to be retained with means of technological support.

In the mid-1980s the notion of *awareness* started to become one central research stream in the field of CSCW. An overview on awareness is provided in several reviews, surveys, and overviews [Gross 2013; Gross *et al.* 2005; Rittenbruch & McEwan 2009; Schmidt 2002; Schmidt 2011]. But with the field of awareness research steadily growing, reaching its peak in the 1990s, the notion of awareness became more and more an anamorphous term bloated with multitudinous meanings and attributions [Schmidt 2002]. As a result, the term was combined with other words, to easier refer to these different, specific conceptions and nuances of *awareness*, among which are: informal awareness, general awareness, peripheral awareness, group awareness, workspace awareness, social and task-oriented awareness, synchronous and asynchronous awareness, intentional and unintentional awareness, structural awareness, collaboration awareness, background awareness, passive awareness, reciprocal awareness, mutual awareness, etc. [Harr & Wiberg 2008; Rittenbruch & McEwan 2009; Schmidt 2002]. While a more detailed review of many of these notions is provided in [Gross *et al.* 2005, p. 324], discussing all of them is out of the scope of this work. In the Following we are concentrating on those notions of awareness that are the most relevant and established for the context of *availability* and *presence*.

One authoritative definition of *awareness* describes it broadly as "an understanding of the activities of others, which provides a context for your own activity" [Dourish & Bellotti 1992]. Gross *et al.* use the more distinguished term *group awareness* defining it "as consciousness and information of various aspects of the group and its members" [2005, p. 326]. Almost 10 years later Gross refines the definition of awareness as a "user's internal knowing and understanding of a situation including other users and the environment that is gained through subtle practices of capturing and interpreting information" [Gross 2013, p. 432].

As CSCW is rooted in supporting the eponymous *work* [Grudin 1999], naturally also *awareness research* cultivated around a vision to improve the support of people when cooperating on tasks in a work environment. Therefore, a great amount of awareness research is interested in supporting awareness on the state and process of work-related tasks and of the artefacts involved in this tasks, like documents, workspaces, etc. However, in order to support smooth

coordination among co-workers and colleagues, also awareness about their *activities*, *presence*, and *availability* in general is important, as the knowledge about these parameters facilitates smooth interaction (e.g., without interrupting someone in an improper moment). In the Following I solely focus on this perspective of awareness, and blank out other perspectives, as for example *workspace awareness*.

2.1.3.2 Presence and Availability in Awareness Research

Mediating information on *presence* and *availability* is one central aspect of awareness support. Berlage and Sohlenkamp [1999, p. 207] account for this perspective on *awareness* when stating that awareness allows to "check the availability and accessibility of others, providing a base mechanism for establishing communication". Prinz [1999] explicitly distinguishes between task-oriented awareness and social awareness. The first is concerned with awareness on activities that are pursued in order to execute a specific shared task. The second, social awareness, is concerned with "information about the presence and activities of people in a shared environment" [Prinz 1999, p. 392]. Fogarty exemplifies the kind of information provided through *social awareness* support with the information we obtain in the real world when someone is walking by our office door [Fogarty *et al.* 2004b]. Others use the notion of *informal awareness*, also refered to as peripheral awareness or general awareness [Rittenbruch & McEwan 2009]. *Informal awareness* "involves knowing who's currently around, whether they're available or busy, and what sort of activity they're engaged in" [Gutwin *et al.* 1996, p. 205] and thus is an enabler for informal interaction. *Informal awareness* provides "knowledge of presence, activity and availability" [Rittenbruch & McEwan 2009, p. 33] of others. Because of its subliminal and concurrent nature Rittenbruch [2009] also describes this form of awareness with the term *background awareness*. Along these lines, Pedersen and Sokoler differentiate [1997] between *intentional awareness* and *unintentional awareness*. While *unintentional awareness* is maintained on others with no specific purpose, the *intentional awareness* is used to actively assess if a person is currently available for communication. In the Following, when using the term *awareness*, I am referring to the types of awareness that relates to presence and availability.

In order to provide such awareness, different technological solutions in form of *awareness support systems* have been developed. Each aims to collect and share relevant information, which allows people to assess an ideal moment for contacting someone. These so-called awareness cues "are computer-mediated, real-time indicators of people's undertakings, whereabouts, and intentions" [Oulasvirta 2009, p. 125]. Again, also awareness support systems have been developed that explicitly support the cooperation on a task or awareness about a

shared artefact in a work environment (e.g., in a shared editor). Here we are singularly interested in systems supporting awareness that allow assessing the *presence* and *availability* of a person. Gross [2013] clusters awareness systems from an end-users' perspective into two groups: coexistence awareness and cooperation awareness. Coexistence awareness is concerned with systems "providing users with information on each other's presence and availability" [Gross 2013, p. 436], and hence are of concern for this work.

2.1.3.3 Presence Services and Availability Management Systems—Conceptions, Chances, and Challenges

In the CSCW literature, many different terms can be found for conceptualising such systems—or aspects of such systems—that provide *presence* and *availability awareness*. Among the terms that can be found are: *media spaces* [Bly *et al*. 1993], *notification systems* [McCrickard *et al*. 2003b], *attention-aware systems* [Bailey & Konstan 2006; Roda & Thomas 2006] or *attentive user interfaces* (AUI) [Vertegaal 2003; Vertegaal *et al*. 2006], *availability management systems* [Diacakis & Cohen 2002; Harr & Wiberg 2008] as well as *availability sharing systems* [Hincapié-Ramos *et al*. 2011] and *presence services* [Day *et al*. 2000; Open Mobile Alliance Ltd 2014]. In the Following I explore these notions, look at the different underlying conceptions, and discuss their design space. Specific systems and prototypes however are discussed later, in an analysis of related work in section 2.4.

As a good entry point, Harr and Wiberg, who studied the use of technical support systems for availability management, provide a helpful categorisation. First, they define *availability management* as "the ways in which a person signals to other persons in the surroundings (including also online contacts) if he/she is open to communication or not" [Harr & Wiberg 2008, p. 245-246]. Then, they differentiate between two general approaches for managing one's availability (visualised in Figure 7): the *explicit approach* and the *implicit approach*. They further subdivide the implicit approach in systems that present *awareness information and such* systems that use *automated calculations* of availability states.

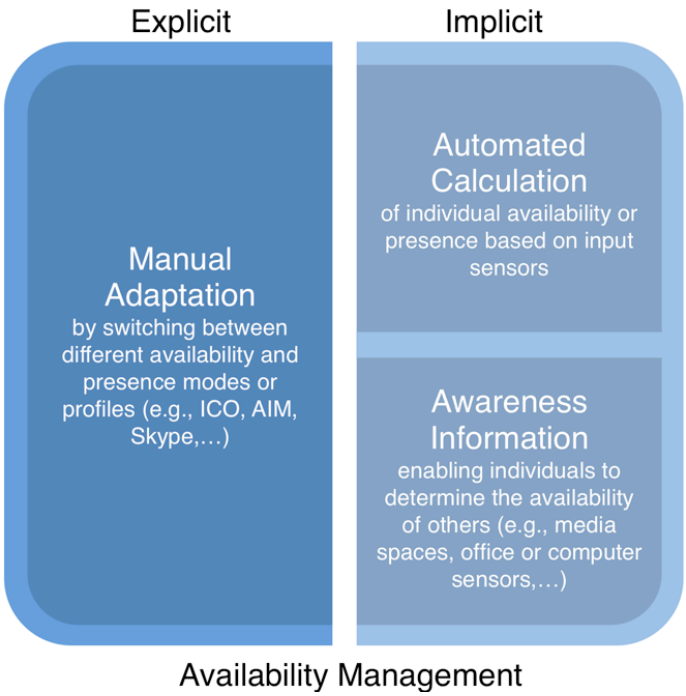

Figure 7. A visualisation of Harr and Wiberg's conception of explicit and implicit approaches to availability management.

In detail, the categorisation is as follows:

- *Explicit availability management* (also *proactive availability management* [Begole *et al.* 2004]) relies "primarily on a user's explicit statements or actions to reveal presence and availability" [Horvitz *et al.* 2002, p. 224]. That is, users are manually changing their *availability and presence indicator* [Oulasvirta 2009], often by selecting from a few different options in a menu. The most prominent examples for this approach are of course classic IM clients like ICO [ICQ LLC 2013], Skype [Skype Limited 2011], etc. However, manually adapting one online status and status message were repeatedly found to only offer insufficient support for users to find an appropriate moment for initiating communication [Harr & Wiberg 2008; Mileswki & Smith 2000; Voida *et al.* 2002]. Further the manual adaptation adds an overhead of additional work to the user [Grudin 1994; Mileswki & Smith 2000].
- *Implicit availability management* on the other hand does no rely on a direct interaction of the user to manage the availability. Of course, some user interaction is still required to setup or configure such systems. According to Harr and Wiberg, the implicit approach again can be divided into two types. A analogous distinction was made by Begole and Tang [2007] in their discussion of *human* vs. *machine interpretation* of *availability cues*. Harr and Wiberg distinguish between:

- The *presentation of awareness information* aims at providing users with "context information they need to assess availability, as in face-to-face communication where both parties engage in negotiating availability" [Begole *et al.* 2004, p. 512]. *Media spaces* [Bly *et al.* 1993]—basically permanent audio/video links that connect two or more places or people—were one of the earliest advances in this direction. Others looked at more abstract representations by sensing and collecting bits of information about a users whereabouts, activities, etc. and sharing them in form of events with others [Schmidt 2002]. The goal of these abstractions was two-fold—to decrease the disclosure of private information and provide the information in a way that can be processed more quickly.
- *Automated calculation of availability* relies on machine learning to build predictive models, often based on the users' past behaviour. The idea is to "present an inferred abstraction of the prospective recipient's availability" to provide users "with the salient information (the recipient's availability) without exposing more details about the recipient's context than necessary" [Begole *et al.* 2004, p. 512]. While often the notion of attention-aware systems or Attentive User Interfaces (AUI) is found in this context [Bailey & Konstan 2006; Horvitz *et al.* 2003; Roda & Thomas 2006], those systems often go beyond *presence* and *availability* management for communication. This means, although such AUI are also centred on "statistically modelling attention" based on "the urgency and relevance of information or actions" and the "context of current activity" [Vertegaal 2003, p. 32] their area of application goes beyond attention towards technology-mediated notifications (i.e., availability for communication) but also accounts for technology-initiated notifications (e.g., interruptibility to system notifications).

Hincapié-Ramos use the term *availability sharing system* to describe technologies that "help collaborators identify the most appropriate (and least costly) times to initiate interactions" [2011, p. 85].

A more technical description can be found in the IETF RFC 2778 [Day *et al.* 2000]. There, a model for *presence* sharing is described, defining a *presence service* as following (cf. Figure 8): "The PRESENCE SERVICE has two distinct sets of "clients" (remember, these may be combined in an implementation, but treated separately in the model). One set of clients, called PRESENTITIES, provides PRESENCE INFORMATION to be stored and distributed. The other set of

clients, called WATCHERS, receives PRESENCE INFORMATION from the service."

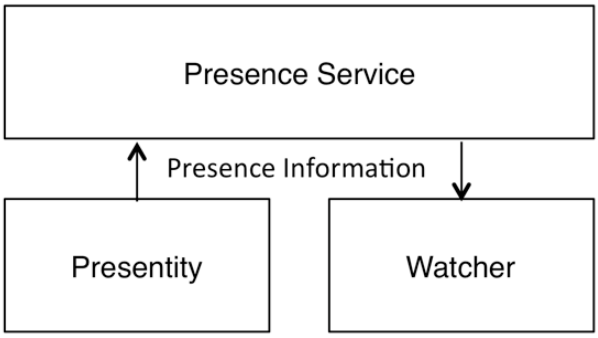

Figure 8. Overview of Presence Service adapted from [Day *et al.* 2000].

Based on this description, different stakeholders have tried to implement unified and standardised presence services. For example, the *3rd Generation Partnership Project* (3GPP) and *Open Mobile Alliance* (OMA) developed the specifications for the *OMA Instant Messaging and Presence Service* (IMPS) [Open Mobile Alliance Ltd 2014] enabling the exchange of presence information between mobile devices and services as well as instant messaging services. Thereby the OMA IPMS leverages on [Poikselkä & Mayer 2009] the IETF-Standard *SIP for Instant Messaging and Presence Leveraging Extensions* (SIMPLE) [Internet Engineering Task Force 2011], which is an extension of the *Session Initiation Protocol* (SIP) [Rosenberg *et al.* 2002] with presence information. Finally, the *eXtensible Messaging and Presence Protocol* (XMPP) [Saint-Andre 2011a]—which is discussed in more detail later—also represents a standardisation effort for a presence service.

While most of such systems provide the information synchronous, in real-time, a variant of such systems exist that do not present the current availability or awareness about current activities, but rather a "visualization of historical patterns" [Hincapié-Ramos *et al.* 2011, p. 85], as for example in *Awarenex* [Begole & Tang 2007] or *LocaRhythms* [Fetter & Gross 2009b]. Those systems aim to generate a shared understanding of the availability of an individual.

The design space of presence services and availability management systems is continuously analysed, debated and refined in the CSCW community. General design goals for *presence* and *availability* support in *awareness systems* were established early. For example, the design goals of the *DIVA Virtual Office* environment published 1994 included—among others—the provision of synchronous awareness functions that allow users to "ascertain a co-worker's availability for contact", "control their own level of availability", and "control the information about themselves which is broadcast to others" [Sohlenkamp & Chwelos 1994, p. 332].

Hincapié-Ramos *et al.* [2011] delivered an extensive analysis of the design space of availability-sharing systems based on a review of existing systems, the top-down review of existing taxonomies in literature on human interruption, and the bottom-up analysis of the established design space in related areas. They identified six dimensions: *abstraction* (of the collected and shared data), *presentation* (of the availability data, e.g., as icon, gradual fade, video), *information delivery* (i.e., when the information is delivered), *symmetry* (of the reciprocity of information sharing), *obtrusiveness* (of the presentation e.g., focal or peripheral), and *temporal gradient* (i.e., the time scale from historical availability data to predicted data). Further they identified for the different characteristic values of the six dimensions whether they are beneficial for the cost/benefit trade-off of the interrupters or of the interruptees.

One central theme in the design consideration is the *dual trade-off* between privacy and awareness on the one hand, and between awareness and disturbance on the other [Hudson & Smith 1996]. Davis and Gutwin [2005] use the privacy-related terms *confidentiality* and *solitude* to refer to the information control processes with awareness servers. As we learned earlier, thereby confidentially is the control over information that is going out, and solitude is the control over incoming information. The ability to control outgoing as well as incoming information is important to satisfy both functions of privacy.

Another theme is the amount of *configuration effort*. Richards and Christensen [2004] ask "how much effort" one is "willing to put into controlling what can be seen by whom". Current systems that are based on rules become complex quickly and also fail to adjust to the different nuances of the respective current situation of a user. Others advocate for a "need for a lightweight mechanism to control privacy" [Lee *et al.* 1997, p. 389]. They go on stating that systems should facilitate "the tight coupling between the means to change privacy and the means to obtain feedback that privacy is attained". This idea of *feedback and control* was already formulated by Belotti and Sellen [1993] in the field of UbiComp. Especially when it comes to configuring who receives what information, Barkhuus [2012, p. 371] claims that "digital technologies are not very good at managing groups of relationships, except laboriously".

Some research analysed the *interpretation of availability states* on the receiving end to formulate design recommendations. For example, in a study of mobile instant messaging practices 93% of the users would send a message to a contact despite their state of being "busy" or "away". Patterson *et al.* [2008, p. 67] concluded from that finding that the state is often interpreted as an "expected response time". Teevan and Hehmeyer [2013] even found for an enterprise communications tool signalling phone availability, that people that are stated busy are significantly more likely to answer the phone. Their conclusion is that, that callers mainly ignore the availability states when the information is

important, thus incoming calls are perceived as more important when the user is currently busy. Another point for critic is the *binary notion* of availability often found in CSCW. Hudson argues [2002, p. 103] that this does not account for the "continuum in which managers typically work" and hence often hinders the potential "benefits of being interrupted." Others are not content with the simplified view of presence, which has a tendency "to be confined to activity within a single application" [Richards & Christensen 2004, p. 86], and argue for a richer, aggregated form of awareness information. Davis and Gutwin [2005] also advocate for new means in awareness servers that allow for a richer range of differentiation for information disclosure. In respect to the support of impression management [Birnholtz *et al.* 2012a] in communication and collaboration technologies Birnholtz *et al.* [2012b, p. 33] argue that current technology does "not provide good support for managing unavailability and inattention."

As a last thought, Erickson and Kellogg [2000] use the notion of making systems *socially translucent*. A notion that hints, that awareness systems can be more than only tools transferring information in order to support instrumental activities like collaboration and coordination [Vetere *et al.* 2009]. They also have a chance to emotional enrich social interaction and support the feeling of human connectedness [Agamolis 2005]. Seeing somebody online fulfils functions beyond simply knowing the person is available for communication right know. It can inform us that an old friend is still awake late at night, or a loved one has just safely reached a travel destination online and the cell-phone reconnected after a long distant flight. For example, Chen *et al.* aim for a more "rich emotional communication" of awareness information with their *ComSlippers* [Chen *et al.* 2006, p. 370].

2.2 Instant Messaging

In the Following the concept of Instant Messaging (IM) is introduced, accompanied by a brief historical overview of relevant systems, services and technologies, and a summary of important work from the HCI and CSCW community. This section concludes with a statement about the relevance of IM and IM research in the context of this work.

The Merriam Webster online dictionary very broadly defines Instant Messaging (IM) as "a means or system for transmitting electronic messages instantly" [Merriam-Webster.com 2013a]. Nardi *et al.* [2000, p. 80] in their seminal work on Instant Messaging provide a more detailed definition of IM as "near-synchronous computer-based one-on-one communication" based on a "dyadic 'call' model". By comparing IM to telephone calls, this definition adds two essential points: a) in the classical sense IM is mostly concerned with the

message exchange between a pair of two users and b) the users are notified about new messages that are directed to them. This differs IM from chat rooms, where users are taking part in one or more conversations by monitoring these conversations for messages that are directed to one self. In any case, IM is a form of computer-mediated communication (CMC) [Baron 2013, p. 135]. A more complete definition of *IM systems* however has to encompass one more feature: a permission based subscription model in combination with a presence service as mentioned in the definition of the IETF Model for Presence and Instant Messaging (RFC 2778) [Day *et al.* 2000]: "A presence and instant messaging system allows users to subscribe to each other and be notified of changes in state, and for users to send each other short instant messages". The three key properties that characterise an instant messaging system in the scope of this thesis thus are the following:

- Allows for the almost *real-time, dyadic exchange of text messages* between two users; the users are notified in some form about new incoming messages.
- A *subscription based permission model* allows to specify users, which persons are allowed to send them messages; the subscription process aims at achieving a reciprocal approval, resulting in the contacts being added to each others lists of approved contacts (i.e., buddy list or roster).
- An additional *presence service* provides information about the current status and availability of the approved contacts, thereby allowing assessing adequate moments for establishing communication or to estimate when a message has reached its recipient.

While current IM clients often offer many additional features like audio and video connections, multi-user chats, file exchange or screen-sharing possibilities, these three properties characterise the core aspects of the IM concept, that are important in the scope of this work. In order to see the development of this concept, in the Following the history of IM is briefly discussed.

The basic concept of sending instantaneously messages to logged in users came with the advent of the first multi-user operating systems, like the *Compatible Time-Sharing System* (CTSS) developed at the MIT. The CTSS's WRITE command enabled logged in users to send short messages to other users logged in to the same computer [Crisman 1969, p. 4 in section AH 2.19]. This "interconsole communication" feature also included a minimal permission system that allowed to block specific users from writing messages to oneself—a functionality replicated in the *send_message* utility of the *Multiplexed Information and Computing Service* (MULTICS) and later the UNIX *talk* text chat program. In the following years, the Zephyr Notification Service [DellaFera *et al.* 1988],

BitNet Relay (later Internet Relay Chat (IRC) [Oikarinen & Reed 1993]), the chat rooms of so called Bulletin Board Systems (BBS) and the first major commercial online chat service—the CompuServe CB Simulator launching 1980—made the exchange of text messages available to a broader user base. Although, the focus of those systems was on connecting multiple users that joined chat rooms (comparable to teleconferencing), in comparison to the dyadic message exchange (comparable to phone calls) that is the core of IM.

The release of the text-based Instant Messaging service ICQ [ICQ LLC 2013] in the year 1996 by Mirabilis is widely regarded as the start of Instant Messaging in it's modern and current form, with a GUI client, features like buddy lists and online presence, and the first to reach a broad user base. Other services quickly followed like the AOL Instant Messenger (AIM) [AOL Inc. 2014] starting 1997, that also used the Open System for CommunicAtion in Realtime (OSCAR) protocol, the Yahoo Messenger (YM) [Yahoo! Inc. 2014] starting 1998 and the MSN Messenger Service [Microsoft 2014] (later .NET Messenger Service, Windows Live Messenger Service, Messenger) starting 1999—the later both using own proprietary protocols.

In 1999 with, the start of Jabber, the first open source IM application with an accompanying open standards protocol and free accounts via the jabber.org server was available [XMPP Standards Foundation 2014]. The Jabber protocol also is the origin of the IETF formalisation effort that lead to an Internet standard for instant messaging and presence technology in form of the protocol XMPP [Saint-Andre 2011a]. XMPP till now finds a widespread adoption and is also used in major commercial services like Facebook Chat [Facebook Inc. 2014] and the discontinued Google Talk (now replaced by the proprietary Google+ Hangouts) [Google 2014]. In the year 2003 Skype was released. The free commercial peer-to-peer service offered high quality voice chats together with IM and presence capabilities, and quickly attracted a wide user base, when adding high quality video chats. Since then, many major IM services and clients added options for audio and video chats. In the last years, Instant Messaging and Presence services spread from the personal computers to other types of devices, including mobile phones, game consoles, TV sets, etc. Multi-protocol IM clients like Trilian [Cerulean Studios 2014] enable users to stay in contact with friends on different services, without the need for running several clients. And finally, new services like WhatsApp [WhatsApp Inc 2014] and Apple iMessage [Apple Inc. 2014] further blend Instant Messaging and Presence concepts with those of SMS [Thurlow & Poff 2013].

IM is used in different contexts, spanning private life [Grinter & Palen 2002; Grinter *et al.* 2006; Li *et al.* 2005; Lowry *et al.* 2001; Patterson *et al.* 2008] and work life [Bradner *et al.* 1999; Fussell *et al.* 2004; Handel & Herbsleb 2002; Herbsleb *et al.* 2002; Isaacs *et al.* 2002b; Nardi *et al.* 2000]. Accordingly, the

effects of its usage have been the subject of work in the fields of HCI and CSCW, but also in other contexts as information systems, communication studies, etc. Among the specific aspects of IM usage that have been intensively researched in the HCI and CSCW community are for example: the influence of interruptions on task performance (e.g., [Cutrell *et al.* 2001; Cutrell *et al.* 2000; Czerwinski *et al.* 2000a; Czerwinski *et al.* 2000b; Garrett & Danzinger 2007]); privacy attitudes and practices (e.g., [Patil & Kobsa 2004; Patil & Kobsa 2005]) as well as general norms and behaviours around the usage of IM (e.g., [Avrahami *et al.* 2008; Fetter & Gross 2009a; Hancock *et al.* 2009; Nardi *et al.* 2000]). Based on this research, numerous systems and solutions have been proposed (e.g., [Tang & Begole 2003]) in the HCI and CSCW community to improve the utility of IM for its users.

In the CSCW research community, the properties of IM made and still make it an ideal test bed for exploring the many facets of awareness. The combination of a permission model, an underlying presence service, and the interruptive nature of the notifications inspired many researchers to extend on IM's capabilities to explore new awareness mechanisms in a plenitude of prototypes—which I discuss in detail in section 2.4. This also results in the fact, that the ideas and concepts underlying IM and the research around IM inform new paradigms in ICT and CMC. For example, Unified Communications (UC) systems and Real-Time Collaboration (RTC) systems [Riemer & Frößler 2007] aim to integrate different communication channels spanning devices (i.e., phone, fax, desktop and mobile computers) and media (i.e., text, audio, video), making a person reachable under one universal phone number "which finds them wherever they wish to be found" [Riemer & Klein 2009, p.267] through the provision of presence information.

2.3 Ubiquitous and Mobile Computing

After the in-depth look at the concept of *Availability* and *Presence*, followed by a brief analysis of the term *Instant Messaging*, I now complete the introduction of concepts from the title of this thesis by discussing the terms *Ubiquitous Computing* (*UbiComp*) and *Mobile Computing*. My main aim thereby is—after their general introduction—to elaborately highlight only the specific aspects of these two concepts that are of particular interest for this thesis.

Mark Weiser [1991] coined the term *Ubiquitous Computing*. The underlying thoughts and ideas can be regarded as the prime result of his work as director of the computer science laboratory at Xerox PARC. He envisioned *UbiComp*—after the era of the mainframe computers (i.e., many persons use one computer) and the era of the personal computer (i.e., one person uses one computer)—to mark the third wave in computing, where for one person many computers would be

available to use [Want 2009]. The term *ubiquitous* hinted at an understanding that in this era computers are more and more "seeming to be [...] everywhere" [Merriam-Webster.com 2014]. In 1988 Weiser described *UbiComp* as "invisible, everywhere computing that does not live on a personal device of any sort, but is in the woodwork everywhere" according to his personal homepage [Weiser 1996]. As superficial this definition might be, its notion fuelled and pervaded multifarious areas in the scientific as well as the practical progression of computing technology [Abowd 2012]. As research field *UbiComp* is a quite unusual contestant on the computer science grounds, as it "encompasses a wide range of disparate technological areas brought together by a focus upon a common vision" [Dourish & Bell 2007, p.133].

On the other side, coming up with a comprehensive definition for *Mobile Computing* is rather difficult. The perception of what the core of *Mobile Computing* is varies through the different disciplines. One apparent, tautological definition, applicable from almost any perspective, is the following: *Mobile Computing* is "a computing environment of physical mobility" [Talukder & Yavagal 2006, p. 5]. *Mobile Computing* as a field emerged as a discipline that was primarily technological driven by advances in miniaturisation and connectivity [Kjeldskov 2014]. However, the aspect of portability of devices and providing access to information anytime and anywhere [Perry *et al*. 2001] made it quickly become a very user-centred endeavour.

From an HCI perspective, *UbiComp* and *Mobile Computing* in big parts leverage on a similar research agenda [Abowd 2012]. This let to a point, where the concepts behind these terms are partially watered down and diluted by multitudinous ideas and notions—all of which vaguely describe similar things. Therefore, in the introduction of Chalmers' book on engineering context-aware systems [Chalmers 2011] the terms *Ubiquitous Computing*, *Pervasive Computing* [Satyanarayanan 2001], *Ambient Computing* [Encarnação & Aarts 2006], and *Disappearing Computing* [Streitz & Nixon 2005] are regarded as coequal terms, with only subtle differences. Chalmers goes on stating that *Mobile Computing* and *Sensor Networks* [Dargie & Poellabauer 2010] can be treated as "significantly overlapping fields" [Chalmers 2011, p. 4]. Also Jones and Mardsen [Jones & Marsden 2006, p. 54] draw this connection between *UbiComp* and *Mobile Computing* by pointing out how *UbiComp* has "fertilized much of the research and development in mobile computing". Other terms like *Nomadic Computing* [Kleinrock 1996], *Wearable Computing* [Mann 2013], *Ambient Intelligence* [Encarnação & Aarts 2006], *Invisible Computing* [Borriello 2000], *Physical Computing* [Igoe & O'Sullivan 2004], *Situated Computing* [Hull *et al*. 1997], *Context-Aware Computing* [Schmidt 2013], *Proactive Computing [Tennenhouse 2000], Cyber-Physical Systems* [Wolf 2009], or *Internet of Things* [Gubbi *et al*. 2013] are further added to this mix—making them more and more undistinguishable

as far as their definitions become a matter of opinion and taste for different research communities. In order to come up with a valid understanding of these terms' relevance for this thesis, I go on by explaining the particular aspects of the underlying paradigms, which are of interest for this work.

Hence, the five aspects from these two research fields I identified as central for this thesis are *Context-awareness* and *Context-aware Adaptation*; *Sensing, Actuation, and Inference; Nomadicity* and *Nomadic Users;* the notion of *Calmness* and *Social Embeddedness*; as well as the conceptions of *Privacy, Security, and Trust*:

- *Context-awareness* and *Context-aware Adaptation*: While the concept of *context-awareness* is often considered to be grounded in *UbiComp* research at Xerox PARC, the term first appeared in a paper in the field of *Mobile Computing*—at the 1994 Workshop on Mobile Computing Systems and Applications [Schilit *et al.* 1994]. In the Abstract the authors describe their idea for systems "that examine and react to an individual's changing context". Thereby context—according to their definition—consists of three aspects: "where you are, who[m] you are with, and what resources are nearby" [Schilit *et al.* 1994, p. 85]. While, over the course of time, various nuanced and refined definitions of *context* were given (e.g., [Chalmers 2004; Dey *et al.* 2001; Schmidt *et al.* 1999; Winograd 2001]) most of them take into account more or less the same dimensions, as there are: location, identity, time and activity [Gross & Specht 2001]. The ways in which a context-aware system reacts to the context also can be diverse. Dey and Häkkilä provide a comprehensive definition, by stating that a context-aware system "uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task" [2008, p. 206]. Often, the reaction is in form of an adaptation—either in the digital or physical world. Context-aware systems therefore can be seen as a form of adaptive interfaces that mediate "interaction with the real world" [Jameson 2006, p. 110].
- *Sensing, Actuation* and *Inference*: Sensors, actuators and inference mechanisms are the underlying building blocks of context-aware adaptation [Gross *et al.* 2006]. As those concepts stand for a more technical perspective on ubiquitous and mobile computing, they are discussed here as separate aspects, to provide an explicit counterpoint to the more human-centred notion of context-awareness. Sensors and actuators are hard- and software components that either observe phenomena (sensor) or produce an effect (actuator) in the physical or digital world [Schirmer & Gross 2011]. Inference mechanisms (also inference engines (IE)) are the links between sensors and actuators.

They allow drawing conclusions from the sensor input by deducing higher-order information from the raw sensor data [Schirmer & Gross 2011]. This ranges from simple mathematical calculations and monitoring thresholds, over rule-based inference from multiple sensors to pattern-recognition and prediction tasks based on machine learning. Together they build the backbone of a context-aware system that monitors (i.e., *sensing*) the environment to determine (i.e., *inference*) the current context and cause (i.e., *actuation*) an appropriate reaction. In order to build an abstraction-layer between the low-level input of sensors and the high-level concept of context, different sensor-based toolkits, environments and infrastructures have been developed (e.g., *Context Toolkit* [Dey *et al.* 2001] or *Sens-ation* [Gross *et al.* 2006]).

- *Nomadicity* and *Nomadic Users:* The notion of *digital nomads* [Makimoto & Manners 1997] surfaced in the mid 1990ies, at a time when technological development allowed to foresee that in near future technology would enable us to work and live more and more location-independent. Researchers from the field of mobile computing were among the firsts to pick up on this new notion of *nomadic users* that would define new needs and requirements towards computing technology. The research field of *nomadic computing* and the idea of *nomadicity* [Kleinrock 1996] originated in this time—the later defined as "the system support needed to provide computing and communication capabilities and services to nomads as they move from place to place in away that is transparent, integrated, convenient, and adaptive." [Kleinrock 2001, p. 42]. Thereby, a twenty-four-seven and ubiquitous availability of networks access as we have now was out of scope at first—from a technical perspective *nomadic computing* started with the idea of semi-mobile computing, where at "a minimum, the client mobile device is transportable to secondary fixed locations with no connection while in transit" [Ingelbrecht *et al.* 2009, p. 37]. Accordingly, the paradigm of nomadic computing is not necessarily technology-centred, but a user-centred view on the Post-PC era.
- *Calmness* and *Social Embeddedness*: Weiser's [1994] idea of UbiComp was to move technology into the background and, by weaving it into the fabric of our daily lives, to make it invisible. Weiser's and Brown's idea of *calm technology* [Rogers 2006] was to move the machinery to the periphery of our attention, when not needed—and to the centre of our attention, when it is needed. They introduce *Calmness* as "a new challenge UbiComp brings to computing" [Weiser & Brown 1995]. In the same vein, many UbiComp researchers also point out how important it is to embed technology in social processes, in order to

make it invisible [Abowd & Mynatt 2000; Dourish & Bell 2007; Fetter & Gross 2007]. They argue that computational technologies—and especially *UbiComp* technologies—are "embedded in social structures and cultural scripts of many sorts" [Dourish & Bell 2007, p. 141], and thus a deeper understanding of those social and cultural practises is needed, when designing UbiComp systems. Accordingly, one recent trend in *UbiComp* is to make systems—within the bounds of possibility—*socially aware* [Lukowicz *et al.* 2012].

- *Privacy, Security, and Trust:* As *UbiComp* and *Mobile Computing* in great parts are founded on collecting data upon users to provide adequate system support, the topics *privacy*, *security*, and *trust* are of central and uttermost importance in these fields. Dourish remarks how *UbiComp* "with its emphasis on capturing aspects of personal and collective 'context'" [Dourish & Anderson 2006] becomes a field were privacy issues become particular visible. Issues of *privacy*, *security*, and *trust*, were a concern of the earliest *UbiComp* systems (e.g., *Active Badge* [Want *et al.* 1992]), persisted ever since, and therefore became a leading research theme (compare e.g., [Beckwirth 2003; Bellotti & Sellen 1993; Boyle & Greenberg 2005; Camp & Connelly 2007; Hong & Landay 2004; Langheinrich 2001; Palen & Dourish 2003; Spiekermann 2007]).

To conclude, we now link these five specific aspects of *UbiComp* and *Mobile Computing* to this thesis. First, the central contribution of the work presented here, is a concept for the automatic adaptation of the availability of users based on their current situation—Hence, a system that puts the ideas behind *context-aware adaptation* into practise. Second, from a technical perspective a framework of *sensors, actuators and inference engines* (*IEs*) is proposed, as well as their reference implementation, to drive the automatic adaptation. Third, the adaptation of availability states is demonstrated for *nomadic users* relying on semi-mobile computing technology (i.e., laptop computers). Fourth, for the design of the concepts and technology in this work the ideas of *calm technology* as well as a sensible understanding of their *social embeddedness* were programmatic. And finally, in this sense a sorrow understanding of the different conceptions of *privacy* is an elementary building block of this work—as was explicated in detail in the previous sections.

2.4 An Overview of the Related Work—Systems for Presence and Availability Management

In section 2.1.3 we briefly heard about the different types of systems that aim to support the management of presence and availability. In the next section, I present a selection of such systems, to highlight the different approaches that

have been conceived in HCI, CSCW, and UbiComp research, to address this problem. The aim is to provide an overview of the related work that is relevant for assessing the contribution made in this thesis. By presenting the related work in context, I hope to also provide a general understanding for the research directions in this field.

2.4.1 Explicit Availability Management through Manual Adaptation

Earlier we learned to distinguish between *explicit* and *implicit availability management*. Systems for *explicit availability management* focus on the manual adaptation of some status or profile, indicating the current availability to others or filtering incoming messages, calls, or other interruptions. *Implicit availability management* can be divided into systems that are based on sharing *awareness information* and such that are based on the *automatic calculation* or prediction of availability. That is, systems that rely on human interpretation versus systems that rely on computed interpretation of awareness information. In the Following I provide a brief introduction in systems for *explicit availability management*. To do so, it is important to recall that *availability management* is concerned with signalling the own willingness to communicate or be interrupted to other persons [Harr & Wiberg 2008]. Setting the ringer mode on a mobile telephone to silent would therefore not be an example for this approach, as it only has an effect on the own interruption. The same is true for current solutions to reduce interruptions on a system level, as for example the "Do Not Disturb" feature introduced in Mac OS X 10.8. When this feature is manually activated, it will stop displaying any notification, including those on incoming communication. However, possible communication partners are unaware of this.

Accordingly, the most classic examples for the *explicit* approach are the previous mentioned IM clients like ICO [ICQ LLC 2013] or Skype [Skype Limited 2011], where a buddy list allows us assessing our contacts' availability, manually set by each contact. With the idea to bring this basic concept also to landline phones, Milewski and Smith [2000] developed and evaluated the *live addressbook* prototype developed for PDAs (and desktop browsers). The network-based address book was able to store and make accessible presence data in form of an availability state (possible states are: *Available, Urgent Only, Away (leave message)*, and *Do Not Disturb*), a short status message, and a landline phone number as location. The availability state, status message, and location could be manually set via PDA or desktop PC by the user. Via a functionality called "click-to-dial", the PDA allowed to connect to people via telephone. *Calls.calm* [Pedersen 2001], also a system for managing the availability for telephone calls, allowed users to manually specify their availability and presence in three dimensions: role (with the values: *at work, offwork, in between*), location (with the values: *at home, at work, nearby, away*),

and social setting (with the values: *alone, in meeting*). The information could be accessed as a WML or HTML page by a WAP-enabled mobile phone. The application also aimed at enabling communication between callers, to negotiate availability.

OwnTime [Rodenstein *et al.* 1999] represents a very different approach for *explicit availability management*. *OwnTime* supports the proactive announcement of the own availability for a meeting with a specific person. The software allows specifying one's short-term availability in minutes for a meeting with another *OwnTime* user. Hence, the announcement of the availability goes hand in hand with a concrete request for a meeting. The application was developed for translucent head-worn displays (i.e., a pair of smartglasses) and evaluated in a study.

2.4.2 Implicit Availability Management in Early Systems for Awareness

In the area of *implicit availability management*, the first type of systems to provide awareness on *presence* and *availability,* were *media spaces* and *event-based notification systems*.

Media spaces are in their most basic form a permanent audio and video connection between two different locations "functioning like an extension of physical space" [Bly *et al.* 1993, p. 34]. The first media space was created at Xerox PARC in the mid1980s. It started as a single permanent audio and video connection between two local office rooms and grew into a network of continuous, always-on connections between several offices and public spaces in two geographical distributed Xerox laboratories [Bly *et al.* 1993; Harrison 2009b]. Soon, other systems followed, as for example *Polyscope* and *Vrooms* [Borning & Travers 1991], *RAVE* [Gaver *et al.* 1992] and *Portholes* [Dourish & Bly 1992]. From a technological perspective, the first systems used analogue audio and video equipment like television sets, VCRs, and video cameras. While some media space implementations started to build on digital technology [Harrison 2009a], all systems basically allow users to glimpse anytime into other offices or public spaces. Also common for most early media spaces is the fact that they were used by the researchers inhabiting the laboratories, in order to experience themself how social practice unfold in mediated ways[1]. Among their topics of interests was privacy as well as awareness. Hence, aspects closely related to the concept of availability.

However, the first mediaspaces have been seen as massive privacy intrusions, where the benefits did not outweigh the costs [Boyle & Greenberg 2005; Boyle *et al.* 2009; Gaver *et al.* 1992]. Pedersen and Sokoler [1997] distinguish between

1 Essentially these are early forms of the *living lab* approach.

direct representations and *abstract representations* of awareness information. While mediaspaces belong to the group of direct representations, some efforts are made to develop systems that degrade or remap the original signal to allow for more privacy and more peripheral consumption of the awareness information (e.g., *shadow-views* by [Hudson & Smith 1996] and *AROMA* by [Pedersen & Sokoler 1997]). Other approaches to reduce privacy intrusion through medias spaces are the integration of intentional awareness mechanisms e.g., by supporting glancing features like in *Montage* [Tang *et al.* 1994; Tang & Rua 1994] or RAVE [Gaver *et al.* 1992]. *Glancing* allows to shortly peek into another person's office for a few seconds—sometimes glancing is even reciprocal. This moment of focussed attention allows assessing the presence and availability of a colleague.

The *Montage* [Tang *et al.* 1994; Tang & Rua 1994] prototype aimed to support— what the author called—*teleproximity*. The aim was to replicate the auditory and visual cues that co-located teams share to establish and negotiate opportunities for interaction and communication. The interaction design of *Montage* is based on the 'hallway model'—the idea that in physical environment a person can peek through open office doors in a hallway, to get an understanding of the occupant's availability and eventually establish contact. Accordingly, the most prominent feature is the *glance* mechanism, which establishes a short, reciprocal video connection between two networked workstations (cf. Figure 9). This glance can either be extended into a full videoconference, or—if the other person is not available—move to other means of communication (e.g., leaving a message, peeking at the online calendar, or writing a email). Nonetheless, in the *Montage* system we already can see, how different types of availability management are sometimes combined. *Montage* for example allowed setting a 'do not disturb' mode, hence integrating approaches for *explicit availability management* in an environment for *implicit availability management*.

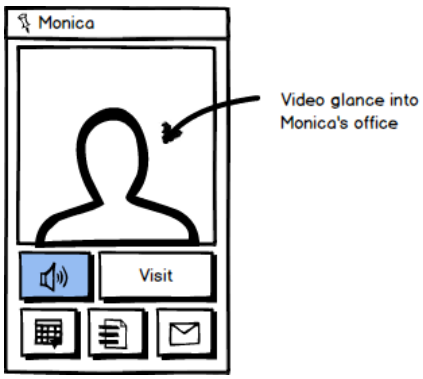

Figure 9. Illustration of the *Montage* prototype (based on a screenshot in [Tang & Rua 1994]).

Another research stream investigated ”computational environments based on ‘event propagation mechanisms’” [Schmidt 2002, p. 268] that aim to support awareness through the collection and dissemination of bits of information on cooperative activities, like *Khronika* [Loevstrand 1991], *GroupDesk* [Fuchs *et al.* 1995], *Elvin* [Fitzpatrick *et al.* 2002; Fitzpatrick *et al.* 1999], or *NESSIE* [Prinz 1999]. The so-called *event-based awareness* is “concerned with providing people with awareness of what is going on around them, as expressed by discrete events” [Rittenbruch & McEwan 2009, p. 11].

Khronika [Loevstrand 1991] was one of the earliest implementations of such systems. In principle it is an interface to a database of events that allows retrieving events mainly in form of subscription-based notifications. Notifications can be presented in different forms, as for example through popup windows, by automatically sending an email, or by the provision of an auditory signal in form of sound effects and synthesised speech. The events are received from a number of different sources (external sensors, a location tracking system, calendar information, etc.), and personal daemons dispatch information about events of interest to the recipients.

Elvin [Fitzpatrick *et al.* 1999] is a general purpose, publish and subscribe notification service that was utilised for awareness support in work settings. This utilisation was mainly triggered by the application *Tickertape* [Fitzpatrick *et al.* 1998] that easily supported the consumption of awareness information in form of automatically scrolling messages in a single line window, often placed at the top of the display. *Tickertape* allowed subscribing to messages based on groups or the content of the messages. New messages could easily be sent via an interface that allowed specifying the message text, the recipient group and a timeout for the message. Further, users wrote their own scripts to receive notifications for example when they received an email or files on a network drive were changed.

Also *NESSIE* [Prinz 1999] and *ENI* (Event and Notification Infrastructure) [Gross & Prinz 2003; Prinz & Gross 2004] are general infrastructures for distributing awareness information. While *NESSIE* relatively early introduced *indicators* (i.e., *actuators*) that allow an ambient presentation of awareness information, *ENI* specifically focussed on how event notifications could be contextualised to better fit to a user’s situation.

2.4.3 Combining Availability Management and Communication

Followed by those early systems, more and more research prototypes combined different approaches for availability management in one client and often integrated means for establishing communication. That is, many clients offered a manual explicit management but enriched it with additional implicitly

collected awareness information by sensors. Often such tools would also allow initiating communication directly via various channels, often including IM.

As a result of the privacy issues that were encountered with some early systems like *media spaces*, the mechanisms to specify the recipients of the information disclosure became more and more sophisticated, and hence also better aligned with users' privacy needs. That is, some systems started to take into account, that users have different availability preferences towards different people as well as preferences on what information they want to disclose with whom. One important research strand here is, the introduction of new mechanisms for the specification of recipients and information disclosure at different granularity levels: From one global setting for all, over the specification on the level of individual recipients, towards assigning preferences on the level of groups of recipients or roles. Along with these considerations, the degree of asymmetry or reciprocity of the information sharing became also of interest.

Much alike the previously discussed awareness systems, also *Grapevine* [Richards & Christensen 2004] is designed as an infrastructure. It allows aggregating and publishing context information. In order to obtain awareness information about a person through *Grapevine*, so-called *e-Cards* enable people to access information about a person's current location and activity as well as a list of the communications channels (e.g., instant messaging, email, telephone...) through which the person is currently available. The *e-Cards* thereby have the form of small, digital business cards, displayed as an active window, which can be placed on a computers desktop. Further, the access to *e-Cards* can be regulated on the basis of either the relationship of discloser and recipient, the membership in a specific group or on basis of the current situation. The discloser of the awareness information can specify those permission rules via a settings view (cf. Figure 10). The permissions in *Grapevine* are not symmetrical so the information disclosure does not need to be mutual or reciprocal.

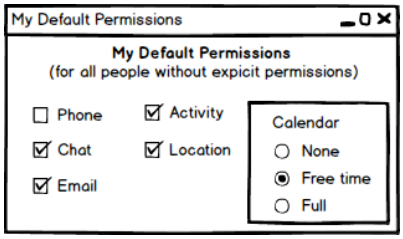

Figure 10. Grapevine's permissions settings for default viewers (adapted from a screenshot in [Richards & Christensen 2004]).

Also *ConNexus* [Tang *et al.* 2001]—developed at Sun Microsystems Laboratories—is a desktop client that combines awareness and communication functionalities. Basically *ConNexus* consist of a contact list—comparable to that

of an IM client—where each contact is augmented with some availability and activity information (i.e., if the user is idle or the user is communicating), a contact toolbar showing the currently best options for reaching a contact—comparable to the information on the *Grapevine e-Cards*—and a set of communication tools, among which the most prominent was an IM chat. Contrary to *Grapevine*, all awareness information is shared *reciprocal* and subscriptions are handled by adding persons to the contact list. Most of the functionality of *ConNexus* was brought to mobile devices (Palm PDAs and RIM Blackberry devices) with the application *Awarenex* [Tang *et al.* 2001]. *Awarenex* also offers a contact list that shows brief information on location, activities and communication of the person's contact. More details are available on demand by selecting a contact and viewing the *Contact Locator*. The *Contact Locator* provides a sense of availability by showing idle times of a contact's various communication channels (e.g., home phone or mobile IM) and means to initiate communication.

The design of *InterruptMe* [Hincapié-Ramos *et al.* 2011] is informed by the upfront analysis of the design space of availability-sharing systems. The systems allows the users to specify their availability sharing preferences for four types of communication channels (face-to-face, telephone, IM and email) according to five categories of people (identified by [Olson *et al.* 2005]): spouse, family, boss and trusted colleagues, other colleagues, and public. The users can follow their contacts by adding them to the different categories, while the non-reciprocal nature of the system does not make it necessary for the contact to approve the availability information request. *InterruptMe* relies on software acquiring electronic activity information and the Phidgets InterfaceKit in combination with different sensors for acquiring physical activity information. *InterruptMe* was not yet evaluated in a user study.

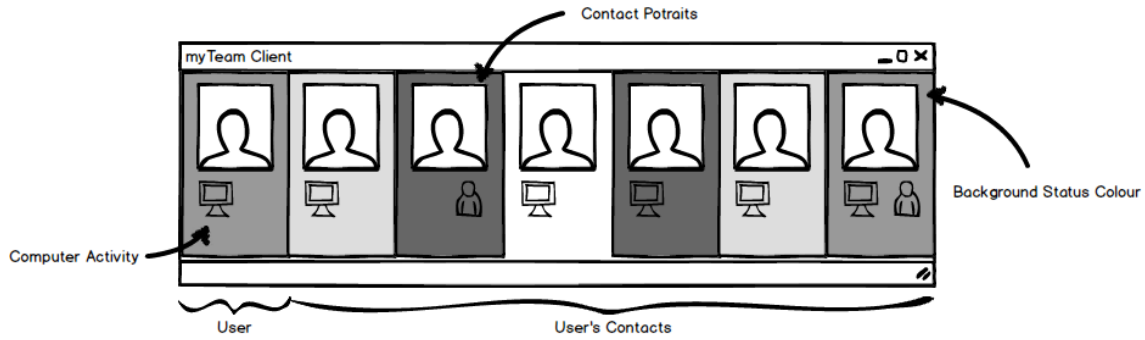

Figure 11. TileSet View of myTeam Client (based on a screenshot in [Lai *et al.* 2003]).

The *myTeam* system [Lai *et al.* 2003] provides awareness on the availability of co-workers in distributed work groups by utilising sensor data in combination with user-selectable states like "do not disturb". The *myTeam* client (cf. Figure 11) shows the pictures of a user's contacts enriched with additional awareness

information in horizontally aligned tiles. The colour-coded background of each tile provides awareness of the respective contact's presence and availability, which is whether the contact is in the office and available, at another known location, at a unknown location, or does not want to be disturbed. Finally, a *hidden* state allows for the *only* asymmetry in the sharing of availability information, by temporally deactivating the distribution of sensor information to all recipients. The information is sensed from different sources, including the Active Badge system [Want *et al.* 1992] for location information, mouse and keyboard activity and network information in form of the IP address. Further, the user can manually enter a status message. If a contact is currently unavailable, there is a possibility to register for a notification, which is send when the contact becomes available again. With only the *hidden state* as a privacy function, the options for attaining adequate privacy are limited.

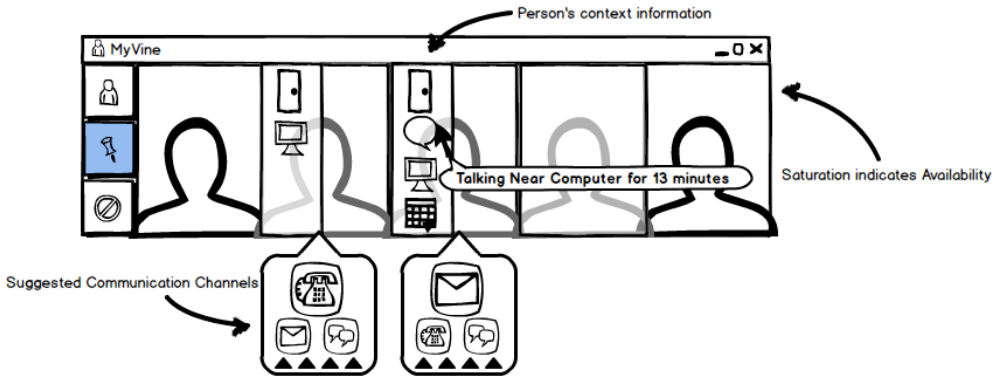

Figure 12. Illustration of the *MyVine* client (based on a screenshot in [Fogarty *et al.* 2004b]).

MyVine [Fogarty *et al.* 2004b] was designed to support small groups of up to ten people with mutual awareness about each other's availability and presence. The client relies on the concept of *progressive disclosure*, as it provides availability in three levels of details, based on the user interaction with the client. First, the saturation of a contacts image allows a first coarse assessment between highly available, moderate available and highly unavailable. In a second level, four icons (cf. Figure 12) indicate if a person is in their office, currently speaking, using their computer, or are currently busy according to their calendar—all information is assessed via sensors. A third level of detail is provided when the mouse hovers one of these icons. In this case more detailed information in form of a tooltip is provided. Further, based on these four availability indicators, an option for the best contact channel is proposed by providing buttons that allow initiating a chat, an email or a phone call. *MyVine* does not allow for selective disclosure of information and asymmetric sharing.

The FXPAL's *myUnity* [Biehl *et al.* 2011; Biehl *et al.* 2013; Wiese *et al.* 2011] also provides an overview of the colleagues' presence, and like *MyVine* also without allowing for selective sharing. However, by permitting the user to

switch off individual sensors, it at least allows for asymmetry on the global level. The GUI is divided in an array of coloured tiles. Each tile in *myUnity* 's awareness dashboard represents a person in the organisation (cf. Figure 13). The system infers one of five possible presence states from a variety of sources, including office cameras, WiFi and Bluetooth networks, mouse and keyboard activity, network connectivity, IM availability, calendar information, etc. Each source can be deactivated, allowing for some asymmetry in sharing. Besides a Microsoft Windows based desktop client, also a mobile variant for Android phone exists. The presence states range from "physically in office, alone" to "not in office building".

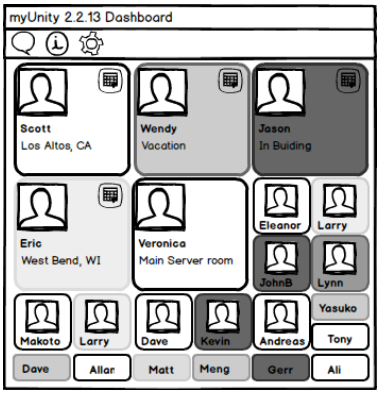

Figure 13. Illustration of the *myUnity* client (based on a screenshot in [Wiese *et al.* 2011]).

Other approaches use a manual specified rule, in combination with sensor inputs. *PRIMInality* [Gross & Oemig 2005b] combines the readings from two sensors measuring the movement at the user's desk and in other parts of the user's office with the current usage of the *PRIMInality* client (i.e., whether user is logged-in and a chat window is open). Based on these inputs a rule-set is used to adapt the availability state, the status message, and the applications behaviour.

With *IMBuddy* [Hsieh *et al.* 2007] availability management it realised in a very different form. An AOL Instant Messaging bot allows the users to ask questions about a contact's interruptibility, location, and current window in focus. For example the user could send a message to the bot asking "*howbusyis* userX" and the bot would answer something like "userX is 60% available 10 mins ago". An *IMBuddy* client on the contact's computer would use various sources (including Subtle [Fogarty & Hudson 2007]) to sense this information in the background. Further the client allows configuring to whom and on what detail level information is disclosed.

PRIMIFaces [Gross & Oemig 2006] aims to overcome the limitations of most current IM systems that only allow maintaining one single online state and one single status information. Gross and Oemig introduce the *Faces* concept to facilitate manageable selective information disclosure in cooperative

environments. Based on the works of sociologist Erving Goffman [1959] they introduce the *Faces* construct as a digital representation of a specific front of a user that comprises the information sources the user wants to disclose to a specific audience as well as the audience in form of a number of contacts to which the information is disclosed. In order to configure their specific information disclosure needs users can create and maintain several *Faces* in parallel and freely add and remove IM contacts and information sources in form of different sensors. Thereby each contact and each information source can be on more than one *Face* at the same time. The concept of *Muting* allows a user to temporarily interrupt the information flow of one *Face*.

2.4.4 Availability Management Approaches in Ubiquitous and Mobile Computing

The solutions presented so far mostly used some sort of graphical user interfaces on a stationary desktop computer to display and control awareness information. With increasing influence of the *UbiComp* research stream, some scholars also started exploring how presence and availability management could be moved away from the screen and keyboard. In the Following I present related work that can be categorised into two distinct research directions:

- In the field of mobile and pervasive computing, researchers developed systems that aim to support nomadic and mobile users, moving between different locations and contexts, with managing there availability.
- In the area of ambient and tangible interfaces, researchers implemented novel prototypes that move availability management away from screen and keyboard into the rooms and offices. Following the notion of invisible and disappearing computing, the prototypes display information in the environment of the user and also allow manipulating it in the periphery.

Again, the categorisation of the related work into strictly distinctive categories is not feasible. Many systems combine several relevant criteria, as has been shown in the analysis of the design space of availability sharing systems by Hincapié-Ramos *et al.* [2011]. This said, of course systems discussed earlier, that rely on some sort of physical sensor as input, can be also seen as approaches utilising concepts of UbiComp. For example, the *myTeam* system [Lai *et al.* 2003] uses *Active Badge* [Want *et al.* 1992] for location information and *myUnity* senses for nearby WiFi and Bluetooth networks.

2.4.4.1 Availability Management for Mobile and Nomadic Users

One common theme for mobile applications is the augmentation of the address book of a phone with additional information. The already mentioned *Awarenex*

[Tang *et al.* 2001] and *live addressbook* [Mileswki & Smith 2000] are examples of this approach. And also *ContextContacts* [Oulasvirta *et al.* 2005; Raento & Oulasvirta 2008] is an address book application for mobile phones that augments the contact entries with additional information—called *situation cues*. *ContextContacts* aims to replace the standard Contacts application on Symbian S60 phones. The *situation cues* are presented next and below each contact (cf. Figure 14) and include information on: the contact's current location and time spent in that location; the user-set ringer and vibration mode; information if the phone is currently used; and nearby BT devices of know and unknown persons. *ContextContacts* provides groups for specifying information disclosure preferences towards different audiences. Further, a *lookup log* allows the user to monitor who, when accessed information about them.

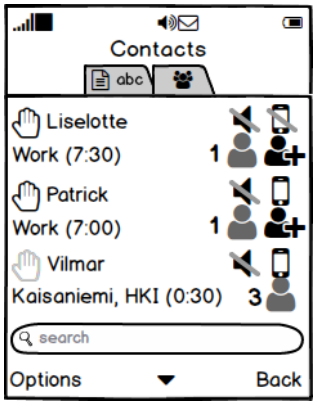

Figure 14. Illustration of *ContextContacts* (based on a screenshot in [Oulasvirta *et al.* 2005]).

Connecto [Barkhuus *et al.* 2008] is a status and location sharing application for mobile phones. Like most applications for mobile phones presented here, it enriches a contact list with information. Here it is the location of a contact, how long the contact stayed at the location, and the phone's current ringing profile. *Connecto* is especially designed to keep small, close-knit groups of friends connected. In a study they found that the tool not only was used as an awareness tool to check the availability and presence of others, but also was used to tell stories and establish a feeling of connectedness. By allowing manually overwriting the sensed location, it put the users in control by supporting vagueness and ambiguity as privacy mechanisms [Aoki & Woodruff 2005; Hancock *et al.* 2009; Reynolds *et al.* 2013].

The *AWARE* architecture [Bardram & Hansen 2004] supports social awareness among clinicians in a hospital. In a participatory design approach Bardram & Hansen developed *AwarePhone,* a mobile application for Symbian S60 phones. The application provides an overview of the colleagues in form of a buddy list, with the following information for each contact: a user-entered personal status, the current activity retrieved from the calendar, and the location

on a room-level granularity from a location system using IR, WiFi and Bluetooth beacons. There are no mechanisms in place for controlling the disclosure in this work related setting.

Hubub [Isaacs *et al.* 2002a] is a mobile IM application, aimed at nomadic users, providing awareness information on the user's device usage activity (i.e., tapping, clicking, or typing) displayed in form of an activity meter next to each contact in a contact list. The design goal of *Hubub* is to encourage opportunistic conversations between geographically distributed co-workers allowing them to stay connected. It extensively uses sounds on wireless Palm devices and in its accompanying PC client. Each user is represented in *Hubub* by a *Sound ID* in form of an *earcon*. The *Sound ID* is played when a specific user is becoming active.

Finally, the *Negotiator* [Wiberg & Whittaker 2005], departs from most approaches we have seen here, as it does not provide awareness information upfront in a contact list. The implementation for a PocketPC smartphone simply provides an interface to negotiate a better time for calling. When the phone rings, instead of answering, the user can use a form to send a message to the caller, with a suggestion for a time the user will have a better availability.

Some more examples for mobile systems are provided further below, in the section on systems relying on *automated calculation*.

2.4.4.2 Availability Management in the Users' Environment

As mentioned before, another part of the research community is interested in moving presence and awareness information away from our computers and mobile devices. They suggest *ambient* [Pousman & Stasko 2006] output or *tangible* [Ishii & Ullmer 1997] input modalities for presence sharing and availability management. Ambient output modalities, as for example ambient telepresence, present information on sensed background activities in target representations that are "not monopolizing attention and not competing for display resources" [Gellersen & Beigl 1999, p. 81]. Tangible input modalities allow for a more natural interaction, when users are manually adapting their online state. Early prototypes based on these ideas are for example the *MediaCup* [Gellersen *et al.* 1999], Jeremijenko's *Dangling String* [Weiser & Brown 1995], Gaver's *EAR system* [Gaver 1991], Ishii's *ambientROOM* [Ishii *et al.* 1998] or *NESSIE* by Prinz [1999]. *Labradoor* [Nichols *et al.* 2002] and *Bluespace* [Lai *et al.* 2002] are both systems that move the presentation of availability information off the computer screen and into the room, by projecting it on doors and walls.

With *FlowLight* [Züger *et al.* 2017] a system was developed and evaluated that aims to reduce in-person interruptions in an open office space. *FlowLight* is a physical LED, placed on the users' desks, that uses a traffic-light metaphor to

signal interruptibility to people approaching. *FlowLight* automatically calculates an interruptibility state based on a detailed analysis of the users' mouse and keyboard interaction and calendar events. In a long-term field study with 449 participants they found that the system was able to raise the awareness of the potential disruptiveness when approaching others. *FlowLight* also was able to reduce the participants' interruptions by 46%.

Figure 15. Illustrations of *StatTube* (based on [Hausen *et al.* 2012]) on the left and *Hangsters* (based on [Peek *et al.* 2003]) on the right.

StaTube [Hausen *et al.* 2012] is also and ambient display and tangible input device in form of an illuminated multi-layered tube (cf. Figure 15 left). Each layer represents a contact in a user's IM client, while the topmost layer represents the user's own state. Each layer can be lid up with RGB LEDs, while different colours represent different online states: green stands for *Online*, yellow for *Away* and red for *Do Not Disturb*. By turning the topmost layer the user can change the own state, and by pressing it activate a timer, that signals the estimated duration of this state.

Hangsters [Peek *et al.* 2003] are one more example for tangible, ambient displays of IM availability states. A hangster is a small, motorised box (cf. Figure 15 right) that is hanging on a string (e.g., attached to the ceiling) and communicates via Bluetooth with an IM client on a computer. Each hangster represents one IM contact of the user, and can be customised with a printed exterior, to be discriminable. The position (up means offline, and down means available) and the movement (moving up and down quickly means incoming notification) of a hangster provide awareness about the respective contact. Pulling a hangster down allows tangibly accepting an incoming notification or starting communication. *ComSlipper* [Chen *et al.* 2006] is one of few approaches trying to mediate presence awareness in a more intimate, emotional, and calm way to support *social presence* [Short *et al.* 1976] and *human connectedness* [Agamolis 2005]. Two pair of slippers augmented with pressure sensors as input and actuators in form of LEDs (light), a vibration motor (vibration), and a heater

resistor (warmth), are connected via a network and allow sharing presence information through gestural and tactile interaction.

2.4.5 Availability Management Approaches in Automatic Calculation

As said before, the idea to *automatically calculate* or predict the availability, presence, or interruptibility of a user is a further research stream in the area of availability management. In 2004 a first workshop on forecasting presence and availability was held [Tullio *et al.* 2004] to discuss technological and usability challenges. In my analysis of related work, I identified basically three levels of contributions in this field:

- First, there is work that looks at the predictability of availability (also interruptibility, presence, etc.) per se in different settings. Usually, the data is collected in a study—using multifarious combinations of sensors, fitting to the setting. For collecting the ground truth in form of labelled data often the Experience Sampling Method (ESM) is used—which is explained in detail in section 3.2. Once the data is collected, it is analysed in an explorative machine learning approach. That is, to see what machine learning performance can be achieved for predicting the respective construct, by training models with different classifiers and evaluating the results.
- Second, some researchers went a step further, and deployed the pre-trained models in an application, that automatically manages the availability (e.g., by adapting the online state of an IM client).
- Third, only few researchers so far have built systems that learn individual models ad-hoc from the user. Such systems would obtain new labels by either querying a user or observing their interactions, to continually train a model that is used for live adaptations.

The complexity of the settings, used sensor data, and accordingly learned models usually decreases with the level of contribution, whereby research at the first level usually addresses settings with a higher complexity.

Microsoft Research was among the first commencing research in this direction in their projects *Lumière* [Horvitz *et al.* 1998] and *Attentional User Interface* [Horvitz *et al.* 2003] where they among other things developed *Coordinate* & *Priorities* [Horvitz *et al.* 1999; Horvitz *et al.* 2002]. The *Priorities* system is able to learn the urgencies of email messages (i.e., "the expected cost of delayed review" [Horvitz *et al.* 2002, p. 225]) by training a Support Vector Machine (SVM) with features extracted from the header and body of the email and labels derived from the users combined with some heuristics. *Priorities* further learns about the users' *presence*. By monitoring desktop computer activity and building cumulative probability distributions for specific periods of time, it is able to provide the probability for queries like: Will the contact return to the office in

the next 15 minutes? Based on the prediction of *presence* and *urgency*, *Priorities* among other things is able to forward urgent messages to the users mobile phone when the user is not likely to return to the office soon. While some information on the prediction performances for classifying the mails is provided, no global statement on the quality of the presence forecasts is given in the paper, which would be of interest in this work. *Coordinate*—based on *Priorities*—is an automated service for predicting presence and availability and is trained from observational data combined with user input. *Coordinate* for example collects data from all meetings stored in the calendar of a user. Using Bayesian networks, they analysed how predictable the likelihood of attending a meeting for a certain user is. The researchers also asked a user to annotate 659 appointments from the last 6 months, if the interruptibility during these meetings was low, medium, and high. They spared one eighth of the data for testing, used the rest for training and achieved an accuracy of 81% for predicting the interruptibility for this single user.

Fogarty et al. [2004a] collected interruptibility data from 10 participants in form of 975 oral self-reports measuring interruptibility on a five-point Likert-scale in an office environment. As sensor input they logged data on computer activity (mouse and keyboard activity, applications used, etc.) and room activity (door sensor, telephone usage sensor, motion sensor, etc.). Using a naïve Bayes classifier they trained four models: one general model, and three models for the different types of participants (i.e., manager, researcher, and intern). Overall the models build for each type (with accuracies from 80.1% to 87.7%) outperform the general model (with an accuracy of 79.5%). However, they achieve these accuracies only by reducing the complexity of the prediction from predicting five classes to a two-class problem (1-4 and 5). Further, they had a relatively controlled setting of one office room, which also reduces the complexity.

Sarker *et al.* [2014] analysed the availability of mobile users to attend to notifications from a mobile health monitoring system. They collected from 30 participants in a weeklong Experience Sampling Study 2,064 hours of data from physiological and mobile phone sensors (ECG, respiration, accelerometry, and GPS location data) as well as 2,717 ESM self-reports (42 items, including questions on the current activity or social interaction). They used the data from the sensors and the self-reports, to craft 99 features, which were reduced to 30 features in a feature selection process. They took the delay in responding to an ESM as the basis for labelling the data either as unavailable or available, according to a defined threshold. They achieve an accuracy of 74.7% with an SVM in a 10-fold cross-validation for a global model over all users. It needs to be noted, that the good accuracy for a global model is likely based on using mainly the ESM data as features. For example, using semantic location data from the ESM like “Home” or “Work” made location a valuable feature for a global

model, whereby using real sensor data in form of the GPS location only could have been used for building individual models per user.

The *SocioXensor* [Ter Hofte 2007] for PDAs and mobile phones was developed to collect data about human behaviour and social context. Ter Hofte used ESM to collect 785 availability estimates for phone calls together with other items regarding the current context, as for example the conversation status, location, and nearby people. Based on the data of 10 participants collected over 7 days in a nomadic setting, he is able to train a naïve Bayes classifier using the answers on the context data as features (e.g., features like `InConversation=face to face` or `LocationCoarse=at home`). With a 10-fold cross-validation he showed that an accuracy of 63.9% is achievable for classifying participants as either *Available* or *Not Available*. The work shares similar limitations like the work discussed before

The user feedback collected for *ConNexus* and *Awarenex* [Tang et al. 2001] inspired the *Lilsys* system [Begole *et al.* 2004] that was also developed at Sun Microsystems Laboratories. The *Lilsys* sensor and data acquisition module extended the keyboard and mouse sensors from *Awarenex* with additional physical sensors (cf. Figure 16) for detecting the door state and phone usage with binary switches and motion and speech detection with specialised sensors. The *Lilsys* system uses the filtered data from these sensors in combination with a decision tree to estimate the unavailability of a user. The two inferred states *possibly* and *probably unavailable*, and the *neutral* state (i.e., "no inference"), are presented to the user through the *Awarenex* contact list. The system is discussed based on user feedback and observations from several months of usage of the *Lilsys* system by four users. There are no details given on the quality of the inference mechanism based on the decision tree.

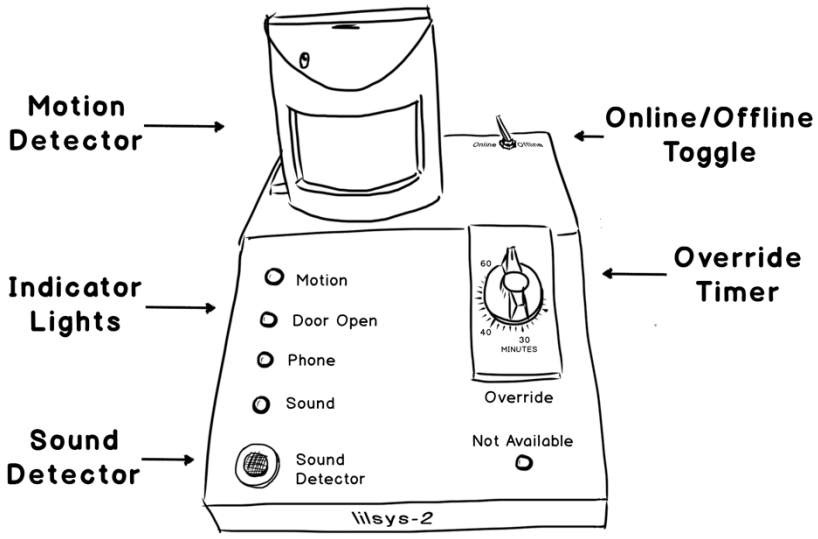

Figure 16. Sketch of the Lilsys device (adpated from a photo in [Begole *et al.* 2004]).

The previously mentioned *FlowLight* [Züger *et al.* 2017] not only adapts the LED, but also the users' Skype status based on its interruptibility calculation mainly computed from mouse and keyboard interaction of the users. Hence it is an example for systems that use pre-trained models.

The *BusyBody* system [Horvitz *et al.* 2004] uses an interval-based sampling on a computer desktop, to obtain labels ("Busy" and "Not Busy") for training a Bayesian network online. The prediction is based on the collection of low- and high-level events of computer activity and some contextual variables (i.e., calendar entries, WiFi location system, voice activity detection, time and date). Between 789 and 2,365 interruptibility samplings were answered from four participants over the course of several weeks. The achieved accuracy in average is 78.25% for this two-class problem with an online learning system.

Subtle [Fogarty & Hudson 2007] is a toolkit that able to automatically learn statistical models of human situations. It comes with an extensible sensor library, however only provides a limited number of sensors out of the box. While the system generally can be deployed as an adaptive system, which is able to continually learn a concept from sensor data by querying the user for labels, the incremental learning is only achieved through a form of iterative batch learning. While *Subtle* is able to perform automated feature construction, it needs to rely on batch learning for reducing the dimensionality of features with offline feature selection mechanisms.

Xu *et al.* [2012] collected 1,445 ESM responses from 23 office workers over a two-week period to analyse the predictability of the affective state and communication preferences of *myUnity* users [Biehl *et al.* 2011; Biehl *et al.* 2013; Wiese *et al.* 2011]. In a performance evaluation in form of a 10-fold cross validation they were able to predict a positive vs. negative manifestation on the arousal dimension with an accuracy of 76,5%, respectively with an accuracy of 76,4%, on the pleasure dimension. The used features are based on the presence states of *myUnity* and activity on the users' desktop computers. The idea is to provide contacts with a better estimate of the recipient's mood, to find an applicable moment for communication. For estimating the preferences for being contacted over one of the four different communication channels (i.e., email, phone, IM, face-to-face) a model for each channel was build. The accuracy is between 74% for face-to-face and 87% for IM.

With *Nomatic* [Patterson *et al.* 2009] an approach of predicting IM status messages for nomadic users with sensor data from laptop computers is presented. As IM users tend to reuse previous status messages in the same contexts, *Nomatic* provides an easy input mechanism in form of a GUI for composing IM status messages. Each message is concatenating previously used phrases from three categories: place, activity, and other (e.g., "in my office; writing; please don't interrupt"). When the user wants to change the status,

Nomatic presents a suggestion based on the current context. The users can either: accept the suggestion; select other phrases from a ranked list for every category; or type in a new phrase. The three phrases are stored together with a current sensor reading, to train a machine learning model for future suggestions. In an evaluation study they collected 7,154 IM status lines with corresponding sensor readings from 14 nomadic laptop users over the duration of 3 month. Based on the collected data, they did a within-subjects performance evaluation with five different classifiers. The best performance was obtained with a SVM, where they achieve an accuracy of 93% for predicting the phrase for activity, 94% for place, and 96% for other.

Ambush [Mynatt & Tullio 2001], a probabilistic calendar, provides forecasts of the probability of an individual attending a future calendar event. The system uses the ”attendance habits of calendar owners” [Mynatt & Tullio 2001, p. 122], in form of information about past events to build a Bayesian model that is able to predict the likelihood for a specific user attending a particular future event. The aim of *Ambush* is to provide information that support informal visits in the workplace. The paper provides no information about the accuracy the *Ambush* system can achieve. They also use the approach in their group calendaring system *Augur* [Tullio *et al.* 2002].

Some researchers looked into automatically adapting the volume of a mobile phone [Fisher & Simmons 2011; Kern & Schiele 2006; Khalil & Connelly 2005; Rosenthal *et al.* 2011] in order to reduce interruptions. Although these contributions are more concerned with interruptibility than with availability, it is nevertheless interesting to look at the respective approaches. Thereby especially the work by Rosenthal *et al.* [2011] shows some parallels to aspects of what is presented in this thesis. They logged different features (e.g., GPS longitude and latitude; x-,y-, and z-acceleration; noise-level; user is in meeting; user is on phone) on 20 students’ Android phones that are likely to help characterising the users situations. They further analysed incoming notifications or calls to extract features about the contacting person (e.g., if the person is among the users contacts or even favourites; how often the person had contacted the user before). This information allows characterising the importance or context of the alert. They used different sampling approaches for the ESM—including a decision-theoretic approach [Kapoor *et al.* 2007]—to learn user preferences regarding the muting state of the phone. They had a separate phase for collecting and training the system and for testing the trained models. Both phases lasted two weeks. Out of 19 participants 13 were satisfied with the accuracy of the trained model. The achieved accuracies were between 83% and 90%.

Pielot [2014] analysed the predictability of the availability of users for mobile phone calls in a large-scale study. The predictions are based on data collected

from roughly 418 users that were logging data on over 30 thousand incoming calls in an large-scale, in-the-wild study [Pielot *et al.* 2014; Sahami Shirazi *et al.* 2014]. They were able to predict the availability with an *Random Forest* classifier, achieving an accuracy of 83,2%, based on 15 different types of information, among which are: the charging state, day of the week, hour of the day, display proximity sensor, accelerometer and gyroscope states right before the call, ringer mode, number of calls from the same caller, etc. Thereby the information whether a call was picked up or not, was used as and indicator of the availability, and therefore as a label for training the classifier. By building individual models for each user, only based on their respective data, the result could be slightly improved to 87% accuracy.

Other explored the predictability of interruptibility based on observing biometric data. Züger and Fritz [2015] were able to predict the interruptibility of software developers during a programming task from a combination of different biometric sensors (i.e., electroencephalography (EEG), electrodermalactivity (EDA), skin temperature, and photoplethysmography (PPG)). They achieved an accuracy of 43.9% for predicting five states from 1 (highly interruptible) to 5 (not at all interruptible) based on data of a lab study. With data from a field study they achieved an accuracy of 32.5%.

Tani and Yamada [2013] used a pressure sensor as a desk pad that covered about one square metre of a user's desk and measures pressure with a resolution of 10 mm^2 to estimate users' interruptibility based on table-top pressure. They computed four features (e.g., pressure, centre of gravity, etc.) for six regions of the sensor (e.g., region around the mouse, the arms, etc.). In an experiment they collected sensor-data together with statements from 20 participants, if they are currently interruptible or not. Evaluating different machine learning algorithms, they reach 76.8% accuracy with a Support Vector Machine (SVM).

In the area of predicting interruptibility, many researchers also look into *identifying breakpoints* [Okoshi *et al.* 2017], the brief moments that are opportune for presenting non-time-critical notifications [Hashimoto *et al.* 2013; Ho & Intille 2005; Iqbal *et al.* 2005; Poppinga *et al.* 2014; Tanaka & Fujita 2011]. For example Ho and Intille [2005] use body-worn wireless accelerometer sensors to detect activity transitions in real-time in a study. Each time an activity transition is detected, they ask the participant "How receptive are you to a phone call?" The participants could answer via a PDA on a scale from 1 ("not at all receptive") to 5 ("extremely receptive"). As a control condition, they also randomly triggered the participants with the same questions. They found that the participants were more receptive to messages delivered at activity transitions than at random points in time.

A system called *InterruptMe*[1] [Pejovic & Musolesi 2014] uses activity, location, time of day, emotions and engagement, to identify suitable moments for interruptions. The authors also used an Experience Sampling study to collect data from 20 subjects, resulting in 906 collected answers. In an explorative machine learning approach, the data was analysed, and based on those findings, an online learning system was implemented in form of an Android Library. While the prediction performance is not outstanding, the system is one of few that actually was evaluated with real users.

An exception of course is the large-scale study with over 680,000 user of the Yahoo! JAPAN Android app [Okoshi *et al.* 2017]. For the study, they embedded their interruptibility estimation logic into this widely used app and adapted the delivery times of real notifications of the Yahoo! Service based on this logic. In their study they found that the response time is significantly reduced (49,7%) when applying breakpoint-based adaptive notifications and overall the user engagement level with the application increased.

2.4.6 Summarising the Related Work

From this presentation of related work we can see that various approaches and concepts have been already developed in order to support presence and availability sharing and management. Along the way, a number of insightful lessons have been learned by the community, on how to build such systems. And even though *Harr & Wiberg* [2008] introduced this clear categorisation, we see that in many research prototypes different approaches are combined.

Hence, we can derive from the state of the art several areas of improvements. It is clear, that the manual adaptation of one online state only provides a limited expressiveness. While the mechanisms to specify *selective* disclosure of awareness information to different users account for actual user needs, current approaches are often tedious and complex to configure. Such mechanisms also do not account for the *dynamic* nature of privacy, where preferences continually change over time.

The automated calculation on the other hand seems to be promising in adapting to the dynamic nature of privacy. However, in all presented related work on automatic adaptation, nobody investigated the need for *selective* availability. Also, the settings in which the approach was evaluated were often rather simple and partially controlled. And finally, only very few have shown how the predictability of availability can be implemented in a real adaptive system.

---

[1] Please, do not confuse with *InterruptMe* by Hincapié-Ramos *et al.* [2011] which is discussed earlier.

2.5 Conclusions

In this chapter, I aimed to provide a sound understanding of the concepts of presence and availability, their grounding in HCI, the multifarious requirements around managing such delicate privacy aspects, and the many different solutions so far proposed to tackle this.

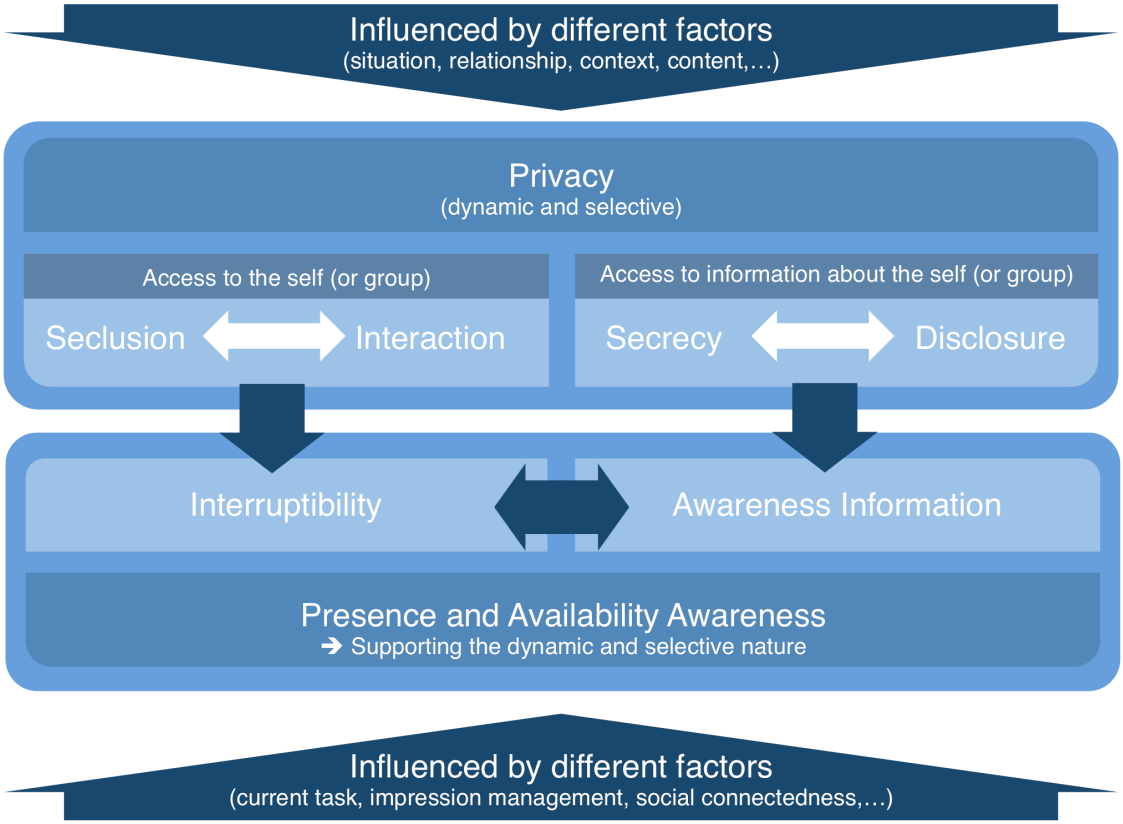

Figure 17. Illustration on how the two main functions of privacy (cf. Figure 3) relate to the concepts underlying presence and availability awareness. It highlights, how interruptibility and awareness information are dependent on the two main privacy functions but also influence each other.

I now finally can relate our understanding of *privacy* and its two main functions as they were illustrated in Figure 3 with the concepts underlying *presence* and *availability* awareness (cf. Figure 17). At this point we are able to conclude that the concept of *interruptibility* is highly influenced by the privacy function of regulating *access to the self*. On the other hand, the provision of *awareness information* is affected by the privacy function of regulating *access to information about the self*. Accordingly, they also inherit the *dynamic* and *selective* nature of privacy, which consequentially needs to be supported by systems for presence and availability awareness. As Patil and Kobsa [2009, p. 187] formulated, the benefits of awareness systems "typically come at the cost of the risks of reduced privacy for the individuals whose information is disseminated."

Furthermore, *interruptibility* and disclosure of *awareness information* also mutually influence each other, as it was formulated in the *dual trade-off* between privacy and awareness, and between awareness and disturbance [Hudson & Smith 1996]. By providing more information about the self, the likelihood of being interrupted in inappropriate moments is reduced. From what we have deducted so far, we actually can reformulate the *dual trade-off* as a *privacy-privacy trade-off*—a trade-off between *privacy* (as in seclusion) and *privacy* (as in secrecy).

By summing up, it becomes evident that the approach of *explicit availability management* does not address well the *dynamic* nature of privacy, and also falls short for supporting the *selective* aspects. That is, the constant manual adaptation of several online states is exhausting and tedious for the users.

When looking at the approaches for *implicit availability management*, some research prototypes in the related work provided functionality to configure the *presentation of awareness information* in a way that it supports human needs for *selective* information disclosure. In some of the presented systems different types of information, or at different levels of detail, can be revealed to different persons. By enabling the contacts, to draw conclusions from the provided up-to-date awareness information on the current interruptibility, these systems also partially support the *dynamic* nature of *privacy*, in terms of *access to the self*. However, such systems fail to support the *dynamic* nature of *information disclosure*, as the provision of awareness information does not adapt to the up-to-date privacy needs of the discloser. To deliver an example: While sharing the personal location to the significant other might be an acceptable trade-off for minimising the interruptions during day-to-day life, there still might be some situations were the location reveals unwanted information (e.g., the visit to a jeweller for buying a gift). As the negative effects of being interrupted are sometimes short-lived, the negative effects of revealing unwanted information can have a long-term impact. Consequently, the *implicit availability management* through the *presentation of awareness information* also can be challenging.

Hence, in this work, I propose means to automate the *Selective Availability* by applying machine learning, thus supporting the selective and dynamic nature of privacy adequately. Accordingly, the work falls in the category of *implicit availability management* through *automated calculation*. From a human-centred perspective, it is clear that an approach of *automated calculation* is not ideal in every sense. To start with the critique, Brown and Randell rightfully state in their guidelines for building a context sensitive telephone [2004, p. 334] for "every situation, there is an exception". Also others share this assumption, that *automated calculation* is not ideal, as interruptibility is "not a state of the world (or of the person) that can be simply inferred from sensor data" but "a result of social construction among the interrupter and the interruptee and interruptibility therefore is not visible per se but constructed in the minds,

actions, and interactions among the involved parties." [Oulasvirta et al. 2005, p. 172]. While I can agree with this perspective, I would add the following: These mentioned processes of social construction are between people—hence, the assessment of the interruptibility by an interrupter and the actual interruptibility of an interruptee also not necessary align in their result. There is also the chance of misinterpretation. And this is already true in settings that are free from technological mediation—as for example when entering somebody's office—as was shown by Avrahami et al. [2007]. Furthermore, this happens despite the fact that the expressiveness of social cues we have in real life are much richer than what is usually offered as *presentation of awareness information* in technology-mediated communication. On the other hand, the approach of automated calculation can incorporate much more information in a prediction, which neither can be easily observed nor interpreted ad-hoc by a person.

In a way, when looking at the technology that is currently available, it is interesting how little of this decades of research has found its way into actual software products, while on the other side the communication and information load is steadily increasing. It looks as we gave up on managing our availability and interruptibility. Hence, I would argue that, even though the *automated calculation* is not always correct, it is a least helpful at the moments when it is correct, and therefore an improvement of what is currently available. For the times when it is not correct, a thorough understanding of user needs can help to implement a system, that minimises the negative consequences for the users, for example by building in functionality that ultimately leaves the users in control.

3 Understanding User Needs for Selective Information Disclosure and Availability

We earlier learned, that in our daily face-to-face interaction with different people of different social groups we constantly adopt our behaviour, our mimic, gestures, and language, as well as the information we disclose depending on the image we want to convey. On the other hand, in current awareness support systems our possibilities to do so are often limited. On the one hand, computer-mediated communication lacks certain channels, which does not allow transporting all the nuances of human interaction. On the other hand, manually adapting all the available channels in order to manage the impression we want to make for each individual is a cumbersome effort. I present two studies on user needs that provide us with a better understanding of human-centred requirements for selective information disclosure and availability.

3.1 Patterns in Selective Information Disclosure

In my analysis of related work I showed that the presentation of awareness information (i.e., information disclosure) comes with clear challenges for the privacy of the individual users. Even though some suggested systems provide fine-grained controls for configuring privacy settings, the process of configuring such preferences in computer-mediated communication systems like instant messaging or presence- and awareness services is very cumbersome and time consuming for the users. Boyle and Greenberg [2005] discuss the trade-off between the necessity to give the users fine-grained and precise control over what information is disclosed to whom in which detail and at the same time to make these mechanisms lightweight and effortless—two requirements that were also previously formulated by Bellotti [1998]. Also Richards and Christensen [2004] stress the complexity of permission control and argue for new ways to address these concerns with easy and intelligible mechanisms. Herbsleb et al. [2002] also found in a study of their presence and availability awareness tool *Rear View Mirror* that making the privacy setup tremendously difficult and time-consuming lead to a rejection of the tool.

Accordingly, in order to further extend the efficiency of selective information disclosure the concept of *Disclosure Templates* [Gross & Fetter 2010] is introduced. The aim is to reduce the extra effort for users when configuring their preferences for selective information disclosure for the first time, by relying on a number of predefined, common patterns in form of templates.

In order to identify such patterns, a literature study and an online survey was conducted. In the Following I first provide further details on the literature study,

the survey and its results, and then briefly explain how it informed the concept of *Disclosure Templates*.

3.1.1 Concept of Disclosure Templates

The concept of *Disclosure Templates* [Gross & Fetter 2010] departs from the understanding that people in their daily life cluster the individuals they interact with in different social circles. These circles often come with different preferences for information disclosure. Olson *et al.* [2005] revealed in their study on selective information disclosure that there are dependencies between such groups of people and the different kinds of information that individuals disclose to them. Accordingly, they identified clusters of recipients and clusters of information types that are treated similar by their subjects in respect to disclosure. Others confirm this with similar findings [Davis & Gutwin 2005; Lederer *et al.* 2004; Patil & Kobsa 2004; Patil & Kobsa 2005; Patil & Lai 2005]. Based on this presupposition the concept of *Disclosure Templates* was developed. Upfront I introduce the definitions of four central elements to the information disclosure process: the *information* that is disclosed (i.e., personal data that can be made accessible to others); the *discloser* of the information (i.e., the person that makes personal data available to others); the *recipients* of the disclosed information (i.e., persons that can access the data disclosed by a discloser); and the *precision level* of the disclosed information (i.e., the degree of exactness and completeness the information is disclosed with).

Based on these definitions ten *information types* for the presentation of self, as they are typically used in research on computer-mediated communication and presence- and awareness systems (cf. section 2.4), were selected. I distinguish here between six low-level information types that can be easily captured via sensors or user input and four high-level information types that need to be inferred by using more complex approaches like machine learning or by combining data from several sensors. The low-level information types are:

- *Personal information* [Lederer *et al.* 2004]: Basic data that can be input by the discloser like the name, office postal address, private email address, or work telephone number. This information type can for example help to choose the right mode of contacting the discloser.
- *Location* [Lederer *et al.* 2004; Patil & Lai 2005]: Data on the whereabouts of the discloser that can easily be captured with GPS sensors, which are ubiquitously available in today's smart phones. This information type can be valuable when the recipient wants to physically meet the discloser, but also can give insights on the activity of the discloser.
- *People in proximity* [Lederer *et al.* 2004]: Information if and what other users are in a short geographical distance (e.g., the same room) as for example reported by an indoor location system. This information types

allows the recipient to draw conclusions about the social context of the discloser.

- *Calendar* [Patil & Lai 2005]: Data about the upcoming schedule of the discloser retrieved from their digital calendar. This information type can be helpful for the recipient for example to schedule a meeting or postpone a contact request.
- *Phone status* [Patil & Lai 2005]: Information about the state of the telephone giving insights whether the discloser is currently speaking on the telephone or not. This information type can also help to estimate if it is a good time to contact the discloser and can also suggest other modalities for contact in urgent cases.
- *Applications* [Patil & Lai 2005]: Data about the applications that are currently running on the computer of the discloser. This information type allows drawing conclusions on the current tasks of the discloser.

For the high-level information types the following were selected from the literature:

- *Computer activity* [Patil & Lai 2005]: General data about the disclosers interaction with the computer as it can be inferred by e.g., combining information on the current front-most application, input idle times, current used network, etc. This information type can allows insights about the discloser's current context and also can suggest adequate means for contacting the discloser.
- *Activity* [Lederer *et al.* 2004; Patil & Lai 2005]: Data that reveals details on the current actions of the discloser. Such data can for example be retrieved via activity recognition from sensor data of body-worn accelerometers [Bao & Intille 2004] or from the interaction with RFID tagged objects [Philipose *et al.* 2004]. This information type can lead to a very detailed understanding of the activities of the discloser and so allows for a good appraisal of the discloser's situation.
- *Person in conversation* [Patil & Lai 2005]: Information about whether, and with whom the discloser is currently in a conversation inferred by means of voice activity detection and speaker recognition. This information type may help the recipient to assess if an interruption of the discloser is currently appropriate.
- *Availability* [Patil & Lai 2005]: Data on the current availability of the discloser based on the combination of several information types. This information types provides insights if it is currently possible to establish contact with the discloser.

In order to specify the degree of exactness and completeness of disclosed information also four ordinal precision levels [Lederer *et al.* 2004; Patil & Lai 2005] were identified:

- *Precise*: The information is disclosed unchanged—exact and complete.
- *Approximate*: The selected information is disclosed exact but incomplete.
- *Vague*: The selected information is disclosed rounded and incomplete.
- *Undisclosed*: No information is disclosed.

Based on these definitions I am now able to describe the seven *Disclosure Templates*. A *Disclosure Template* thereby is an epitome of permutations of information types and precision levels, directed at an archetypical group of recipients the discloser trusts in the same way. A *Disclosure Template* therefore can be seen as a typical configuration for the disclosure of information directed at recipients that belong to a specific category of trust. In order to name these *Disclosure Templates* labels were needed, that correspond with these categories of trust (i.e., a *Disclosure Template* is named after a group of people that are typically recipients of such information). Seven categories were identified, based on different studies [Davis & Gutwin 2005; Lederer *et al.* 2004; Patil & Kobsa 2005; Patil & Lai 2005] researching the ability of people to classify their information disclosure and have been verified with an own study. The seven categories in descending order of the overall disclosure are: *Family, Friends, Partner, Team, Superiors, Subordinates,* and *Public* (cf. Table 2).

In order to assign specific, epitomical configurations of information types and precision levels to these seven categories that form the seven *Disclosure Templates* a user study was conducted.

Table 2. The seven identified categories of trust: *Family, Friends, Partner, Team, Superiors, Subordinates,* and *Public*.

	Definition	**Examples**
Family	Information disclosure to persons standing in a closer family relationship with the person	Father, Children, Grandmother
Friends	Information disclosure to persons with a close, personal relationship with high sympathy and a high degree of trust	Good friends, Confidante
Partner	Information disclosure to the significant other of the person	Boyfriend, Fiancé, Wife
Team	Information disclosure for groups of persons that are united by a common goal	Members of project group, Soccer teammates
Superiors	Information disclosure to persons that the person reports to	Manager, Coach
Subordinates	Information disclosure to persons that report to the person	Employee, Apprentice
Public	Information disclosure to other person, independent of the level of familiarity	Fellow passenger on the bus, Passers-by in the mall

3.1.2 User Study and Results

The study was conducted in form of an online survey. The subjects for this study were recruited via word-of-mouth, announcements in courses and email

invitations. From 56 registered subjects, 40 completed the survey and 38 provided complete information that was analysed. The subjects were predominantly male university students (12 female, 26 male) and in average 23,6 years old.

The survey itself consisted of four parts and a short introduction upfront. In the introduction a written scenario on computer-mediated communication, contact management, and information disclosure was used to familiarise the participants with the topic and give them a proper setting. The first two parts were concerned with verifying the seven categories of trust that were identified in the literature study.

First, the subjects were asked to cluster their social environments into seven categories of trust and provide label for these categories. In the second part the participants were required to select an adequate precision level for each of the ten information types for each of the seven own categories they came up with. The provision of a description for each information type and a concrete example for each precision level helped them to accomplish this task. In the next part of the study the participants were again asked to specify precision levels for all ten information types—but this time the seven categories were given: *Family*, *Friends*, *Partner*, *Team*, *Superiors*, *Subordinates*, and *Public*. In the last part the effect of feedback [Bellotti & Sellen 1993] on information disclosure behaviour was evaluated. Some studies (e.g., [Patil & Kobsa 2005]) found that revealing the actually disclosed information to the discloser leads to an adaptation of the disclosure configuration. In order to identify similar effects in this study, the participants were provided with a concrete example on what they would currently reveal to their recipients based on their answers in part three. They were then invited to alter the answer they gave in part three if they felt like they are revealing too much or too less.

Table 3. Matching of the seven categories with category labels given by subjects.

	Matching (%)
Family	73,7%
Friends	100%
Partner	10,5%
Team	65,8% (university related) 55,3% (work related)
Superior	5,3%
Subordinates	0%
Public	28,9%

The answers of the participants allowed advancing the *Disclosure Templates* concept. While the sample size does not allow for making generalisable statements and universally valid templates, the study helped developing the concept.

The first step was to verify if the seven categories of trust are reasonable choices, by checking if they are also reflected in the answers of the users in the first part of the survey. Therefore the occurrence of labels for each category over the entire number of participants was counted. In this process synonyms were taken into account. So for example entries like 'soccer' or 'band' would be assigned to the category *Team*. The results in Table 3 show the percentage of the participants' own labelled categories that are comparable to one of the seven categories based on the semantic of the label. For example, 73,7% of the participants (i.e., 28 participants) had a category that was labelled 'family' or had a label of similar connotation.

In order to further inspect the mapping between categories provided by the participants and the seven provided categories, the similarity of the permutations of information types and precision levels of the second part and the fourth part were analysed. This was done for the categories that matched with more than 50%, in particular Friends, Family, and Team. This test revealed that for the three categories in average the precision levels for nine out of ten information types matched. This coherency can strengthen the conclusion that the chosen categories for the Disclosure Templates are relevant, as they not only match on basis of the labelling but also in regards to their form with the user labelled categories.

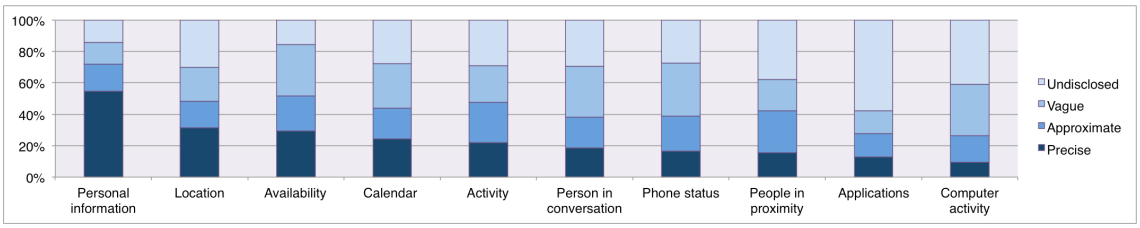

Figure 18. Distribution of the selected precision levels for the ten different information types independent of the categories.

Figure 19. Distribution of the selected precision levels for the seven different categories independent of the information types.

Overall—as expected—the given answers revealed that the users have different privacy needs based on the information type of the revealed information (cf. Figure 18) and the recipients of the information (cf. Figure 19).

Further the study revealed that tendencies for specific recipients in combination with specific information types exist.

Based on these findings the configuration of the seven *Disclosure Templates* was finalised as depicted in Table 4. The configurations are based on the answers of the study that were given in part four. The precision levels Precise (4,3%) and Undisclosed (21,4%) appear overall less often than Vague (30%) and Approximate (44,3%).

3.1.3 Interaction Concept and Conclusion

In order to make the concept of *Disclosure Templates* accessible to users it was integrated into the *FamilyFaces* [Fetter & Gross 2008] application. There it supports users by lowering the threshold when configuring the client the first time for selective information disclosure. The templates can be used as a basis for quickly setting up a first generally adequate configuration, which then can be fined-tuned to personal preferences.

Table 4. Seven Disclosure Templates (Precise=P, Approximate=A, Vague=V, Undisclosed=U).

	Family	Friends	Partner	Team	Superiors	Subordinates	Public
Personal information	P	P	P	A	A	V	U
Location	A	A	A	A	V	V	U
People in proximity	A	A	A	A	V	U	U
Calendar	A	A	A	A	V	V	U
Phone status	A	A	A	A	V	U	U
Applications	V	V	V	V	V	U	U
Computer activity	V	V	V	A	V	U	U
Activity	A	A	A	A	V	U	U
Person in conversation	A	A	A	A	V	V	U
Availability	A	A	A	A	V	V	U

The approach of *Disclosure Templates* minimises the effort for the user and thus provides a *lightweight* solution for the configuration of privacy preferences, thus fulfilling requirements for privacy preserving systems formulated by [Boyle & Greenberg 2005] and [Bellotti 1998]. Overall the work aimed at exploring the design space for lightweight mechanisms for privacy control. The sample size of the survey does not allow for postulating universally valid templates, yet enabled us to illustrate the underlying concept. A long-term study on the acceptance of *Disclosure Templates* as well as an in-depth study of the efficiency are out of the scope of this thesis and remain future work. To cope with a larger variety of

information types work towards a taxonomy of information types as well as heuristics for the operationalisation of precision levels will be needed.

While the proposed approach greatly reduces the effort for configuring a system for availability management, it does not help to deal with the dynamic nature of privacy. While the proposed templates reflect the privacy preferences in many situations, they can never be adequate for every situation. In this sense, the *Disclosure Templates* only can be seen as a good compromise or a first approximation.

3.2 An Experience Sampling Study on Selective Availability

In order to get more insights into actual user needs in regard to the dynamic nature of selective information disclosure I narrowed the perspective in respect to the type of information that is disclosed, and conducted a study focussing on *Selective Availability*[1]. The following section briefly describes the concept of *Selective Availability* and gives a short introduction to the Experience Sampling Method (ESM). Further, I provide details on the design and the execution of the study—including a small pre-study that helped to define some constructs for the main study—and conclude with the presentation of the study results and their discussion.

The concept of *Selective Availability* departs from the basic assumption that users only have one availability state for all their contacts, as it is manifested in most current IM systems. An IM system with *Selective Availability* would allow the users to have distinct sets of contacts, each with a separate online state that reflects the user's availability towards this group of people. The possibility of expressing the availability differently towards different groups of contacts in computer-mediated communication better reflects how people deal with incoming requests in their daily lives, where we also adapt our availability based on the persons that request our attention. Patil and Kobsa confirm a general need for *Selective Availability* in computer-mediated communication from their observations of IM usage. In their paper based on interview findings of practices of people using different IM accounts for work-related and personal contacts, Patil and Kobsa [2004] state that these "practices point to a desire for different levels of availability for different groups of people—such as co-workers, family, friends". Voida *et al.* [2002] revealed the same sort of practices in a study. These practices are often workarounds to achieve selective availability with current technology, for example by using multiple accounts.

---

1 It should be noted that in other context the term *Selective Availability* describes the intentional degradation of GPS signals by adding artificial noise through the U.S. government for reason of national security before the year 2000.

In the study the actual need for *Selective Availability* of IM users was researched over the duration of four weeks with the help of the Experience Sampling Method (ESM) and combined with an extensive collection of sensor data for later use. The ESM stems from the field of social psychology and was created by Csikszentmihalyi [Csikszentmihalyi & Larson 1987; Hektner *et al.* 2006; Kubey & Csikszentmihalyi 1990]. The method enables researchers to better capture the inner states, feelings or reactions of individuals towards experiences in the moment they occur. ESM particularly is a well-established method when systematically looking for within-person differences and between-person differences for internal phenomena as well as for understanding fluctuations for individuals across time. Among the advantages [Barrett & Barrett 2001; Thomas & Azmitia 2016] of ESM is the reduction of retrospective biases and issues with forgetting. The probability of sampling authentic experiences is increased by the randomised nature of this method. The participants of an ESM study are being signalled in some form (e.g., by a stopwatch, pager, or PDA) during their normal daily routines to record (e.g., write down) their current feeling or inner state towards a specific research question. This signalling is repeated several times over the course of the day and usually lasts for several days or weeks. Through its repetitive character, the method is able to capture rich and in-depth data. The method is especially suitable in situations where the data that needs to be collected is not directly observable (e.g., feelings or thoughts of the subject). It therefore was adapted to and used in HCI research in different ways:

- *In-situ field studies*: ESM has been used in HCI research as a means for in-situ field studies in the design and requirement analysis phase. For example, the ESM was used to get insights on information needs and available infrastructure of users to inform the design of personal servers [Consolvo & Walker 2003], or to get insights on the availability and interruptibility in work life [Hudson *et al.* 2002] and home life [Nagel *et al.* 2004].
- *Context-Aware Experience Sampling*: Context-Aware Experience Sampling (CAES) is an extension to the normal ESM approach. This method leverages ubiquitous computing technologies to trigger an experience sampling, each time an event of interest is detected. This approach was pioneered by the Massachusetts Institute of Technology [Intille *et al.* 2003] and pursued by tools like the *My experience* (also *MyExperience*) toolkit [Consolvo *et al.* 2007; Froehlich *et al.* 2007] as well as in my own research (cf. 6.2.1) [Fetter & Gross 2011b; Fetter *et al.* 2011a]. The CAES approach relies on the usage of sensors to detect events of interest and to trigger the according experience sampling. The sensors—usually embedded in the subjects mobile device—

request a sampling from the subject when specific criteria are met—for example when the subject arrives at a specific place.

- *Collecting labels for Machine Learning*: Finally, ESM has been applied to acquire labels for the hidden states of computer users, to build predictive models on the sampled data by combining it with data from sensor logs [Horvitz *et al.* 2004; Kapoor & Horvitz 2007]. In this approach, the subjects mostly are triggered to choose one of the several possible answers or a value on a rating scale. The answers are later combined with the data from sensors, and together are used as training data for a machine learning task.

In the study the ESM was used in form of an in-situ field study to get insights into the actual requirements and user needs for *Selective Availability*. In contrast to an experiment in a controlled setting the in-situ approach enables us to collect real data on the subjects' availability preferences, as these preferences are strongly influenced by environmental factors that cannot be simulated in a controlled environment. ESM was chosen, as the momentary availability preferences of individuals are not directly observable.

Further information in form of sensor logs was collected. This information, on the one hand, allowed us to compare it to the data from the experience samplings in the analysis phase, as I did for the Skype sensor. On the other hand, this data enabled me in a further step to explore the predictability of *Selective Availability* states through machine learning which is discussed in details in section 4.1. In the next section, I only report on those results and details of the study that are concerned with insights on user requirements and needs towards *Selective Availability*.

3.2.1 Study Design—Availability Levels and Availability Categories

First, I introduce the constructs used in the study, namely *Availability Levels* and *Availability Categories*. I begin with characterising these notions and further show how they helped to normalise the insights obtained in the subjects' availability preferences. I also show how these constructs were identified and validated in a small pre-study.

3.2.1.1 The Concept of Availability Levels

In order to assess the users' needs for Selective Availability a measurement is necessary, which allows the subjects to indicate their current availability towards others. In related studies on the interruptibility of users a five-point Likert scale (i.e., from 1—highly interruptible to 5—highly non-interruptible [Fogarty *et al.* 2004a]) or a dichotomous variable (i.e., "Busy" and "Not Busy" [Horvitz *et al.* 2004]) was chosen as the measurement. While the Likert scale allows for a more fine-grained expression it is also more abstract than the simple binary options

"Busy" and "Not Busy". I therefore looked for a measurement that comprises both qualities: more concrete for the subjects and allowing for fine-grained data collection. As the study aimed at subjects with long-term IM experience, the prevalent *Availability Levels* that exist in most IM systems as a means to measure the subjects' availability were chosen. The underlying assumption is that IM users are familiar with expressing their availability preferences with these means. Further, in the subsequent work, where we look on the predictability of user's online states, the aim would be to predict exactly those *Availability Levels* in order to automatically adapt the online status of the users IM system.

Availability Levels signal, in form of the online status of an IM application, to what extent a user is currently available for communication. For a person that plans to contact a specific user, the set *Availability Level* reflects this user's expectable answering behaviour. As predefined *Availability Levels* exist in every IM application, frequent IM users are versatile in their usage. For example, the Extensible Messaging and Presence Protocol (XMPP) [Saint-Andre 2011b] specifies six *Availability Levels*: besides the two basic presence states *available* and *unavailable* the `<show/>`-element allows for four further "availability sub-state"s. These four sub-states described in paragraph 4.7.2.1 of the IETF[1] RFC[2] for XMPP [Saint-Andre 2011b] are: *away*, defined as "The entity or resource is temporarily away"; *chat*, defined as "The entity or resource is actively interested in chatting"; *dnd*, defined as "The entity or resource is busy (dnd = 'Do Not Disturb')"; and *xa*, which is defined as "The entity or resource is away for an extended period (xa = 'eXtended Away')". All of these six basic *Availability Levels* can be completely or partially found in current IM systems. For example, in *Skype* [Skype Limited 2011] the possible levels are *Skype Me!™*, *Online*, *Away*, *Not Available*, *Do Not Disturb*, *Invisible,* and *Offline*. On this basis I decided for the following six *Availability Levels* with the given underlying definitions in the user study:

- *Text Me!*—The *Availability Level* Text Me! expresses a strong desire of the users for communication with members on their buddy list. Incoming Messages will be answered immediately. This *Availability Level* resembles the *Skype Me!™* of the Skype client respectively the level *chat* defined in XMPP. In Skype this status also has the special function, to be open to chat invitations from users that are not on the users' buddy list.

---

1 IETF stands for Internet Engineering Task Force—A Organisation concerned with the development and promotion of Internet standards

2 RFC stands for Request for Comments–A form of document for describing technical or organisational concepts around the working of the Internet. Some of these RFCs evolve into Internet standards.

- *Online*—This *Availability Level* signals that the users are fully available for communication requests and able to participate in chat sessions. A contacting person can expect a short response time.
- *Away*—With an online status set to *Away* the users signal that they are generally willing to answer incoming requests. However, a short on-going concurrent task might slightly delay the users' ability to answer.
- *Not Available*—The users signal that they currently are involved in other activities of certain duration. While they are generally able to give a quick response to a contacting person, they are not able to answer incoming requests to full extend and probably suggest postponing the communication.
- *Do Not Disturb*—The users are currently only available for very brief communication because they are involved in a concurrent task that requires full concentration. Incoming communication requests will be answered very briefly. For example, a user would only quickly type "In a Meeting!", to give further awareness to the contacting person.
- *Offline/Invisible*—In the study, I treated both Availability Levels similar, as I was interested in what the users want to signal to people that want to initiate a conversation. If users are *Offline*, they are not connected to the network and therefore do not receive incoming messages. If users are *Invisible*, they are technically online but withhold their actual online status. For a contact, the user in both cases appears to be offline, and so in both cases would not expect a timely answer. The *Invisible* status thereby is a special status that goes along with the concept of Plausible Deniability [Aoki & Woodruff 2005; Nardi *et al.* 2000] wherein recipients can read an incoming message, but decide to not answer this message and act like they never received this message, or received it later.

The *Availability Levels* were designed to imply a form of ordinal scale. In the study, the participants were instructed to assume an underlying order in respect to the amount of time it would take them until they would answer an incoming Instant Message.

3.2.1.2 Defining Availability Categories through a Pre-Study

Further, in order to research the concept of Selective Availability, it was necessary to give the subject means to express their current availability towards different recipients. As expressing selective availability on a level of individual recipients is too laborious, and as it would not allow for later comparison between subjects, the concept of *Availability Categories* was introduced. An *Availability Category* (AC) is defined as a set of contacts to which a user is available in the same way. For example, an AC could comprise all colleagues of

the user, that have in common that the user is available to them during normal working hours, but appears offline to them in the evening and weekends. Much similar to a Face, an AC describes a certain privacy need. It allows controlling the degree of interruptibility of a person with respect to a specified audience.

While the users normally would assign their own labels to an AC for easier management, for the study it was necessary to come up with a small number of predefined and universal ACs to keep the results comparable among the participants. I found the seven categories of trust (i.e., Family, Friends, Partner, Team, Superiors, Subordinates, Public)—which were already the foundation of the Disclosure Templates (cf. 3.1.1)—to be a good starting point for defining the ACs for this study. However, as the study involved repeatedly answering the questions regarding the availability towards all ACs, I aimed at further restricting the number of ACs in order to make the study as effortless as possible for the subjects. So, in order to further reduce the number of ACs from the seven trust categories a small pre-study was conducted, which is explained in the Following.

In the pre-study, a paper-based questionnaire was used in combination with a card sorting approach, in order to reduce the number of ACs. The aim was to reduce the number to at most four and at least three categories. The study consisted of three parts, whereby I here focus on the main part that was concerned with reducing the number of ACs. The other parts of the study were concerned with the subjects' exposure to computer-mediated communication in general and the collection of demographic data.

In the main part the basic idea behind the card sorting technique [Spencer 2009] was used. The card sorting technique is usually applied to identify structures in a previously unsorted list of concepts. Thereby a group of subjects is asked to sort a set of concepts written on index cards, to form clusters or a structure the subjects are familiar with. In the pre-study the participants were asked to group the seven categories of trust (*Family, Friends, Partner, Team, Superiors, Subordinates, Public*) into three (Task A) respectively four clusters (Task B) that later shall form the ACs for the study. Further the subjects were asked to write down a descriptive label for each cluster. As it was a paper-based questionnaire, the participants were asked to mark the concepts belonging together with a circle (instead of using index cards). This was a feasible option, as only a low number of concepts (i.e., the seven categories of trust) had to be sorted. Finally each participant rated on a five-point Likert scale which of the two resulting sets of ACs is more suitable: The results from task A with three or from task B with four ACs.

30 participants (12 female) between 19 and 30 years old (M=25,4) completed the questionnaire. 26 of the participants were currently studying graduate or undergraduate courses at the University while 19 of the students studied

computer science. Most of the participants were versed in using computer-mediated communication tools as the majority of them stated using instant messaging and email on a daily basis. The participants came up with 24 labels to refer to the three clusters in task A and respectively 32 labels in task B. The most common labels used in task A were: work (20x), private (17x), and public (11x). In task B the most common were: work (14x), friends (13x), and private and public (both 12x).

In order to identify the most generic availability clusters, a cluster analysis of the data in SPSS and the Weka toolkit [Hall *et al.* 2009] was performed. Therefore the answers given by the participants in the card-sorting task were arranged in form of 8-dimensional feature vectors. While the first seven features are binary attributes (true/false) for the seven trust categories in the order Family, Friends, Partner, Team, Superiors, Subordinates, Public, the eighth feature holds the descriptive label the participants choose. Accordingly the vector ac={f,f,f,t,t,t,f,"Office"} denotes a proposed availability category with the three selected (t=true) trust categories Team, Superiors, Subordinates which the subject labelled as 'Office'. For each of the 30 subjects three vectors were built for Task A (90 vectors) and four vectors for Task B (120 vectors) from the card sorting results. Based on this data a hierarchical cluster with Between-groups Linkage was performed with Squared Euclidean as the distance measure in SPSS. The resulting dendrogram (cf. Figure 20) allowed analysing similarities in the labelling of clusters. Further k-means clustering was used to partition the data into respectively three (Task A) and four (Task B) clusters.

The dendrogram depicted in Figure 20 shows the clusters for the results of Task A from the hierarchical clustering. The dendrogram is reduced in its depth to show the five largest clusters named C1 through C5. We are able to identify three major clusters: C1, C2 ∪ C3 and, C4 ∪ C5. Where C1 (32 labels), C3 (25 labels) and C4 (23 labels) are the largest and C2 (5 labels) and C5 (5 labels) are much smaller. We can see that some labels like friends are spread over more than one cluster, as their underlying concepts seem to differ. On the other side the concepts behind the labels work, private and public seem fairly stable as all (private and public) or most (work) occurrences belong to the same cluster. The labels that are given to the ACs in C1 reveal that most of them are concerned with private contacts like *friends* or *families* and the label *private* being the most dominant. The dominant labels *public* and *others* given in C2 ∪ C3 (e.g., *public* or *others*) allows to conclude that these clusters are comprised of contacts with which the subjects only have loose social interaction. Finally the labels used in C4 ∪ C5 suggest that mostly work-related contacts are grouped into this cluster. The analysis of the dendrogram for the results of Task B also revealed three major clusters that can be characterised as: private, public and work. The data showed that, although the subjects now could use four clusters, the fourth

cluster was too diverse to become relevant in the analysis. Hence only three major clusters could be identified.

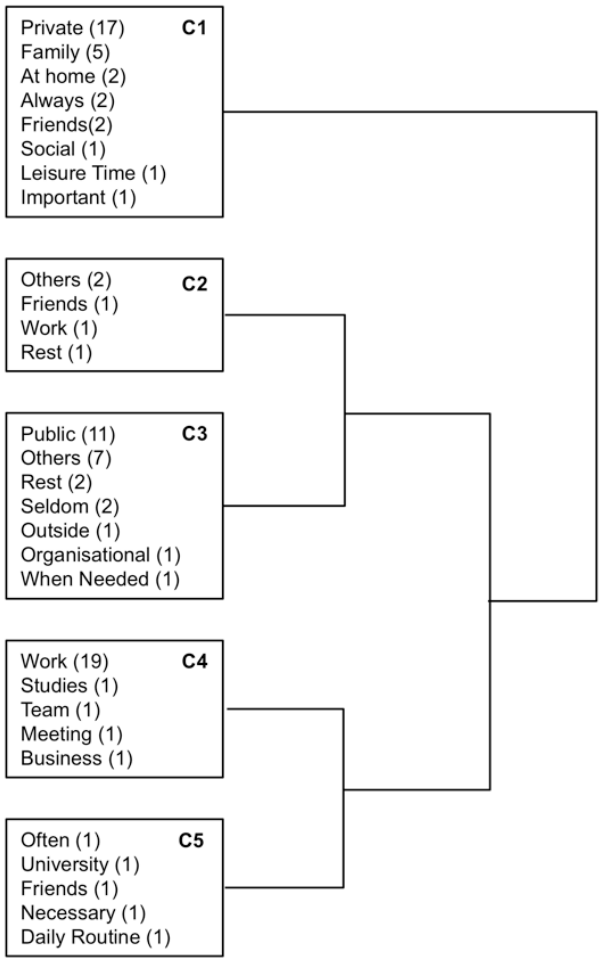

Figure 20. Dendrogram depicting the clusters for Task A. The numbers next to the labels depict how many items with this label are in this cluster.

Further, Weka toolkit's implementation of k-means clustering was used to cluster the results of Task A into three clusters (k=3) and the results of Task B into four clusters (k=4). The results in Table 5 show the centres of the clusters for Task A (A1-A3) and Task B (B1-B4). For Task A the clusters A1, A2 and A3 contained respectively 30, 31 and 29 of the altogether 90 items. For Task B the clusters B1, B2, B3 and B4 contained 39, 29, 31, and 21 of the altogether 120 items. The resulting clusters B1 and B2 from Task B thereby are similar to the clusters of A1 and A3, while the cluster B3 and B4—which together had the smallest squared Euclidean distance $d^2=2$—together form A2.

Both cluster analyses show that, while the results from Task A form three disjunctive clusters, the results from Task B show two clusters (B3 and B4) that are slightly overlapping (but include Family) and could be merged to form one cluster. As B1 is similar to A1 and B2 is similar to A3, merging B3 and B4

would lead to a cluster that is similar to A2 and hence an identical result between Task A and B.

Table 5. Results of k-means Clustering for Task A and Task B, where a '✓' marks items belonging to a cluster and a '– ' marks items not belonging to a cluster.

	Clusters for Task A			Clusters for Task B			
Trust Category	**A1**	**A2**	**A3**	**B1**	**B2**	**B3**	**B4**
Family	—	✓	—	—	—	✓	✓
Friends	—	✓	—	—	—	✓	—
Partner	—	✓	—	—	—	—	✓
Team	✓	—	—	✓	—	—	—
Superiors	✓	—	—	✓	—	—	—
Subordinates	✓	—	—	✓	—	—	—
Public	—	—	✓	—	✓	—	—

The participants also rated how suitable the four respectively three clusters were for managing their privacy on a Likert-scale between not suitable (1) and very suitable (5). A paired t-test revealed that the results are not statistically significant ($t(29) = -0.556$, $p = 0.582$) to favour one approach over the other. Three categories were rated on average with 3.3 (SD=0.97) and four with 3.47 (SD=1.08). Accordingly, based on the results of the clustering and the reduced user effort, I decided for three ACs for the study. Based on the most prominently used labels I chose *Private, Work* and *Public/Other* as the names for these ACs:

- Private subsumes contacts to which the subject of the study has a personal, close, maybe intimate connection (e.g., significant others, parents, close friends, etc.)
- Work refers to contacts to which a work-related connection exists (e.g., colleagues, boss, subordinates, etc.)
- Public/Other collects those contacts, which do not fit in the previous two categories. (e.g., old school mates, former project members, etc.)

These three *Availability Categories* as well the six *Availability Levels* were used in the ESM study. In order to compare the data with some ground truth I also introduced the concept of *General Availability*. This *General Availability* represents the availability towards all contacts, and refers to the online state a person would choose in a current IM solution with only one online state.

3.2.2 Study Design—Technical Means for the Data Collection

Besides the some form operationalisation and measures for the object of investigation, another important aspect was the technical setup of the ESM study. As I was also interested in collecting rich sensor data together with the ESM data, the elaboration of the study design involved the conception of the software artefacts for collecting the sensor logs as well as the experience

samples. The aim was to have a single, self-contained application that collected the necessary data on participant's computers while being as unobtrusive as possible to the subjects and their daily tasks. As none of the few available solutions (e.g., myExperience [Froehlich 2009] or ESP [Barrett & Barrett 2014]) supported my approach and the targeted platforms, I came up with an own solution.

For the collection of the sensor logs the PASS Sensor Daemon from the PRIMI Advanced Sensor Suite (PASS) was used. PASS is described in more detail in 4.4 (from a conceptual perspective) and in 5.3.2 (from an implementational perspective) and therefore is only briefly introduced here. The PASS Sensor Daemon—developed by us in Java—allows continuously collecting sensor data on a computer in form of an unobtrusive background process. From the various options the PASS Sensor Daemon offers for making the collected data persistent, the option of XML logs was used in the study. Each collected sensor-event is stored in form of a short XML sequence in a larger XML file. The PRIMI Advanced Sensor Suite comprises a growing number of sensors[1] that monitor different aspects of the hardware and software of a personal computer that in some way allow to draw conclusions on the usage context of this computer. These sensors can be utilised by the PASS Sensor Daemon via a plugin-mechanism. At the time of the study the following 29 sensors were available and accordingly used in the study: Active Access Point Sensor, Active Chats Sensor, Active Network Interfaces Sensor, Ambient Light Sensor, Applications Sensor, Application Focus Sensor, Battery Sensor, Bluetooth Devices Sensor, Calendar Sensor, Connected FireWire Devices Sensor, Connected USB Devices Sensor, CPU Sensor, Email Sensor, Ethernet Connected Sensor, Face Detection Sensor, Focus Title Sensor, Headphones Connected Sensor, Input Idle Sensor, IP Address Sensor, Volumes Sensor, Motion Sensor, Mouse Connected Sensor, Power Connected Sensor, Screensaver Active Sensor, Second Monitor Sensor, Skype Sensor, Voice Activity Sensor, Volume Settings Sensor, WiFi. The values of a 30th sensor, the Time-Sensor, was synthesised. A detailed description for each of these sensors is given in 5.3.2.1. Depending on the type of sensor, the sensors collect data either when the observed phenomena changed or when triggered in a fixed interval.

The experience samples were collected via a small modal dialogue that popped up on the subjects' screen in a specified interval. From the technical perspective the PRIMI Advanced Sensor Suite infrastructure was utilised, and the ESM dialogue accordingly realised in form of a sensor plugin. This way the experience sampling data could be easily collected together with the data from

1 Compare 4.4 for a definition of sensors in the context of this work.

the sensors in form of XML logs. From a conceptual perspective the experience sample collection was influenced by two further parameters: The design of the graphical user interface—the ESM sample dialog—and the timing of the sampling.

For the design of the dialog, I aimed to make the sampling as convenient and efficient as possible for the participants on the one side, and on the other side I took care to introduce as little response bias as possible through the design of the dialog. Each time the dialogue is presented, I wanted the participants to assess their *General Availability*, as well as the availability according to the three *ACs*, and indicate their current location. By iterating through several prototypes the usability of the dialogue was refined. For example, a first prototype used a dropdown menu (cf. Figure 21) for selecting among the six *Availability Levels*. In a later iteration of the prototype the dropdown menu was replace with six radio-buttons, as this allowed for quicker selection times. For the location a combo box with autocomplete function was used. It allowed the users to enter a location by typing or by quickly selecting one of their previously typed locations—either by mouse click or by typing the first letters of the location. This way also the quality of the data improved, as it was more likely that the same label and spelling is used for the same location.

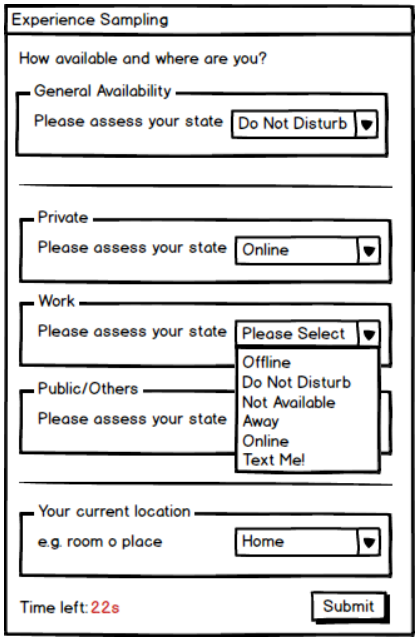

Figure 21. An early prototype of the popup window of the ESM sensor dialogue with dropdown menus instead of the later used radio-button design.

I further abandoned the idea of pre-populating the dialogue with the results from the last sampling. While this would make form-filling more efficient for the participants if the availability did not change between two subsequent samplings, I was afraid of introducing a bias where the users would simply confirm their last values to get rid of the dialogue. On the other side, I refrained

from the idea of shuffling the items each time the dialogue was presented to reduce order effects. In informal pre-test it was found that this considerably irritated and hence slowed down the users. The final design of the dialogue is depicted in Figure 22.

A further aspect was the timing of the sampling. In order to balance the amount of collected samples and the effort—and disturbance—for the participants, I decided for a 30-minute interval for showing the dialogue. To be precise, in order to introduce some fuzziness and further reduce eventual effects, I decided to calculate a random duration between each sampling that lies between 25 and 35 minutes. The dialogue was then presented for 30 seconds as the topmost window, with the remaining seconds counting down in the bottom left corner of the dialogue (cf. Figure 22). When the participant started to fill out the form the countdown stopped. They participant was able to completely fill out the form and dismiss the dialogue by pressing the submit button.

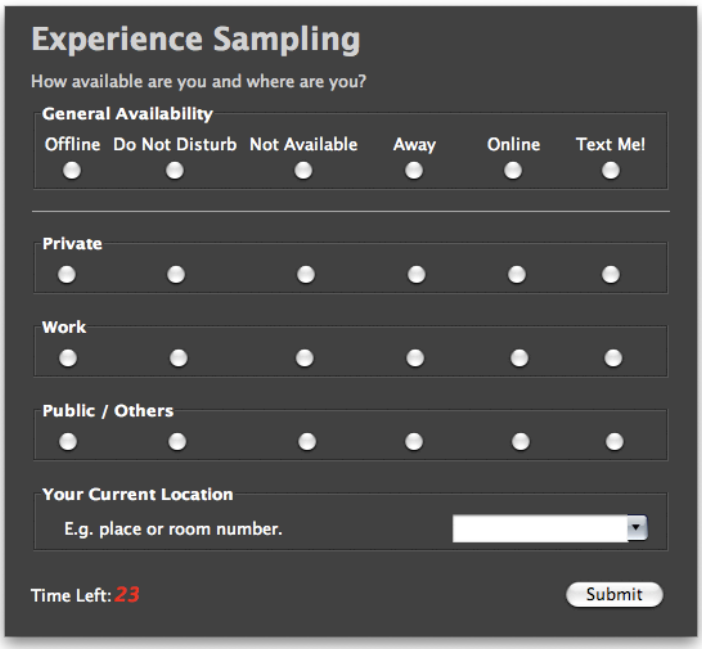

Figure 22. Final design of the dialogue window of the ESM sensor as it was used in the study [Fetter *et al.* 2011b].

If the participant did not start to interact with the dialogue in these 30 seconds, the dialogue disappeared and reappeared again 5 minutes later, until the participant finally fills out the form and the interval is set back to 30 ±5 minutes again. This way I tried to maximise the number of collected samples, even so a participant is often away from the monitor for a short time. I deliberately did not treat unanswered dialogues as a statement of unavailability based on insights of Ho and Intille [2005]. The sampling was started two

minutes after the user logged into the computer, or the computer woke up from sleep-mode.

For the study, the PASS Sensor Daemon was installed on the subjects' laptop computers and configured for the data collection. Besides the ESM dialogue, the collection of data was subtly running in the background, more or less unnoticed by the user. The sensor data collection naturally resulted in some resource constraints, as for example, a reduced battery life, which is discussed later in 4.4. Via a small script, accessible through an icon on the desktop, the participants had the possible to stop and (re-) start the PASS Sensor Daemon at any given time.

3.2.3 Study Execution

For the four weeks long study four male subjects (25-33 years old) were recruited from members of the department. While ESM is a very intense study method, where the effort and disruption for individual subjects is very high over a prolonged time in their normal life, the data this method allows to collect is very rich. While the small number of subjects of course limits the generalisability of the results, the richness of the data partially compensates for that circumstance. The method somehow falls between quantitative approaches and more qualitative approaches like observations.

Based on a small pre-questionnaire, participants that are long-term and frequent IM users were selected. Three of the participants stated that they use IM several times a day, and one participant reported usage on at least 3-4 days a week. The experience varied between 4 to 15 years and each participant handled between 2 to 6 different IM accounts in parallel. Over all theses accounts a participant would have 61 contacts in average (SD=23.4), with 11.2 (SD=2.5) of those contacts being active communication partners, meaning that they communicated at least once in the last three months.

Each subject had the PRIMI Advanced Sensor Suite installed on his laptop computer in a short session before the study started. As most sensors of the PRIMI Advanced Sensor Suite were available and tested for Mac OS X, only participants that primarily used an Apple laptop computer during work hours and leisure time were selected. Other than previous studies, I was interested in more diverse contexts than one fixed setting (like e.g., an office [Avrahami *et al.* 2007]) as I wanted to get insights in the availability preferences of nomadic users. Therefore participants that would use their laptop computers at different locations and in different contexts were selected. While the subjects had means to pause the data collection when necessary (e.g., when giving an important presentation), they were urged to collect as often and constant as possible—at least on four days a week.

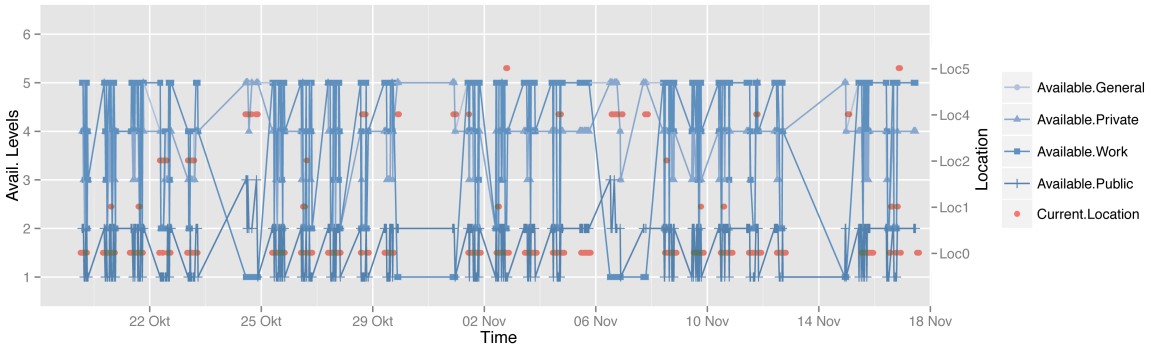

Figure 23. Time series of Availability (1:Offline to 6:Text Me!) and Locations for P1.

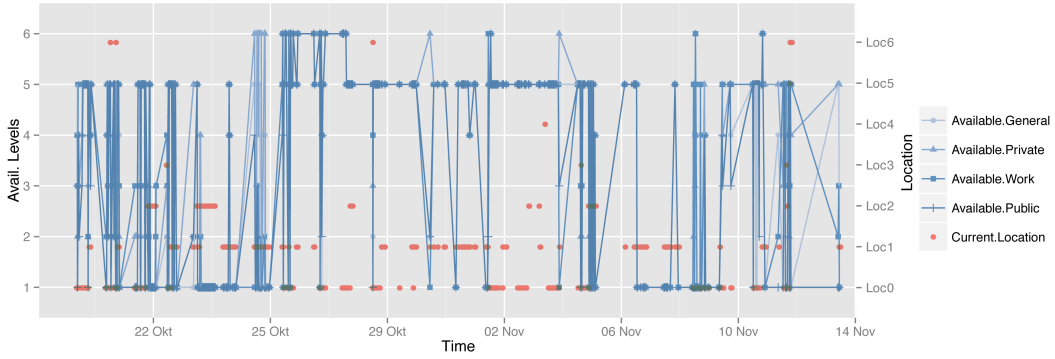

Figure 24. Time series of Availability (1:Offline to 6:Text Me!) and Locations for P2.

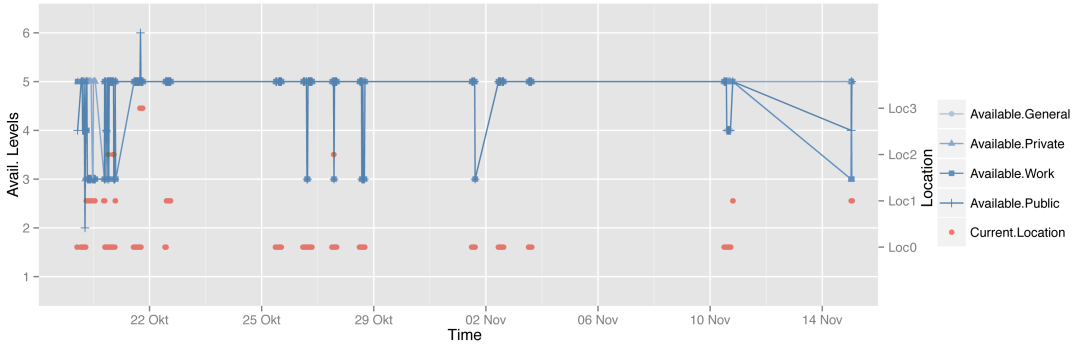

Figure 25. Time series of Availability (1:Offline to 6:Text Me!) and Locations for P3.

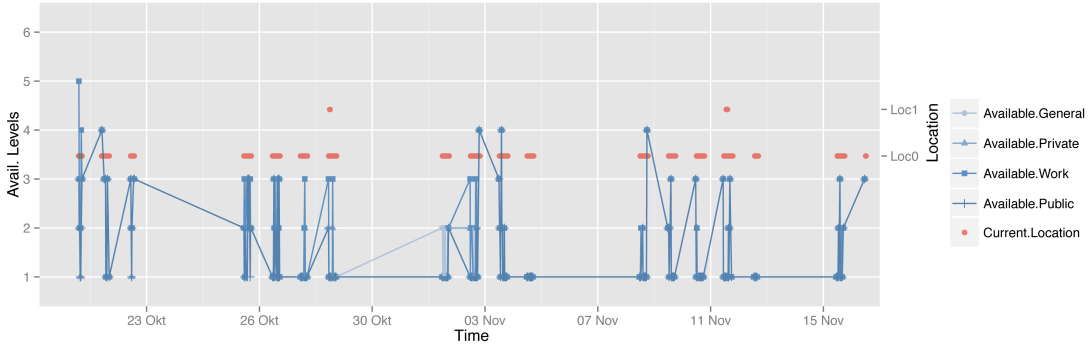

Figure 26. Time series of Availability (1:Offline to 6:Text Me!) and Locations for P4.

For the analysis of *Selective Availability* needs, I focussed on the results of the ESM (i.e., General Availability, Selective Availability and the location) as well as on the result of one sensor—the Skype sensor. As every participant stated to be an active user of Skype, this sensor allowed us to capture data from an actual IM system. The Skype sensor, among other information, constantly collected the online status in an interval of 60 seconds. Further, during the study altogether 29 sensors collected data in the background, on which I report later in 4.1.1.

After the completion of the four weeks the participants arranged a short meeting with the study investigator. In this meeting the data from the computer of the subjects was collected, and each subject was briefly interviewed on the study process. Further the collection process was usually deactivated and the software uninstalled. Appointments for these meeting were fixed individually between the participants and the investigator. Depending on how quickly a convenient date for both parties could be found after the official study period ended, the participants collected a few days more data. Hence, the data I report on in the Following varies between 28 and 31 days in length. Two participants decided to continue with the ESM for a short period after the data was collected from their laptop, as the personally liked the life logging aspect.

3.2.4 Results of the Experience Sampling Study

In total 1353 ESM samples were collected from all participants, resulting in 338.2 samples per person. The relatively large deviation (SD=171.2) was mainly due to the different intensive computer usage over the four-weeks. For example, two of the participants had each two days per week (i.e., the weekend), on which they almost did not use the computer (cf. Figure 28).

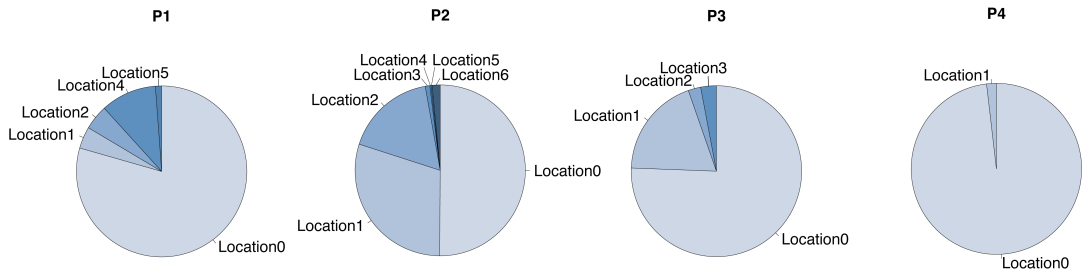

Figure 27. The four charts provide an overview of the locations for P1 to P4. The percentage provides an estimate of the relative duration participants worked on their computers at different locations, based on the number of collected ESM samples at each location. The descriptive names provided by the participants were anonymised during the collection and replaced with numbered labels (e.g., Location2).

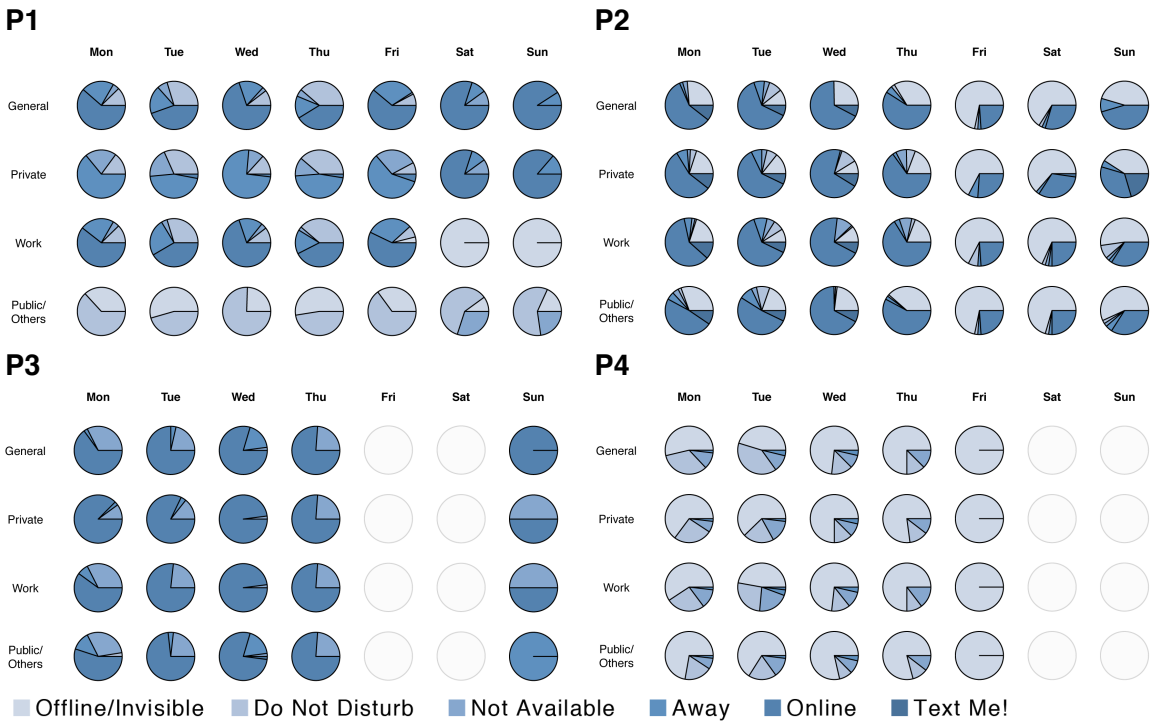

Figure 28. General and Selective Availability data for P1 to P4 for each day of the week (charts for which the number of collected estimates were lower then 10 were omitted).

In Figure 23 to Figure 25 I provide an overview of the individual logging histories, which allows a coarse grained look on the availability samples and the locations. The participants collected samples from a variety of different locations, including their own offices and private living rooms, other people's offices, as well as meeting and lecture rooms. In average the computer was used at 4.8 (SD=2.2) locations by the participants, whereby in average 76% of the samples were collected at one dominant location: their office (cf. Figure 27).

The interviews allowed an insight on the work and private activities during which they collected samples. Among these activities were programming, editing documents, meetings and lectures as well as watching movies or browsing the web at home.

For P3 and P4 we can see very different global tendencies towards being either more available (P3) or less available (P4). P2 on the other side almost exclusively uses the two ALs *Offline/Invisible* and *Online*, and largely omits the rest of the ACs, which is clearly visible in Figure 29. These results already hint that very personal perspectives on availability exist—this is for example supported by a study by Ruppenthal and Chignell [Ruppenthal & Chignell 2002], in which they identified *interruptibility* as one of 19 different communication traits for users of a unified communication service.

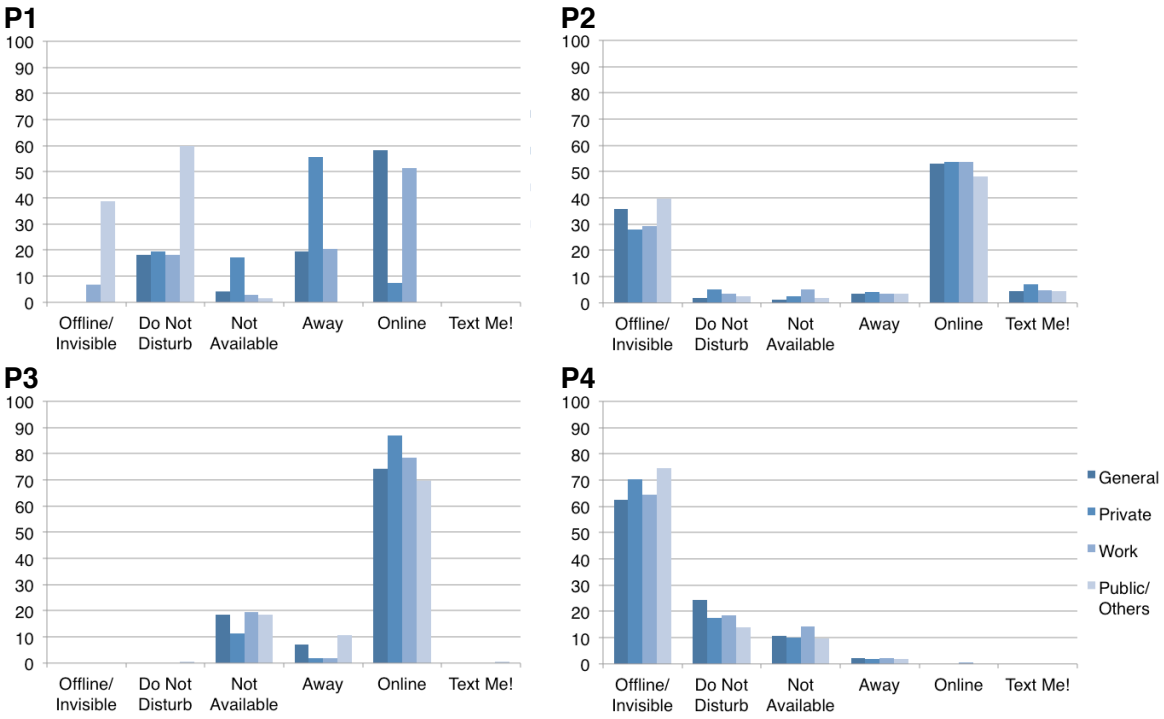

Figure 29. Frequency of occurrences for availability estimates for participant 1 to 4.

In order to get more quantifiable measures on the subjects' different needs for *Selective Availability*, I counted the number of ESM samples were the ALs over the three ACs diverged. That is, I counted the samples were a different AL was chosen for at least one of the three ACs (e.g., Private=Available; Work=Available; Public/Others=Do Not Disturb).

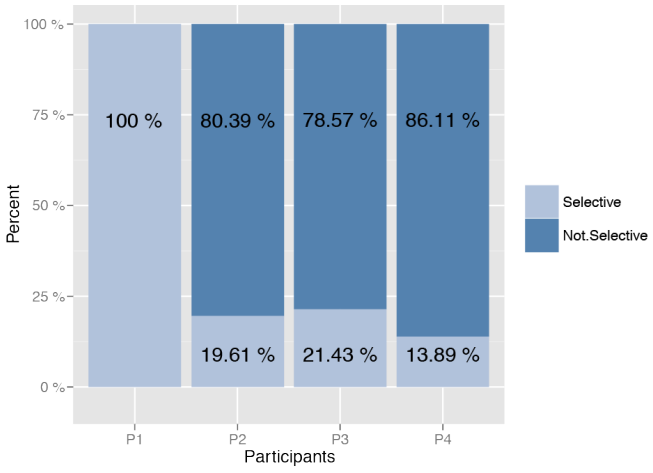

Figure 30. Overall expression of a need for Selective Availability for P1 to P4 based on the ESM data.

I found that while one subject (P1) in 100% of the samples expressed a need for *Selective Availability*, the other three did not use the option to express different ALs for different ACs so exhaustive. However I found for the other

three that 18.3% (SD=3.9) of the samples in average differed, and expressed a need for *Selective Availability* (cf. Figure 30). When analysing the need for *Selective Availability* per location, it revealed that the need for *Selective Availability* also varies based on the current location (cf. Figure 31).

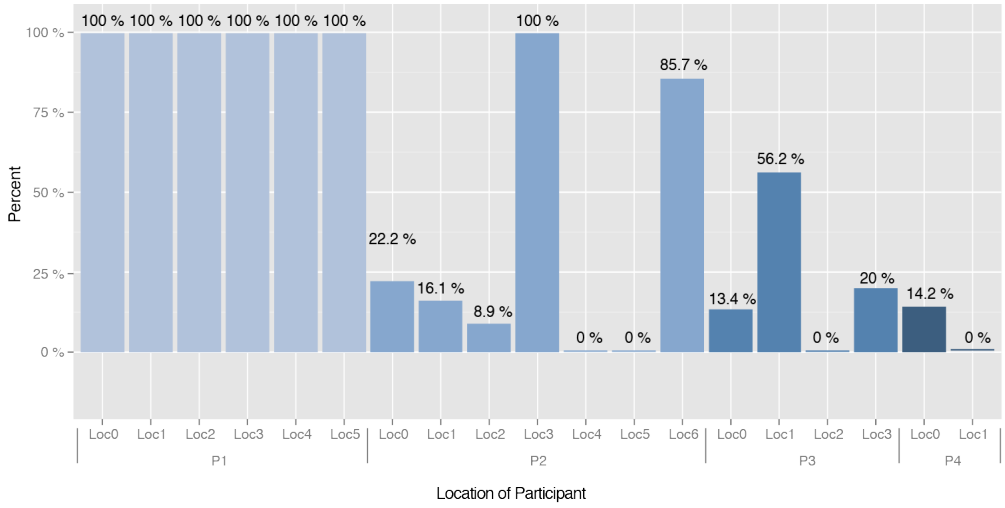

Figure 31. Per location expression of a need for Selective Availability for P1 to P4 based on the ESM data.

In the interviews the participants affirmed many of the findings. Interestingly, all of the three participants who only expressed *Selective Availability* for roughly one-fifth of the times were surprised about this low percentage and stated they would have estimated their need for *Selective Availability* higher. One subject, for example, told the interviewer that "in most situations I [he] treated work and private equally [...] but for public I [he] mostly chose at least one AL lower". When reflecting on this aspect of a low percentage of *Selective Availability*, the subjects also stressed that even though they did not express *Selective Availability* often, when they did, it was quite important to them to make this differentiation. One participant added, that the ability to choose an own set of ACs would allow for more fine-grained expression of availability and would have probably increased the usage of *Selective Availability* for him.

From the numbers I further identified that in 97.6% (SD=2.6) cases at least one AC was set to the same availability level as the General Availability. Finally, I compared the General Availability to that of the Skype sensor, as all participants used this IM platform for communication with contacts from different ACs. I found that the Skype status at the time of a sampling only matched the General Availability 41.5% (SD=16.8) of the times. In general the Skype status was changed only between 6.9% (SD=3.8) of the total samples, while the *General Availability* almost changed between every third sampling (31.6% (SD=11.4)). A finding that was confirmed in the interviews, where subjects told that they hardly update their IM status. One subject commented

that the few occasions when he was changing the status were mostly triggered by an incoming message during a meeting. The person added, that most times he would also forget about the *Do Not Disturb* status directly after the meeting and only change the status back when he realises that no messages are coming in after some time. A second participant stated that the ESM study changed his awareness of his online status and he started adapting the Skype status more often towards the end of the study.

3.2.5 Conclusions

The findings in many parts are affirmative with results of other studies on IM usage and availability as discussed in Chapter 2.

Particularly from this study I was able to confirm that users have a need to express their availability selectively—also this might not be the case 100% of the time. But, there are situations where all subjects valued such an option, as it was either confirmed in the interviews or clearly visible from the data. Second, from the interviews and the data I also can conclude that continuously adapting the IM status manually is too tedious and therefore tends to be neglected. Accordingly the presented online status and the actual availability rarely match.

In total, the collected data helped to form my conception of the requirements for *Selective Availability* of nomadic users, which use their computers in various settings and want to announce intricate availability to a variety of contacts.

3.3 Discussion

In this chapter I enhanced the understanding of user needs when managing their availability for computer-mediated communication. For once, I presented concepts that support users in setting up complex selective information disclosure configurations. Concepts like *Disclosure Templates* and *Availability Categories* can contribute to a reduction of the overall specification effort. However, both concepts are not taking into account the dynamic nature of privacy. From the study we can conclude that current solutions that require the continuous manual management of complex availability configurations quickly can become too tedious and accordingly will not be kept up-to-date. However, the currentness of an online state is uttermost essential to its utility—once out-of-date it becomes meaningless. In order to support users adequately the system has to take account for the selectivity and the dynamics of privacy preferences.
In order to achieve this, I investigated to what extent it is possible to automatically keep the availability of a user updated in a system. In the Following I report on the two steps that were performed to investigate the automatic adaptation: First, I analysed if a user's privacy preferences are inferable from the situation, the context, and actions of a user, which would be a

premise for supporting users keeping their privacy preferences automatically up-to-date. In a second step, I came up with a concept for a system that would continuously learn a user's privacy preferences and automatically update all respective settings.

4 Towards the Automatic Adaptation of Selective Availability

In chapter 2 I showed that a trade-off exists between the benefits and the drawbacks of constantly being available for communication. I also discussed how people differ in their openness for social interactions in respect to the type of relationship to the recipient and the current situation. One illustrative—when also relative narrow and simplified—example is the distinction between such contacts that are related to the private life and such, that are related to work life, and situations that can be seen as work-related and those related to private life. But we also learned that privacy is not only a selective but also a dynamic construct. To stay in the example, the amount in which we are receptive to communication to members of one of these two categories varies over the week, the day, the hour, the minute.

In chapter 3 we saw how new concepts can optimise the balance between the effort for configuring the own information disclosure and the granularity of the disclosure settings. For example, instead of configuring the disclosure for each contact individually, categories like team or family can help to reduce the effort. However, we also learned from the ESM results that privacy requirements not only are selective, but also are dynamic. Constantly adapting the settings according to changing privacy needs for all this categories still overburdens the users. This is especially true in times of nomadic computer usage, where mobile phones, computer-mediated communication and laptop computers give us the freedom of working from everywhere. People not only work from their offices, but also from their homes, coffee shops, hotel rooms, airports, etc. They take these technologies and use it on a train while commuting, in a meeting while listening to a presentation, or in a lecture room while following a university course. These ever-changing circumstances lead to constantly changing needs and preferences for interaction with and isolation from different persons. Thus, tools are required that allow us to dynamically communicate and announce our availability for communication to others. Not only is our availability constantly changing, also the information that is currently disclosed is constantly changing, and accordingly our disposition to the disclosure of this information. In order to achieve optimal privacy, the announced availability status as well as the configuration regarding the disclosed information would need to be adapted continuously to reflect the current situation.

Currently, in conventional IM solutions the online state has to be set manually by the user. In order to adapt the online state, the users need to interrupt their primary tasks, leading to interruption and resumption lags [Trafton *et al.* 2003]. Further, as the transitions between different phases of work

are often fluent and subtle, the users often forget adapting the online state [Harr & Wiberg 2008]. Milewski and Smith [2000] found in their study of a mobile personal presence system that users only changed their status 1.4 times a day in average. I made comparable findings for the Skype status as reported earlier. Accordingly, the announced online states are often incorrect, and thus loose their significance for the contacts of the users; they are considered unreliable and are accordingly ignored. Voida *et al.* [2002] observed this behaviour in a study and found people are textually enquiring via IM if a contact is available, despite the online state telling them otherwise. Consequently, while *Selective Availability* and selective disclosure of context information provide people with more nuanced controls, and thus allow them to adequately announce their openness for communication, keeping several online states manually up-to-date is not feasible for users that work in a variety of different contexts—such as nomadic users. To come up with a solution, I suggest a concept that allows the automatic adaptation (compare *automated calculation* [Harr & Wiberg 2008] in 2.1.3) of the availability of a user tailored towards different audiences.

Hence, in the Following I present the results of my analysis, whether the previously collected availability estimates from the study are predictable from the logged sensor data. Based on my results, I conclude that an automatic calculation of *Selective Availability* purely from sensor data is possible, even in the more complex domain of nomadic users. I therefore introduce the concept of and requirements for context-aware and adaptive systems, which would allow automatically maintaining up-to-date user statuses towards different audiences with a minimised configuration effort. I further show, in a short digression, how some of my previous work on online learning personalised models informed this effort. I close the chapter with an overview of requirements and design decisions for a sensing infrastructure, which lays the foundation for a context-aware adaptive system.

4.1 Analysing the Predictability of Selective Availability

Accordingly, the first step was to explore whether *Selective Availability* of nomadic users [Fetter *et al.* 2011b] is predictable from sensor data. While previous work (e.g., [Begole *et al.* 2004; Fogarty *et al.* 2004a; Horvitz *et al.* 2002]) has shown that a prediction of availability (or interruptibility) is generally feasible in fixed settings like an office room—a controlled environment, with well-known parameters and a constricted number of simple sensors—my aim was to further exploit these approaches, to see what results can be achieved in less controlled environments. I therefore analysed the data collected in our study (cf. 3.2)—to gain an understanding of the general feasibility of predicting *Selective Availability*—in an explorative [Patel *et al.* 2008] machine learning

approach from data of real nomadic users collected in the wild. That is, I analysed how well I am able to predict the availability of a nomadic user in form of one of the six *Availability Levels* for a specific *Availability Category* based on the collected sensor data from a study participant. Thereby each *Availability Category* is treated as a different classification problem [Alpaydin 2010a], cumulating in four classification tasks for each participant—*General*, *Private*, *Work*, and *Public/Others*. In machine learning classification is a supervised learning problem [Alpaydin 2010b]. The goal is to identify the one class from a given set of classes to which a new observation belongs, based on the provision of training examples for which the class membership is known and indicated by a label. In the case of this work the classification problem is to identify for a number of sensor readings that describe the current situation (i.e., an observation), which of the six *Availability Levels (*i.e., the classes*)* is currently the most adequate for a certain *Availability Category* (i.e., the classification problem).

The process involved the step of manual engineering features based on the study sensor data, as well as the semi-automated selection of algorithms and parameters, and their evaluation, as explained in the Following.

4.1.1 Data from Experience Sampling Study

As previously mentioned in the description of the study, besides the participant's availability estimates, a wide variety of sensor data was collected in the study on *Selective Availability*. While the general study setup is already discussed in great in details section 3.2, the focus here is on briefly summarising the nature of the collected data from a machine learning perspective.

As class labels for the classification task I used the availability estimates given by the study participants via the Experience Sampling dialogue in form of the six *Availability Levels*. While in average the dialogue was presented 817.75 (SD=500.52) times to a subject, it was only answered roughly 4 out of ten times (cf. Figure 32), resulting in 355 (SD=167.00) availability estimates per participant in average. That is, in average 355 possible instances of test and training data for each of the four *Availability Categories*.

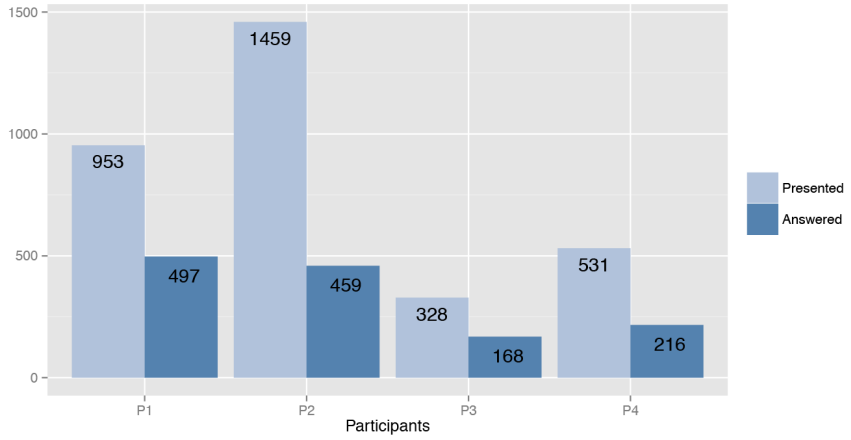

Figure 32. Number of presented versus answered ESM dialogues for each participant.

Here it should be noted, that the fact, that more than half of the ESMs were not answered by the participants was expected and part of the design of the study. As previously described, every time a dialogue was not answered, the interval for sampling was reduced from 30 minutes to 5 minutes. This was done to avoid longer gaps in ESM feedback, when a computer was left unattended for a short time e.g., when a participant left for the bathroom. For longer periods (e.g., a lunch break), this of course led to a high number of dialogues that remained unreturned. A one-hour break for example now led to twelve unanswered dialogues, versus two with the fixed interval of 30 minutes. As this effect was introduced intentionally, the high number of unanswered dialogues can be neglected. Still it should be mentioned, that I also intentionally decided to not treat dialogues that remained unanswered as an evidence for unavailability. While at first glance this idea seems appealing to increase the number of labelled data, it is not clear whether the participant was really not available or just unreceptive for the dialogue [Ho & Intille 2005].

As basis for the classification task I use the sensor data[1], which was collected in the background on the participant's laptop computers during the whole study. The 29 sensors collected 270,726 (SD=133,612) sensor events in average per participant. The sampling rate, in which the sensors delivered data, was configured individually per sensor, and ranged between 30 and 150 seconds. Some of the considerations for choosing the sample rates are discussed in section 4.4. Generally I tried to balance the sample rate to maximise the richness of the data while minimising negative effects like for example battery and CPU usage.

---

1 In this work, sensor is defined as "a soft- or hardware component, or its combination that monitors and records the state of an artefact, an entity or the environment in the digital or physical realm" [Fetter *et al*. 2011b, p. 505] as elaborated on in 4.4.

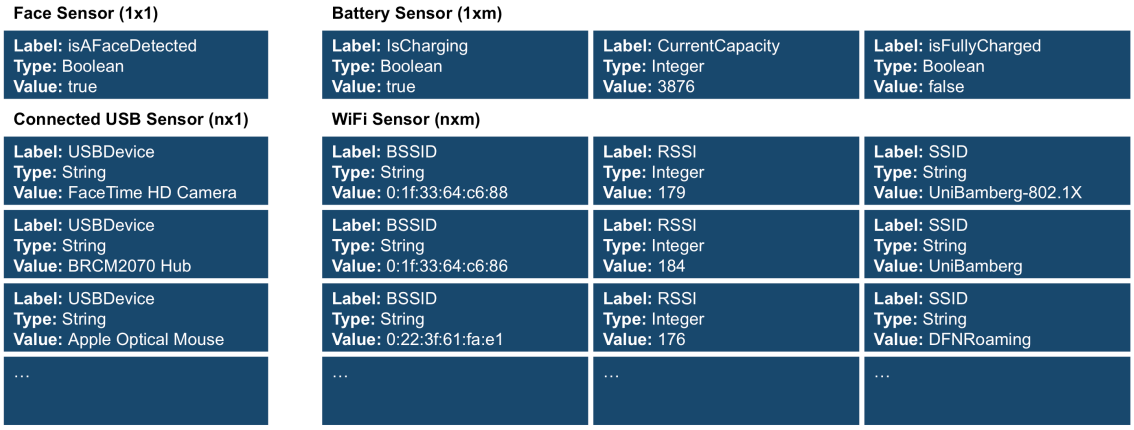

Figure 33. Visualisation of sensor readings of a 1×1-, 1×m-, n×1-, and n×m-sensor.

Depending of the individual sensor's implementation and the measured constructs, the sensors deliver data of different complexity and in different structures. From this data-structure perspective we can distinguish between four types of sensors: 1×1 sensors (one single key/value-pair per sensor reading; cf. *Face Sensor* in Figure 33), 1×m sensors (*m* semantically distinct key/value-pairs of potentially differently typed values; *m* is fixed for each sensor; cf. *Battery Sensor* in Figure 33), n×1 sensors (a list of *n* semantically associated key/value-pairs of the same type; *n* is determined by the individual sensor reading; cf. *Connected USB Sensor* in Figure 33), and n×m sensors (a table of values with *m* columns and *n* rows; where *m* is fixed and *n* is determined by the individual sensor reading; cf. WiFi Sensor in Figure 33). More details on sensors are provided later in sections 4.4 and 5.3.2.

The 29+1 sensors I used in the study were the following:

- Active Access Point (1×1): BSSID of current WiFi access point
- Active Chats (1×1): no. of open chats
- Active Network Interfaces (n×1): list of interface IDs
- Ambient Light (1×1): level of ambient light
- Applications (n×1): list of all running applications
- Application Focus (1×1): application in focus
- Battery (1×m): battery capacity; charging state; fully charged
- Bluetooth Devices (n×1): nearby Bluetooth devices
- Calendar (1×m): upcoming and on-going events
- Connected FireWire Devices (n×1): list of devices
- Connected USB Devices (n×1): list of devices
- CPU (1×m): user and system CPU load; CPU idleness
- Email (1×m): no. of unread, received, and sent email
- Ethernet Connected (1×1): whether Ethernet cable is plugged in
- Face Detection (1×1): whether a face is detected in the laptop web cam
- Focus Title (1×1): title of window in focus

- Headphones Connected (1×1): whether headphone cable is plugged in
- Input Idle (1×1): whether keyboard and mouse are idle for a specific period
- IP Address (1×m): external and local IP address
- Mounted Volumes (n×1): list of volumes
- Motion (1×m): acceleration of X-, Y-, and Z-axis
- Mouse Connected (1×1): whether a mouse is plugged in
- Power Connected (1×1): whether the power cable is plugged in
- Screensaver Active (1×1): whether the screensaver is active
- Second Monitor (1×1): no. of connected displays
- Skype (1×m): no. of online contacts; mood message; status; whether a call is on-going
- Time (1×m)[1]: hour of the day; day of the week; weekend; part of day
- Voice Activity (1×1): whether a voice is detected via the laptop's microphone
- Volume Settings (1×m): output and alert volume; whether volume is muted
- WiFi (n×m): nearby WiFi access points' SSID; BSSID; and RSSI

The aim of the different sensors was to capture hidden evidence on different aspects of the users' current situation, which would allow making a prediction on the users' availability. Among these aspects were: the location of the user, social interaction, current tasks, etc. While it is obvious that nearby WiFi networks can help with a good approximation for the location or knowing which application is in focus can provide some ideas on the current task, also the other sensors allow making valid estimations on the current situation. For example, whether a laptop is connected to a mouse, Ethernet, power or a second displays also allows drawing some conclusion on the location or the task of the user.

However, in order to be able to make predictions based on this data, I needed to bring together the sensor data and the labels in a meaningful way, as explained in the Following.

4.1.2 Feature Engineering—From Raw Sensor Data to Meaningful Features

The sensors deliver data in a very raw—and in combination also unstructured—format. Yet, machine learning algorithms perform better on data with a very strict structure. Therefore the data has to be pre-processed accordingly. In machine learning each *observation* (also *instance* or *example* [Witten *et al.* 2011b]) is represented in form of a *feature vector*—an n-dimensional vector of predetermined attributes (i.e., the features) and their respective values. The

1 The data for the Time-Sensor was synthesised from the time stamps of other sensors after the data collection.

features thereby are mostly simple data types, often only numeric[1]. The sensor data on the other side can become quite complex, and varies from sensor to sensor. The *Connected USB Device-Sensor* (1×n) for example delivers a list with variable-length of all connected devices' names with each sensor reading. In order to be processable for a machine learning algorithm, the sensor's data needs to be converted into a simpler representation. In the case of the *Connected USB Device-Sensor* each device name from the list is extracted in an own binary feature. This process is called *feature extraction*, and is the first *feature engineering* step [Liu & Motoda 1998]. The idea of feature extraction is to transform arbitrary, often unstructured data—as for example text documents or image data—into numerical features to make it processable for a machine learning algorithm. To give simple examples, this can happen by counting the occurrence of specific words in a text document or by calculating the percentage of colours present in a specific image. However, often more complex approaches like Principal Component Analysis or Singular Value Decomposition, or specialised approaches like e.g., Term Frequency–Inverse Document Frequency (for text documents) or Mel Frequency Cepstral Coefficents (for audio) or Hough transform (for images) [Guyon *et al.* 2006] are used.

In the second step, *feature construction*, the aim is to boost the prediction performance by using domain knowledge to construct higher-level information from the already available features. To give a very simple example, imagine a feature vector containing a *birthdate* and a *deathdate*. By applying domain knowledge, we can easily construct a new feature called *age*. Accordingly, we often can apply simple operators to existing features to construct new features. This starts with simple mechanisms like checking equality (=, ≠), applying arithmetic operations (+,–,×, /), calculating basic statistical values (average(), max(), min(), median(), standardDeviation()), to more complex domain-specific operations.

Finally, in the last step called *feature (subset) selection* (also *variable* or *attribute subset selection*), the aim is to reduce the dimensionality of the dataset by excluding features present in the data that have limited predictive power (i.e., unnecessary, irrelevant, noisy, or redundant features) [Guyon & Elisseeff 2003]. Basically, three different approaches exist: filter, wrapper, and embedded methods. Filter methods calculate a statistical score of each feature (e.g., information gain [Roobaert *et al.* 2006]) that is used to rank and then select or remove the respective features. The computational more expensive wrapper methods evaluate different combinations of features by comparing the

---

1 For example, the Weka Toolkit [Hall *et al.* 2009] supports besides numeric attributes further nominal, string, and date as data types for attributes—in most cases Weka however uses simple distance measures or converts them in numeric binary indicator attributes.

accuracies of the resulting predictive models in form of a search process. Finally embedded methods, try to combine both approaches, by learning which of the features contribute the most to the predictive model, while the model is created [Navin Lal *et al.* 2006]. While from a theoretical view, the selection of the optimal feature subset for a supervised learning problem requires an exhaustive search on all possible combinations of features, from a practical perspective the search should be limited to finding a satisfactory set of features based on pre-set criteria and conditions.

In the Following I show the manual and semi-automatic feature engineering steps I undertook to transform the sensor data logs into valuable feature vectors. On a more practical level, this also meant to transform the collected XML data of each participant into a representation according to Weka's ARFF (attribute-relation file format) [Witten *et al.* 2011a, p. 51ff]. This was done by reading all sensor events sequentially from all XML files of one participant and replicating them into a database. Starting from this database the steps feature extraction and feature construction were sequentially performed separately for each participant. Furthermore, the feature selection was done separately for each learning problem, which is the prediction of the *General Availability* and the three types of *Selective Availability*. The result of the feature engineering process is stored in four ARFF-files—one for each of the four participants and each including all labelled instances for the respective participant. Although the sensor data was collected continuously, labels were only available at specific times. Hence, instances were only generated for the parts of the sensor logs where labels were available.

4.1.2.1 Extracting Features from Sensor Data

As a first step, it was necessary to convert all participants' raw sensor data logs into features that are processable by Weka.

For most 1×1 sensors, the process is quite straightforward. 1×1 sensors that deliver Boolean values (e.g., *Headphones Connected* or *Input Idle*) and sensors that deliver numeric values (e.g., *Active Chats*), are simply converted into respective feature representations. For 1×1 sensors that deliver String values (e.g., *Application Focus* or *Active Access Point*) one additional step is needed to represent them as a nominal feature. As for the specification of a nominal in an ARFF file all the possible values need to be listed, the pre-processing mechanism needs to maintain a list of all possible values (e.g., a list of all applications that had been in focus).

```
<sensVal sid="Volume" date="2009-10-19 14:52:36">
      <row>
            <item label="output-vol" type="integer">25</item>
            <item label="input-vol" type="integer">50</item>
            <item label="alert-vol" type="integer">100</item>
            <item label="muted" type="boolean">false</item>
      </row>
</sensVal>
```

Listing 1. Exemplary sensor value of the 1×m sensor Volume.

For 1×m sensors the approach is only slightly different, as each 1×m sensor delivers not only one but a fixed number (m) of values, each of the m values is handled separately. For example, the *Volume Settings* sensor delivers 4 different values, three Integer values (i.e., *output-vol*, *input-vol*, and *alert-vol*) and one Boolean value (i.e., *muted*) as can bee seen in Listing 1.

```
@RELATION volume-demo
@ATTRIBUTE Volume.output-vol NUMERIC
@ATTRIBUTE Volume.input-vol NUMERIC
@ATTRIBUTE Volume.alert-vol NUMERIC
@ATTRIBUTE Volume.muted {'true','false'}
...

@DATA
25 , 50, 100 , false ,...
```

Listing 2. The corresponding ARFF snippet, showing the results of the feature extraction from 1×m sensor data.

As we can observe in Listing 2 the approach is basically similar to that for 1×1 sensors, just with each item in a 1×m treated as one separate feature. Accordingly the resulting ARFF representation holds four features (i.e., `@ATTRIBUTE`): three of the type numeric and one nominal binary attribute. The name for each feature is concatenated from the name of the sensor, and the item label (for example `Volume.output-vol`).

Other than the 1×m, the n×1 sensor can hold an arbitrary number of items, as the sensor basically represents an array of data. When looking at the two sensor readings in Listing 3 of the *Applications* sensor—which lists all current running applications—we see the first sensor value (*SensVal*) contains three rows, with each item referring to one running application (i.e., the OS X applications Finder, Word, and Terminal) and the second *SensVal* contains four rows representing four running apps (i.e., Finder, Excel, Safari, and Terminal).

```
<!-- Sensor Value 1 -->
<sensVal sid="Applications" date="2009-10-05 09:22:14">
 <row><item label="AppName" type="string">Finder</item></row>
 <row><item label="AppName" type="string">Word</item></row>
 <row><item label="AppName" type="string">Terminal</item></row>
</sensVal>

<!-- Sensor Value 2 -->
<sensVal sid="Applications" date="2009-10-06 11:15:18">
 <row><item label="AppName" type="string">Finder</item></row>
 <row><item label="AppName" type="string">Excel</item></row>
 <row><item label="AppName" type="string">Safari</item></row>
 <row><item label="AppName" type="string">Terminal</item></row>
</sensVal>
```

Listing 3. Two exemplary sensor values from the n×1 sensor Applications

As a representation of this data in form of a single feature would be quite complex, I needed a representation that can handle this kind of data. What is observable in above case is that neither of the sensor values contains a complete list of all applications (i.e., Finder, Word, Excel, Safari, and Terminal). Hence, it was necessary in a first step to read in all possible occurrences of application names. From the complete list of possible applications, I created one Boolean feature for each application (e.g., one feature named `Applications.AppName.Excel` and one feature named `Applications.AppName.Word`).

Now it is possible to compare each of these features with a specific *SensVal*, and check if the respective application is present or absent in this list. And evaluation of the first *SensVal* results in (Finder=true, Word=true, Terminal=true, Excel=false, Safari =false) and of the second *SensVal* in (Finder=true, Word=false, Terminal= true, Excel= true, Safari = true), and accordingly result in an ARFF representation like the one shown in Listing 4.

```
@RELATION applications-demo
@ATTRIBUTE Applications.AppName.Finder {'true','false'}
@ATTRIBUTE Applications.AppName.Word {'true', 'false'}
@ATTRIBUTE Applications.AppName.Terminal {'true', 'false'}
@ATTRIBUTE Applications.AppName.Excel {'true', 'false'}
@ATTRIBUTE Applications.AppName.Safari {'true', 'false'}
...

@DATA
'true', 'true', 'true', 'false', 'false',...
'true', 'false', 'true', 'true', 'true', ...
...
```

Listing 4. The corresponding ARFF snippet, showing the results of the feature extraction from n×1 sensor data.

Finally, n×m-sensors come with the highest complexity. The *SensVal* of a n×m-sensor basically is a table-like structure of columns and rows. Depending on the sensor it comes with a fixed number of *m* columns but can be of an arbitrary length *n*. Although in our data only one n×m-sensor, the *WiFi* sensor was present, I still aimed at a generic approach. The *WiFi* sensor returns a table of wireless access points in proximity, with each access point being described by the items SSID and BSSID as identifiers, as well as by the signal strength RSSI (see 0 for an example sensor value). For this specific sensor, basically a variety of features can be calculated. For once, the SSID and BSSID could be processed in a similar approach like the n×1 sensor. That is, generating a list of all networks (SSIDs)—or respectively the access points (BSSIDs)—in form of Boolean features and comparing the sensor values for absence or presence of said SSIDs (or BSSIDs). While the pure absence or presence of a specific network (e.g., something like *Uni-Bamberg802.1X*) can give an estimate if the person is in a specific environment, the absence or presence of a BSSID (e.g., the Access Point *0:1f:33:64:c6:88*) can even give a finder grained estimate if a person is in a specific room. The same process of course does not make sense for the numeric signal strength RSSI. But, obviously combinations of the RSSI with the SSID (respectively BSSID) would allow for even more fine-grained location estimates. Therefore logically the permutations of multiple items into one feature, like for example the signal strength of a specific SSID or BSSID (e.g., `WiFi.SSID.Uni-Bamberg802.1X.RSSI`, `WiFi.BSSID.0:1f:33:64:c6:88.RSSI`,...) are valuable additions to the feature vector (cf. Listing 5). Again, we have the exception that not all permutations make sense. For example, a feature that permutates the item SSID and BSSID would be a constant, as it would relate two static IDs to each other. However—in order to come to a generic solution that does not presuppose too much knowledge of the data—I decided to first extract all possible permutations and later eliminate those with limited predictive power, based on their entropy, in the final feature-engineering step.

```
@RELATION wifi-demo
@ATTRIBUTE WiFi.SSID.Uni-Bamberg802.1X {'true', 'false'}
@ATTRIBUTE WiFi.SSID.Uni-Bamberg802.1X.RSSI numeric
@ATTRIBUTE WiFi.BSSID.0:1f:33:64:c6:88 {'true', 'false'}
@ATTRIBUTE WiFi.BSSID.0:1f:33:64:c6:88.RSSI numeric
...

@DATA
'true', 188, 'true', 179,...
'true', 176, 'false', ?,...
...
```

Listing 5. An ARFF snippet, showing exemplary the results of the feature extraction from the n×m-WiFi-sensor.

Lastly, I also extracted further features by analysing the meta-data of the sensor values. Firstly, I inferred several features from the date and time of the sensor values. While using the plain date and time hardly makes sense, time in general is important information. For each sensor value the following four additional nominal features were calculated:

- `HourOfDay`, with the possible values {0,1,2,3,4,...,23},
- `PartOfDay`, with the possible values {morning, noon, afternoon, evening, night},
- `DayOfWeek`, with the possible values {sun, mon, tue, wen, thu, fri, sat}, and
- `Weekend`, with the possible values {true, false}.

Secondly, I generated a feature in respect to the length n of the sensor value of an n×1- and n×m-sensors. For example, a high or low amount of WiFi networks in proximity (e.g., `WiFi.ListLength`) can provide some further insight on the environment by allowing conclusions from the pure density of available access points.

4.1.2.2 Constructing Features Taking Time into Account

After extracting all features, the next step was to construct features that would additionally improve the prediction quality. As each sensor reading just reflects a very brief snap shot in time, my main consideration in this feature engineering step was to develop features, that better take into account the history of a sensor value and respectively trends and tendencies of the users context over time[1].

When taking into account sensor values over time, we are basically in the domain of pattern recognition from time series. As working with time series usually comprises working with very high-dimensional data, a first step is to find a higher-level representation of the data by inferring significant characteristics that provide a valid approximation of the data.

In order to accomplish this, I constructed such characteristic figures [Fogarty & Hudson 2007; Markovitch & Rosenstein 2002] for Numeric features by calculating mean (`mean`) and standard deviation (`stdDev`) values for three intervals: the last five (`5`), eight (`8`), and fourteen (`14`) minutes before the labels timestamp. For Nominal features, I considered how often the value changed

[1] As a quick side note: It is important here to understand, that while I continuously collected sensor data in the background, I only have labels for this data for roughly every 30 minutes. As there is no way, to extrapolate the label data, at first sight, it would have made sense to only extract features for these few brief moments in time. However the sensor reading of just the very moment when we receive a label, only provides a poor and rather static impression of the users' context. Therefore my aim was to take into account the sensor data of the last few minutes. In order to achieve this, the data was continuously collected in the background, to construct these history-based features.

(chnge) in those intervals. For Boolean features, I inferred the time slice for which the particular feature was *true* or *false* during the respective period. That is, for each feature and each of the three intervals I calculated the six Boolean characteristics: all the time true (allT), none of the time true (noneT), and at least once true (onceT), at least one quarter of the time true (oqT), at least half of the time true (hlfT), at least three quarters of the time true (tqT). Further I calculated the percentage of time the feature was true as an additional numeric feature (ctoT). All features received a generated name by combining the original name of the feature (e.g., InputIdle) with a suffix for the constructed feature type (e.g., allT) and a suffix for the number of minutes (e.g., 5) as can be seen in Listing 6. Of course this step further increased the dimension of the feature space enormously. Accordingly the next step is to reduce the dimensionality.

```
@RELATION construction-demo
@attribute Motion.Y.mean.5 numeric
@attribute Motion.Y.mean.8 numeric
@attribute Motion.Y.mean.14 numeric
@attribute Motion.Y.stdDev.5 numeric
@attribute Motion.Y.stdDev.8 numeric
...
@attribute IP.externalIP.chnge.5 numeric
...
@attribute InputIdle.allT.5 {'true', 'false'}
...
@attribute InputIdle.hlfT.5 {'true', 'false'}
...
@attribute InputIdle.onceT.5 {'true', 'false'}
...
@attribute InputIdle.oqT.5 {'true', 'false'}
...
@attribute InputIdle.tqT.5 {'true', 'false'}
...
@attribute InputIdle.ctoT.5 numeric
...

@DATA
...
```

Listing 6. An ARFF snippet, showing exemplary the results of the feature construction mechanisms.

4.1.2.3 Selecting the Best Feature Subset

In the last feature-engineering step, the aim was to reduce the high number of features by selecting only those that can contribute the most to the predictive model. In a first step, I removed all extracted and constructed features, which have no variance (i.e., that are constant) or too much variance in the data, by relying on the *RemoveUseless*-filter of the Weka toolkit. This filter removes all

constant features (i.e., features with 0% variance) and all features that exceed the maximum percentage of a given variance parameter, in this case variance of 95% or more. This in average removed 13% of the features for the four datasets of the four participants. The resulting number of features after this reduction still was 1,437.3 in average.

The process of feature selection was done in the Weka Experimenter as proposed here [Hall & Holmes 2003] and following insights from Fogarty et al. [2007; 2004a] on feature engineering for modelling human interruptibility. To find an optimal subset, I performed experiments with five of Weka's evaluation methods: three single-feature evaluation methods and two feature-subset evaluation methods (compare [Witten *et al.* 2011a, p. 489]:

- *InfoGainAttributeEval:* This evaluation method uses the Information Gain to rank features. The Information Gain [Quinlan 1993] in machine learning quantifies the mutual information. It is a measure for the expected reduction in entropy (H) for a class if a feature (a) is added based on a training Set (S).

$$IG(S,a) = H(S) - H(S|a)$$

- *GainRatioAttributeEval:* The *GainRatioAttributeEval* method calculates the Information Gain Ratio to rank features. The Information Gain Ratio tries to compensate the Information Gain's tendency of favouring features with many different values. Therefore the approach also takes into account the number of distinct values a feature can have (i.e., *the* variance*)* [Quinlan 1993].
- ChiSquaredAttributeEval: Also *ChiSquaredAttributeEval* works on a single feature—this method analyses the value of a feature by calculating its chi-squared statistics with respect to the class and accordingly builds a ranked list from the calculated value.
- *ConsistencySubsetEval*: Consistency Subset [Liu & Setiono 1996] identifies a subset of features, by evaluating its worth by the level of consistency in the class values when the training instances are projected onto the subset of features. In conjunction with a search, it aims to identify the smallest subset that has the same consistency as the full feature vector.
- *CfsSubsetEval*: The Correlation-based feature selection (CFS) [Hall 2000; Hall & Holmes 2003; Witten *et al.* 2011a] aims to evaluate the worth of a subset of features by analysing the predictive ability of each individual feature together with the degree of redundancy between several features. This method therefore selects subset of features that have a high correlation with the class while at the same time are not highly intercorrelated.

For the single single-feature evaluation methods (i.e., Information Gain, Gain Ration and Chi Squared)—which do not identify a subset per se, but rather rank the features based on their evaluated worth—I built subsets from the ranked features including the best 500, 250, 150, 100, and 50 features for the evaluation, similar to the approach shown by Fogarty et al. [2004a]. In order to identify which of the feature selection mechanism performs best on the data I randomly selected 4 of the 16 datasets, evaluated their performance with different classifiers, and compared the results of 10-fold cross validations for selecting the best combination. In all four cases the best performance was achieved with the Correlation-based feature selection, in combination with a Support Vector Machine, as described in the Following.

4.1.3 Algorithm and Parameter Selection

The final step was the selection of a classification algorithm and the optimisation of the parameters for this classifier. As mentioned before in the discussion of the feature selection process, the choice of the classifier and the feature subset are strongly interwoven. Therefore, both, the selection of classifiers and feature subsets happened in one-pass, using the Weka Experimenter application. I combined the different feature selection mechanism with four machine learning algorithms for supervised learning: Naïve Bayes [John & Langley 1995], C4.5 Decision Tree [Quinlan 1993], Random Forest [Breiman 2001], and Support Vector Machine [Platt 1999]. As a fifth algorithm I added Weka's ZeroR classifier as a baseline.

Naïve Bayes and C4.5 Decision Tree both are robust and easily applicable algorithms that perform well in many settings, without much tweaking. Therefore both algorithms are not likely to outperform SVM and Random Forest, but are good measure for comparison. As the Decision Tree algorithm—other than the Naïve Bayes—is better in handling feature interactions, both are used.

Random Forest and Support Vector Machines on the other side are more complex algorithms, which have proven to perform very well with high dimensional data. Thereby the Random Forest has an additional strength in its robustness to noise.

The ZeroR algorithm for nominal classes does not make use of any features but simply always chooses the mode—that is, the class that is predominant in the data. It was chosen to get the minimum baseline for the different datasets.

As a result of the Experimenter runs I identified—as mentioned before—the SMO in combination with the CFS generated feature subset to provide the highest accuracy. In the final step I used Weka's GridSearch to go through different parameter combinations, in order to further optimise the prediction quality. This included:

- Varying the four different kernel functions for the SMO algorithm, namely the radial basis function (RBF kernel), the Pearson VII function universal kernel (PUK), the polynomial kernel function, and the normalised polynomial kernel function.
- Varying the complexity constant parameter in 20 steps from 1 to 20.
- And varying the tolerance parameter in 10 steps from 0.001 to 0.01.

The best results were achieved with the normalised poly kernel, a complexity constant of 1 and a tolerance parameter set to 0.001 according to a 10-fold-cross-evaluation performed on four of the datasets. In the Following I discuss the prediction results, I received when applying the CFS feature selection algorithms and the SMO-trained classifier with the mentioned parameters to all datasets.

4.1.4 Classification Results and Discussion

Below I present and discuss the results on the extent to which the *Selective Availability* of users is predictable based on the data collected in the user study. That is, I discuss the performance results of a multiclass classification problem where the task is to assign to a current situation—described through a number of features—the current preference for availability for a certain set of people in form of one of the six availability levels, which are:

- Text Me!
- Online
- Away
- Not Available
- Do Not Disturb
- Offline/Invisible

In the remainder, I primarily discuss the results of predicting the *Selective Availability* based on individual models that were trained for each participant, as this allowed for the best accuracy. That is, one predictive model is build for each of the four *Availability Categories* and for each participant, resulting in 16 predictive models. Technically, this reduced the prediction problem for some of the models from a six-class to a five-, four- or even three-class problem, as not all participants exploited the full range of *Availability Levels* for all *Availability Categories*. For example, P3 never marked himself as *Offline*, hence no labelled data is available to train this class for this participant. However, I also reflect on aspects of building general models over all participants, and other approaches that were explored.

Each probabilistic model was trained with in average 355 (SD=167.0) labelled instances, based on the amount of ESM samples collected by the specific user. The feature selection mechanism in average reduced the number of features to 23.4 (SD=10.8) for each model. As is eminent from the data (cf. Figure 29) the

class distribution for most users is skewed, meaning for each dataset the relative frequency of some classes is much higher than for others. As machine learning algorithm have a learning bias towards the class with the most cases, often resampling is used to shift the class distribution toward a uniform distribution. I therefore applied the supervised Weka *Resample* filter to obtain a more balanced dataset. Finally, I used stratified 10-fold-cross-validations, to come to a performance measure. In the Following the resulting accuracies of this 10-fold cross-validations are shown for all 16 models (cf. Table 6). The classification accuracy thereby is the percentage of correct predictions made in respect to the total number of predictions.

Table 6. The tables show the accuracy in percentage from 10-fold cross-validations performed for each model of each of the four participants and one summary table showing the average over all models. For each models the table shows the accuracy for the ZeroR classifier (i.e., the base probability), the SMO classifier, and the calculated difference between the SMO classifier and ZeroR classifier for all models and the average per participant.

P1 Models	ZeroR (%)	**SMO (%)**	Difference (%)
P1 General	58.15	75.65	17.51
P1 Private	55.53	75.86	20.32
P1 Public	59.76	78.27	18.51
P1 Work	51.31	75.86	24.55
P1 Average	**56.19**	**76.41**	**20.22**

P3 Models	ZeroR (%)	**SMO (%)**	Difference (%)
P3 General	74.40	91.67	17.26
P3 Private	86.90	92.26	5.36
P3 Public	69.64	86.90	17.26
P3 Work	78.57	92.26	13.69
P3 Average	**77.38**	**90.77**	**13.39**

P2 Models	ZeroR (%)	**SMO (%)**	Difference (%)
P2 General	68.62	84.10	15.48
P2 Private	70.29	83.68	13.39
P2 Public	66.53	82.01	15.48
P2 Work	71.13	84.94	13.81
P2 Average	**69.14**	**83.68**	**14.54**

P4 Models	ZeroR (%)	**SMO (%)**	Difference (%)
P4 General	62.50	74.54	12.04
P4 Private	70.37	76.85	6.48
P4 Public	74.54	76.85	2.31
P4 Work	64.35	69.91	5.56
P4 Average	**67.94**	**74.54**	**6.60**

All Models	ZeroR (%)	**SMO (%)**	Difference (%)
Overall Average	**67.66**	**81.35**	**13.69**

The four big tables show the classification accuracy for the participants P1 to P4. For each participant the respective table shows the results for the four models for each *Availability Category* and the average accuracy over all four models. In the smaller, fifth table, we see the averages over all four participants. In each table, the three columns of numbers show first the result of the ZeroR classifier, which acts a baseline for the performance evaluation. It is followed by

the actual result of the SMO classifier and a calculation of the difference between the baseline and the accuracy achieved by the SMO classifier in the last column. In each participant table the average for each participant shows the overall result a user can expect, as the four models together would make up the selective availability of a user.

Already from the descriptive data (cf. Figure 33) it was eminent that each user has one predominant level of overall availability, or at least for one AC. Accordingly this is reflected in the ZeroR results where for each participant the chance that one specific *Availability Level* is selected is always above 50% (in average 67.66%). A system that just assigns this *Availability Level* all the time therefore would be correct in 67.66% of the cases. Hence, the results of the SMO classifier needed to be assessed in respect to this high baseline. From the last column we can see that the SMO for all models performs better, and for some models even considerably better with an average difference of 13.69%.

With an average accuracy of 81.35% the results are competitive with those found in related work (cf. 2.4.5) on interruptibility or availability prediction. To recap, Fogarty *et al.* [2004a] for example achieved accuracies between 79.5% and 87.7% with a naïve Bayes classifier trained from overall 975 oral interruptibility self-reports given by 10 participants, classifying the users as either highly-non-interruptible or not highly-non-interruptible. While the pure accuracy numbers are compatible, it is notable that Fogarty *et al.* limited their problem to a two-class classification and only collected data in a fixed office room setting. Both limitations also hold for the evaluation of the *BusyBody* system, where Horvitz *et al.* [2004] achieved an accuracy of 78.25% with their classification of four participants as either *Busy* or *Not Busy* from in average 1,268.25 (SD=837,14) labelled instances for each participant. With the *SocioXensor* for PDAs and mobile phones Ter Hofte [2007] collected data from ten participants over the course of one week (7 days) in a nomadic setting. From the collected data he was able to predict the availability as *Available* and *Not Available* with an accuracy of 63.9%.

Accordingly, overall the result suggests that the prediction of Selective Availability is feasible and can be an improvement over the status quo (i.e., a manual adaptation of an online status). This becomes even more evident when comparing the 41.5% (SD=16.8) of times in average the manual set Skype status matched the *General Availability* (cf. 3.2.4) of the participants with the 81% (SD=7.9) of times in average the SMO prediction of the *General Availability* is accurate. However, one limitation of this work is the restricted generalisability of the pure numbers as a result of the small sample size, which again is caused by the very specific prerequisites for this study. This is a general issue, also in most examples I discussed in the related work. Therefore, I further present

additional results in the form of a collection of lessons learned, that can inform the design and execution of future work in this direction.

4.1.4.1 Lessons Learned—Predictive Power of Individual Sensors

In the first lesson learned, I present insights on the influence of the different sensors on the results of the prediction. Such information can guide future research when selecting sensors, as they can omit sensors in the first place that might be e.g., too difficult to implement, too resource-intensive, or too privacy-invasive.

Table 7. This table provides a ranked list of the top 15 sensors, based on the number of features from the data of this sensor that were selected to contribute to a participant's model. The numbers show the sum of the number of features that went into the four models for each of the four participants.

#	Sensor	P1	P2	P3	P4	Sum
1	Applications	37	36	50	4	**127**
2	WiFi	20	17	19	23	**79**
3	Voice Activity	2	4	7	18	**31**
4	Focus Title	3	3	4	5	**15**
5	Time	5	4	4	1	**14**
6	Motion	4	0	5	3	**12**
7	Volume Settings	5	3	2	2	**12**
8	Skype	3	3	0	4	**10**
9	Connected USB Dev.	1	3	4	1	**9**
10	IP Address	1	1	2	5	**9**
11	Application Focus	2	3	1	2	**8**
12	Input Idle	3	2	0	3	**8**
13	Active Access Point	2	3	1	1	**7**
14	Bluetooth Devices	1	1	2	2	**6**
15	Calendar	3	1	2	0	**6**

From the results of the feature selection mechanisms CFS, we are able to draw some conclusions on the influence of single sensors on the quality of the prediction. Based on whether one or more of the features related to one sensor was selected for one or more models, we can make assumptions on the respective sensor's relevance. When analysing the feature subsets of all 16 models—that is four per participant—we are able to simply count the number of features that are based on one of the 29+1 sensors, to draw such conclusions. As mentioned before, the CFS algorithm selected in average a subset of 23.4 (SD=10.8) features from the in average 1437.3 (SD=303.6) features per model. In Table 7 we can see that the Applications and WiFi sensor have by far the biggest share among the sensors that contributed features to this models. The two sensors make up already more than half of the 23.4 features of the four models of P1 ((37+20)/4 = 14.25), P2 ((36+27)/4 = 13.25), and P3 ((50+19)/4 =

17.25). In Table 8 we can see that altogether 11 of the sensors contributed at least one feature to at least one model of each participant. We also can see that 10 sensors (i.e., Active Chats, Battery, Connected FireWire Devices, Email, Ethernet Connected, Headphones Connected, Mounted Volumes, Mouse Connected, Screensaver Active, and Second Monitor) did not contribute at all to any model.

From the sensors that were completely unused, I found that some sensor states did not change over the whole study duration, as for example none of the participants attached a FireWire device or mounted a volume during that time. While I assumed that different combinations of attached periphery to the computer (e.g., Headphones Connected, Mouse Connected, Ethernet Connected, and Second Monitor) would allow estimates on the location and situation that data was not used. One plausible explanation is that some of this information is substituted by data of other sensors, as for example the *Connected USB Devices*-sensor would include information on the Mouse and the information captured by the *IP Address*-sensor makes the *Ethernet Connected*-sensor obsolete.

Table 8. This table shows a list of sensors that contributed at least one feature to the final feature subset of a participant's model (✓) and of sensors that contributed no single feature to a participant's model (-).

Sensor	P1	P2	P3	P4	No	Sensor	P1	P2	P3	P4	No
Active Access Point	✓	✓	✓	✓	4	Focus Title	✓	✓	✓	✓	4
Active Chats	-	-	-	-	0	Headphones Connected	-	-	-	-	0
Active Network Interfaces	-	-	-	✓	1	Input Idle	-	✓	✓	✓	3
Ambient Light	-	✓	✓	✓	3	IP Address	✓	✓	✓	✓	4
Application Focus	✓	✓	✓	✓	4	Motion	✓	✓	-	✓	3
Applications	✓	✓	✓	✓	4	Mounted Volumes	-	-	-	-	0
Battery	-	-	-	-	0	Mouse Connected	-	-	-	-	0
Bluetooth Devices	✓	✓	✓	✓	4	Power Connected	✓	-	-	-	1
Calendar	✓	✓	✓	-	3	Screensaver Active	-	-	-	-	0
Connected FireWire Dev.	-	-	-	-	0	Second Monitor	-	-	-	-	0
Connected USB Dev.	✓	✓	✓	✓	4	Skype	-	✓	✓	✓	3
CPU	-	-	✓	✓	2	Time	✓	✓	✓	✓	4
Email	-	-	-	-	0	Voice Activity	✓	✓	✓	✓	4
Ethernet Connected	-	-	-	-	0	Volume Settings	✓	✓	✓	✓	4
Face Detection	-	✓	✓	✓	3	WiFi	✓	✓	✓	✓	4

However, one common detonator of these unused sensors is that most of them are neither computationally nor energetically expensive, as they are just listening for system events. Therefore it might be worth to research if they could not also substitute a more energy consuming sensors, like the WiFi sensor that is constantly polling for nearby networks. Also, these sensors might be very

valuable in order to detect changes in interruptibility, as connecting or disconnecting a second monitor, an Ethernet cable, etc. could indicate a transition between tasks. However, due to the random ESM sampling in a relative coarse-grained 30-minute rhythm, I do not have data, to backup this assumption. An additional context-aware experience sampling study that would be triggered by such events could provide more insights.

Further, i analysed which of the feature engineering steps contributed the most features to the final subsets. I found that 71.93% of the features were features that were constructed by integrating sensor values over time. Thereby 32.89% of the features took into account the 14 minutes interval, 21.39% took into account the 5 minutes interval, and only 17.65% took into account the last 8 minutes. Overall this shows that the feature construction mechanism were able to provide richer features for the classification task.

4.1.4.2 Lessons Learned—Personal versus General Models

In the analysis of the predictability the focus very early was on building personalised models for each individual participant. The two reasons can be found in the data itself: the highly personalised nature of the labels, which are basically preferences of individual users, and the highly personalised nature of the features, which to a wide extent describe the individual users' specific contexts.

So, for once, this step is clearly motivated by the upfront analysis of the ESM data discussed in 3.2.4. Already for the four subjects it became obvious, that the there is a great variance in the selected availability levels, with P3 having a clear leaning to being highly available, P4 having a clear leaning to being unavailable and P1 and P2 both using a broader range of availability levels. We therefore can assume that there is a strong individual perspective on how available they want to be during the day—an inference that can be fairly made based on studies on *individual differences* [Dillon & Watson 1996] in the ability to deal with interruptions [Mark *et al.* 2008; Ruppenthal & Chignell 2002]. However a specific that a general model trained for all users is not able to reflect.

Second, such general models could solely rely on features, which are common for all users and therefore are variables that are able to describe the specific context for all users. In my approach, this would lead to three distinct challenges:

- A feature with a high information gain for one user but low information gain for all others.
- A feature that contains contradictory information for different users.
- And different features that contain similar information for several users.

In order to further illustrate these three challenges, we need to keep in mind that features in my approach are and in future should be extracted in an automated matter. The majority of resulting features at the end of this automated process often denote in form of a Boolean value the presence or absence of a single specific context variable, like for example the proximity to a WiFi-network access point (e.g., `WiFi.SSID.MyHome='true'`).

Hence, for the first challenge, if this WiFi-network for example is the home network of one specific user, it probably has a very high information gain for this specific user. However, for most other users, this feature has no predictive power. Accordingly, when building a general model, such a feature would most certainly be rejected in the feature selection process, as it only contains missing values for all other users.

Second, a feature can contain contradictory information for two or more users. Again, as an example we take a feature, denoting the presence or absence of WiFi-network access points—but this time in an office environment. Again, the patterns formed by the presence of several specific WiFi access points allow us to implicitly infer that a user is at a certain location (e.g., in an office room). However, whether this location refers to the own office versus to the office of a colleague or superior, the availability preferences when at this location might be highly conflicting for two different users. Again, in the feature selection process, features with contradicting information would be disregarded.

And third, while some context properties might correlate very well over a number of different users, on a feature level they cannot be easily merged in an automatic process. To give an example, it seems plausible that the usage of a specific type of application, like for example an email client or spreadsheet software, might lead to similar preferences for being interrupted for different persons. However, on a feature level it is difficult to automatically combine this data, as the values for the sensor *Application Focus* might be "Outlook", "Thunderbird" or the OS X "Mail.app" depending on the user.

All three described challenges could be circumvented, if a semantic layer between the raw sensor data and the features would be introduced, allowing the model to learn from abstracted concepts like e.g., 'home', 'personal office', 'colleague's office', or 'email client'. However, this would in most cases come with an additional effort for the user, to train for example which WiFi Access Points relates to which personal location, etc. and further individual semantic abstractions.

Concluding, while it is clear that from a users' effort perspective, pre-trained models would be preferable, personalised models are likely to achieve far superior adaptation results. In the analysis, I found that the performance of general models— combining the data collected over all participants—did not at all perform up to par with models tailored for individual users. In tests with the

Weka Experimenter the general models performed most times worse than the ZeroR base probability and only few times insignificantly better.

Hence, for a satisfactory adaptation experiences, only personalised models should be used. This can be achieved by employing an interface agent, which is a software system that becomes "gradually more effective as it learns the user's interests, habits and preferences" [Maes 1994, p. 31]. Though, as such an agent-based system needs to learn everything form scratch, the utility for users is very restricted at the beginning or in novel and unknown situations, where the system has no or insufficient information on the user's preferences. An issue referred to as cold-start problem [Schein *et al.* 2002] in the recommender systems community. As a solution in the field of interface agents and personalised user interfaces, the idea of collaborative interface agents [Lashkari *et al.* 1994] was formulated. Such agents, that share their knowledge among several users, were also deemed practical in the field of automatic availability management, where for example Fogarty *et al.* [2004b, p. 315] suggested that there might be "certain indications of interruptibility that are useful across different groups of people". Fogarty et al. therefore built general models for each group of subjects among their study participants (manager, researcher, and intern) in order to overcome the cold-start problem—yet, this approach was never validated in a user test. Yuan *et al.* [2017] propose a two-stage hierarchical model based on personality traits. They perform an upfront personality test to identify the user's personality traits. Instead of using one general model, they built different interruptibility models for groups of users that have a similar personality. For each user, a model from such similar users is used to overcome the cold-start problem. The model then is refined over time with training data from the individual user. With this approach, they were able to improve the average prediction quality for various performance measures over 10 percentage points. However, both approaches do not address the issues with the personalised feature space.

4.1.4.3 Lessons Learned—Patterns for Selective Availability Configurations

One further lesson I learned from the analysis of the collected data was a tendency of the users to reuse specific availability patterns for the three categories in reoccurring situations or settings. For example, during weekday, a user would be highly available to contacts related to work, less available for private contacts and unavailable for other contacts. In fact, I found that of the 6^3 (= 216) permutations that are possible—by choosing one out of the six availability levels for the three availability categories—the participants in the study only used in average 23.25 (SD=14.97). This becomes obvious, as some combinations are very unlikely, as for example being highly available for the category Public/Others and at the same time offline for private and work related

contacts. Hence, I was interested in finding out how many different configuration patterns users utilise to express their selective availability in the majority of time. After retrieving for each user all triplets (3-tuples) that occurred in the data, I used a clustering approach with the EM-algorithm (expectation-maximisation) [Dempster *et al.* 1977] to identify patterns that best fit the respective participant's configurations.

Table 9. Predominant availability patterns (I-IV) for the participants (P1-P4) from the ESM data (1=Offline, 2=Do not disturb, 3=Not available, 4=Away, 5=Online, and 6=Text Me!).

	P1				P2				P3			P4			
	I	II	III	IV	I	II	III	IV	I	II	III	I	II	III	IV
Private	4	5	3	2	5	6	4	1	5	4	3	1	2	3	4
Public	2	2	1	1	5	6	1	1	5	4	3	1	2	3	4
Work	5	1	4	2	5	6	4	3	5	5	3	1	3	3	4

As can be seen in Table 9, four patterns were identified (I, II, III, and IV) for three of the participants, and three patterns (I, II, and III) were identified for one participant. In average these patterns cover 84.32% (SD=3.40%) of the configuration needs of each participant. As the computational complexity of predicting a rather small number of patterns versus the whole of number of possible permutations, I investigate the predictability of these dominant availability patterns for each user. In average 82.15% (SD=7.60%) of the availability configuration patterns were classified correctly with the Weka SMO classifier—however the data for this experiment was highly unbalanced (as can be seen exemplarily in Table 10).

Table 10. Confusion matrix of the results for P1, predicting the four mostly used pattern I-IV (1=Offline, 2=Do not disturb, 3=Not available, 4=Away, 5=Online, and 6=Text Me!).

	Classified as			
	I	II	III	IV
I = {4,2,5}	231	6	5	0
II = {2,1,2}	16	66	5	0
III = {3,1,4}	43	4	21	0
IV = {5,2,1}	0	0	0	20

While overall this result is not too encouraging—as it is worse than the results I obtained for predicting the three availability categories—the general existence of these patterns is an interesting insight for future work. Much like in the *Disclosure Template* concept (cf. 3.1) such patterns could reduce the effort for the manual adaptation, when instead of individual setting the availability for different availability categories, the user can select out of a reduced set of meaningful patterns.

4.1.5 Conclusions

I showed, that the prediction of *Selective Availability* is generally feasible. However, from the experiment—already with this small number of relative homogenous users and settings—the lessons learned show that the individual, highly dynamic, and hardly observable nature of availability preferences requires a new approach.

In the Following I therefore further extend the reasoning and rationale behind my approach towards a concept for adapting presence and availability configurations of users based on preference learning in context.

4.2 Considerations and Requirements for Building Adaptive Systems

The general idea of automatically managing the presence and availability based on the user's context clearly puts it inline with other work on context-aware and adaptive systems. Adaptation through context-awareness allows migrating "complexity away from the user and into the system" [Cheverst *et al.* 2001, p. 5:8]. Hence, context-aware systems support users by letting them delegate tasks to the computing environment. In this thesis, the task is to adequately manage the availability of a nomadic user towards different contacts. As we learned before, context-aware ubiquitous environments perform tasks automatically for the users though monitoring the current situation with sensors and deducing needs of the users based on trained models or specified rules. Yea *et al.* [2012] distinguish thereby two different approaches for identifying situations in pervasive computing: *Specification-based* approaches and *learning-based* approaches. *Specification-based* approaches are based on expert knowledge formulated in form of logic rules or represented as ontologies that allow to reason and infer situations from sensor data. *Learning-based* approaches are based on techniques from machine learning and data mining in order to train models that express association relations between different situations and sensor data.

Specification-based approaches work well, if the rules for such adaptations are of simple nature and can be easily defined manually. To provide an example: a task as easy as muting a mobile phone each time a user arrives at a certain location, like the lecture hall, can be packed into one simple rule. Tools using a *specification-based* approach as for example *Marco Polo* [Symonds 2007] or *Tasker* [Crafty Apps EU 2016] support nomadic users with configuring multiple rules and the resulting actions on OS X laptop computers or Android Phones respectively. In UbiComp research lightweight tools for end-user configuration of UbiComp environments [Salber *et al.* 1999; Schirmer & Gross 2011] aim to

help users with automating their office or home, by supporting the visual specification of rules.

However, if the nature of the adaptations is becoming more complex, a manual specification of rules quickly becomes infeasible. At this point, often *learning-based* approaches are used, to handle more entangled adaptation decisions. Smartphones for example are able to tell apart different types of activities of a user. Activity recognition [Bao & Intille 2004] for example allows applications on a smartphone to estimate whether a user is currently walking, driving, standing, or bicycling through applying machine learning algorithms on data collected by the smartphones accelerometer sensor. Based on the detected activity, the mobile application can now adapt the user interface to the current situation. To enable devices to perform such activity recognition, a trained model has to be provided to the device. Such a model allows the device to relate an effect (i.e., an accelerometer data pattern) to one of several causes (i.e., a form of human locomotion), e.g., based on probability. To generate such a model, labelled data is collected from a very small portion of representative users and provided to a machine learning algorithms for training the model. However, this approach also has its limitations. One effect that can be observed is, that already with this most simple machine learning problems, the generalisability of such models for a wider audience of users can be problematic. By generalisability I mean the external validity of the trained model—i.e., to what extent the prediction results can be generalised to other situations, devices or people. I have highlighted this challenge for the prediction of selective availability in the previous section. Also Bao and Intille [2004] found in their approach of recognising activities from user-annotated acceleration data that "subject-independent training data" allowed general a good identification of different activities. However, some activities could be detected more reliable with "subject-specific training data" by building *personalised models*. One very illustrative example for this challenge in HCI history, are systems that provide speech or handwriting as input modalities. As speech and handwriting is inherently individual, systems that are trained by the individual user for a long time performed significantly better then a generic recognizer [Kienzle & Chellapilla 2006]. Only recently—with the vast amount of data that is now available and processable to companies like Apple or Google—general models for speech recognition become slowly up to par. However, the rationale for relying on general models despite their lower performance is that the users of a system are not bothered with training the system before using it. To overcome this challenge in activity recognition, various different approaches have been developed [Reiss & Stricker 2013; Zhao *et al.* 2011], that try to optimise general models to specific users.

As we have seen in the review of related work, notification systems and attention-aware user interfaces are one prominent area for applying *learning-based approaches*—and are also among the most prominent fields in HCI for applying machine learning techniques [Chen *et al.* 2010]. They also have repeatedly been shown to be an application area for *personalised* user models, as they outperform general models [Fogarty *et al.* 2004a; Horvitz & Kapoor 2008; Horvitz *et al.* 2004; Kapoor & Horvitz 2007; Kern & Schiele 2006; Pejovic & Musolesi 2014; Pielot 2014]. However, most previous work settled on showing this effect of personalisation outperforming general models in a post-hoc analysis of collected user and sensor data—only a few came up with meaningful approaches on how this can be transformed into real systems (e.g., [Fogarty & Hudson 2007; Kapoor *et al.* 2007]). Even more complexity is added when adaptation decisions are made in nomadic environments. Such decisions are based on multifarious, simultaneous input factors as users' whereabouts, actions and social interactions that are sensed in the physical and digital realm, from a plethora of diverse sensors providing constant updates on very individually interpretable situations. As reflected on earlier (also see [Fetter *et al.* 2011b]), under such conditions general, pre-trained models will likely fail to offer adaptations of an adequate quality.

Hence, to overcome this challenge, the notion of lifelong learning systems [Silver *et al.* 2013; Thrun 1998] for user modelling was brought up by Kapoor and Horvitz [2007]. The underlying idea is to continuously train a predictive model by collecting user feedback in form of label data on personal, unobservable states over a prolonged time (i.e., potentially for the whole usage time). The idea is to build systems that continually improve themselves over time, thus successively reducing the number of requested labels as the uncertainty of the model drops. In new situations however, the system is expected to request more labels again, in order to improve the model for such types of situations. In other words, lifelong learning aims to strike a balance between low training efforts for the users while offering the improved prediction quality of personalised models.

In this sense, a lifelong learning system also can be seen as a *Mixed-Initiative* user interface [Horvitz 1999], engaging in a dialogue with the user, to resolve uncertainties. This aligns with Parasuraman *et al.* [2000], who identified four classes of functions to which automation could be applied: *acquisition of information*, the *analysis of information*, the *decision and selection of an action*, and the *implementation of an action*. Thereby—for each of these types—the amount of automation can vary on a continuum between fully automated to fully manual. A system for adapting the availability of a user aims to progressively learn on the *decision and selection of an action* from the user—thus transforming a previous manual task into a mostly automated task. The *acquisition of*

information as well as *analysis of information* should be fully automated for such a system. The same holds for the actual adaptation of changing the availability status (i.e., *implementation of an action*) based on the learned decisions.

Finally, as a lifelong learning system for predictive user modelling falls in the general category of adaptive systems, it should further fulfil certain general properties of adaptive systems, in order to improve the user experience. A variety of such user requirements were identified in previous work, stating that for the user, the adaptation decisions should be *foreseeable* [Cheverst et al. 2001; Höök 2000; Jameson 2006; Shneiderman & Maes 1997; Tsandilas & Schrafel 2004], *comprehensible* [Höök 2000; Jameson 2006; Norman 1997] and *trustable* [Cheverst et al. 2001], *controllable* [Höök 2000; Jameson 2006; Norman 1997; Shneiderman & Maes 1997; Tsandilas & Schrafel 2004], *unobtrusive* [Jameson 2006], *accountable* [Bellotti & Edwards 2001; Rice & Beresford 2006], *intelligible* [Bellotti & Edwards 2001; Lim & Dey 2010] and *privacy preserving* [Höök 2000; Jameson 2006; Norman 1997]. A main focus of this this work certainly is on the requirements of privacy and control, as "the more advanced automation gets, the more crucial it is for the user to have control strategies for regulating automated disclosure with others." [Vihavainen et al. 2014, p. 56]. In a setting with nomadic users, the system should also be stable in term of spontaneous sensor configurations, hence requiring opportunistic activity and context recognition systems [Kurz & Ferscha 2010; Lukowicz et al. 2012] that need "to react properly at runtime to [...] dynamics in the sensor infrastructure" [Kurz & Ferscha 2010, p. 136]. Ideally, this leads to a "self-configuring, self-optimising, and self-repairing system" [Gama 2008, p. 218]. Accordingly, all these aspects further defined the design space and requirements of the proposed LiLoLe Framework.

4.3 Exploration for an Architecture for Personalised Adaptations

As a further step on the way towards this integrated architecture, I—in a short digression—introduce two explorative systems that informed the implementation of the LiLoLe Framework: *LocaRhythms* [Fetter & Gross 2009b] and *CoDaMine* [Gross & Fetter 2008]. Both systems have in common that they continuously learn and so improve personalised models to help users manage their availability and presence. The implementation of both systems informed the final architecture and delivered components and building blocks for the final system.

4.3.1 LocaRhythms

First, *LocaRhythms* [Fetter & Gross 2009b] applies real-time data mining to help people signalling their availability and presence in a privacy sensitive way. With

LocaRhythms I focussed on the physical presence of users as a mean for assessing their availability. Knowing about a contact's whereabouts can help to arrange spontaneous meetings, allows to reciprocally estimate the availability, or simply can give a feeling of connectedness over distance [Ashbrook 2002; Ashbrook & Starner 2003; Begole *et al.* 2002; Hightower *et al.* 2005; Kang *et al.* 2005; Marmasse & Schmandt 2002; Zhou *et al.* 2007].

LocaRhythms uses clustering algorithms on GPS traces of users to identify their *significant places*. A *significant place* is a location "of importance to users expressed by the duration and frequency of past visits" [Fetter & Gross 2009b, p. 64]. It further utilises Markov Models [Rabiner 1989] to allow predictions about a user's current and future whereabouts.

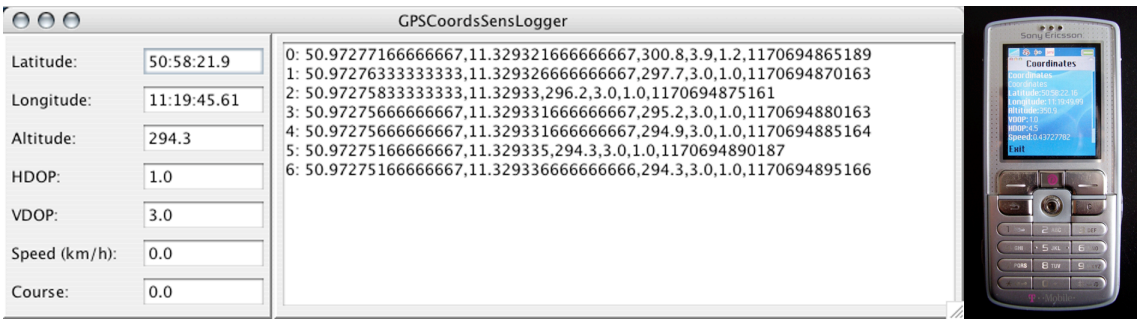

Figure 34. GPSCoordsSensLogger—J2SE version running on a Apple MacBook Pro (left) and J2ME version running on a Sony D750i mobile phone (right) [Fetter & Gross 2009b].

To use *LocaRhythms*, the users need to continuously capture their current position. We implemented an application called *GPSCoordsSensLogger* in a J2SE and a J2ME version (cf. Figure 34). The application allows the users to collect such data on their laptop computer (J2SE) or mobile phone (J2ME)—both in combination with a Bluetooth GPS receiver—in a fixed interval. The collected GPS data is either send directly to the *LocaRhythmsServer* if the client is online, or stored on the device and uploaded later in batch if the client is temporarily offline.

On the server, the collected GPS traces are analysed with the *time-based clustering* algorithm [Kang *et al.* 2005] to identify significant locations. The algorithms is specialised to time-series of position data and sequentially works on the data to identify if a user stays for minimum of a pre-specified time (in our case 300 seconds) within a certain radius (in our case 15 metres) to detect significant locations. In a second step—the fusion step—the algorithm checks if the newly found significant location overlaps with a previously found significant location. If this is the case, the algorithm assumes that it is the same location, and merges the two locations by calculating a new centroid from the two centres of the clusters. As I identified some problems with this algorithm when the GPS signal was lost, I optimised the algorithm for such unsteady GPS data—as described in [Fetter & Gross 2009b]—to achieve better results. The outcome of

this time-based clustering process is a history of the users' stays at significant locations, together with the respective start and end time of each stay.

In order to make predictions from this data, we use Markov Models [Gambs *et al*. 2011]. As the title of the work suggests, people have different rhythms for visiting locations. Some routines are apparent daily patterns, like going to work or to the cafeteria for lunch. However, there are also manifold exceptions from these routines that manifest in different temporal deviations, as a result of holidays, weekends, special weekdays for teleworking, etc. To at least minimally acknowledge for this, *LocaRhythms* builds several models for different temporal granularities. That is, one model for each hour of the day (24), one model for each day of the week (7) and one model for each month of the year (12), resulting in 43 models. In *LocaRhythms* uses Jahmm—a Java library offering an implementations of "Hidden Markov Model (HMM) related algorithms" [François 2006]—to build these models based on the provided observation sequences. An observation sequence is basically a discretised log of the visited locations for certain time frames, which are generated each time the user leaves a significant location. In order to best deal with differences in frequency and durations of stays, we cut longer stays into smaller time-slices of 100 seconds. Depending on the hour(s) of the day, weekday(s), and month, the time slices are added to the different respective files as observation sequences or part of observation sequences (cf. Figure 35). Each time there is a new complete observation sequence for one of the 43 models, the respective model is updated by computing the state transitions. This happens for every hour, day, and month accordingly. A simple weighting mechanism later allows combining the models for different temporal granularities (i.e., hour, day, and month).

```
4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;2;2;2;2;2;2;2;2;2;2;2;2;
4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;4;2;2;2;2;2;2;2;2;2;2;
4;4;4;4;4;4;4;4;4;4;4;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;
7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;7;2;2;2;2;2;2;2;2;2;
```

Figure 35. Four exemplary observation sequences of the 11 o'clock data of one user. The data shows the sequences of placeIDs for the time slot between 11 and 12 am for this user from 10 to 13 February 2009. The sequence just covers the stays at significant locations, and omits the transition between these locations—hence the difference length of the sequences.

If now a user that wants information on current or future whereabouts of another user, there are different possibilities for querying the *LocaRhythmsServer*. For the prototype system, *LocaRhythms* offered two simple GUIs to query the server for a specific moment or time slot. In both cases, the result was returned in form of a KML file [Open Geospatial Consortium Inc. 2017], which automatically was opened and displayed in Google Earth (cf. Figure 36). For the query, the current GPS position—if available—was

combined with the 43 models. If the user queried for a specific moment—defined by a weekday and the hour of the day—the result would be the actual (if available) or most likely current location, and the most probable next locations. The visualisation in form of bars at the respective locations also allowed visually inferring the probability for the locations from the different shades of green (cf. Figure 36). If the query is for a specific time slot, the probabilities for all significant locations are presented in form of segmented cylinder. Each sector refers to an hour in the specified time slot and the shading of each sector allows assessing the probability a user's availability at a significant location.

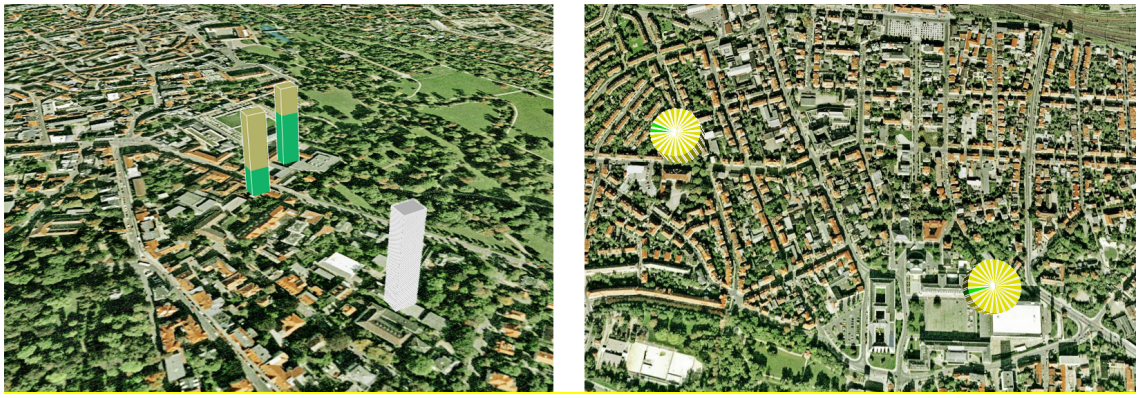

Figure 36. Screenshots of the two types of visualisations of LocaRhythms data [Fetter & Gross 2009b].

LocaRhythms fostered my understanding on how to deal with time-series of sensor data in Sens-ation. Important factors are the preservation of the sequential ordering of sensor events before processing, strategies for windowing mechanisms, especially when dealing with events and states, and how to integrate different temporal granularities. From the perspective of the sensor data collection, the implementation also provided new building blocks for Sens-ation, as for example handling of sensors that go beyond single values, like the GPS readings with values for longitude, latitude, altitude, etc.. Further, the *GPSCoordsSensLogger*, denotes the first step towards an integrated client for the background collection of sensor data on the client side, and informed the design of the *PRIMI Advanced Sensor Suite*.

4.3.2 CoDaMine

Also *CoDaMine* [Gross & Fetter 2008] constantly observes the user interaction to improve the privacy of users. In short, *CoDaMine* analyses the IM exchange of users to make suggestions for the users' privacy configurations. Based on the content of the exchanged messages, the system is able to suggest adding an IM contact to other contact groups, with different privacy settings.

From a software architecture viewpoint, *CoDaMine* was the foundation of embedding the Weka machine learning toolkit [Hall *et al.* 2009] into Sens-

ation's inference engines [Gross *et al.* 2006] in a way that sensor data can be easily transformed into Weka's data structures.

One central aspect of the implementation was to provide also mechanisms for updating the data structures with new emerging classes and attributes for incremental learning. Incremental learning is most often only regarded as the problem of E-IL (Example-Incremental Learning) [Zhou & Chen 2002]. That is, introducing new trainings examples, after a system has been trained. The already trained system should in the best case utilise this new examples while conserving the previously learned knowledge. In the case of *CoDaMine* incremental learning is also strongly concerned with C-IL (Class-Incremental Learning) and A-IL (Attribute-Incremental Learning) [Zhou & Chen 2002]. That is, additional classes (C-IL) or new attributes (A-IL) are emerging after a learning system has been trained. Other than for E-IL, solutions for C-IL and A-IL are still sparse, even though they are becoming now an emerging topic in data stream mining (e.g., classification under streaming emerging new classes (SENC) [Mu et al. 2017] or classification with streaming features [Yu et al. 2015]). In *CoDaMine* new classes in form of new contact groups can emerge as well as new features from the steadily growing corpus of messages. Therefore parts of *CoDaMine* were reused when implementing the LILOLE Framework.

Overall, the implementation of *CoDaMine* and *LocaRhythms* greatly helped to inform the design, concept, and architecture of the final system and the proof-of-concept implementation.

4.4 Continuously Sensing the Context

The continuous collection of sensor data is the foundation of building context-aware applications. In order to allow applications to continuously adapt themselves to the current context, sensors offer the underlying means for such applications—they allow detecting changes in or transitions between contexts. In the Following I motivate the design of a system that is able to continuously sense the context of a user by describing the requirements I identified. I further explain the ideas and mechanisms underlying the design of the *PRIMI Advanced Sensor Suite* and its *Sensor Daemon*, a system that was developed to provide flexible and simple means to collect data from various sensors.

In order to establish some common ground on the meaning of the term *sensor* as it is used in this work, I start with a brief definition that departs from a conventional understanding of the term *sensor*, which often limits the term to some form of electronic component. One example of such a too narrow definition is given by the US National Institute of Standards and Technology (NIST) [2011] describing a sensor as a "transducer that converts a physical, biological or chemical parameter into a electrical signal, for example:

temperature, pressure, flow, or vibration sensor." The definition goes on saying: "Sensors also measure some physical parameter and return digital data representing that parameter [...]". While this definition works for many areas in the field of engineering, many papers written in the fields of HCI, CSCW and Ubiquitous Computing use the notion sensor often with a slightly broader understanding. As reflected on among others in [Gray & Salber 2001; Kurz & Ferscha 2010; Lukowicz *et al.* 2012; Prinz & Gross 2004; Salber *et al.* 1999] they depart from this classic understanding of a sensor, as a tiny piece of embedded hardware. Then again, other definitions are too broad, as for example describing a sensor as a "device that receives and responds to a signal or stimulus" [Fraden 2010, p. 1]. Accordingly, in this work a sensor is defined as "a soft- or hardware component, or its combination that monitors and records the state of an artefact, an entity or the environment in the digital or physical realm" [Fetter et al. 2011b, p. 505].

This definition fits to the appropriation of the word sensor, and accordingly the idea of *context sensing* [Gray & Salber 2001; Salber *et al.* 1999], in the area of context-aware computing. Accordingly, such sensors can produce a broad range of results. Simpler sensors monitor single properties in the physical realm (e.g., the temperature of a room) or the digital realm (e.g., whether a user is logged into their IM account or not) or both (e.g., physical mouse and keyboard activity measured through software). More complex sensors have means to filter, to amplify or to normalise the sensed data. They are able to combine several sources, to take into account their own history, and to apply available knowledge and rules to the data like learned statistical models, in order to provide higher-level information. One example for such more complex sensors is an inertial sensor, which can provide information on a physical object's velocity, orientation, and position based on combining data from accelerometers, gyroscopes and magnetometers. Another example from the digital domain is a sensor that can sense the mood or the stress-level of a person by using text-mining algorithms to analyse emails or instant messages. What most sensors have in common, however, is that they usually continuously collect data over an on-going period of time.

In the field of context-aware or ubiquitous computing basically two different approaches for collecting sensor data [Voida *et al.* 2014, p. 293] can be distinguished, defined here as *environment-centric* and *user-centric* sensor data collection. In the first approach, the *environment-centric* approach, the sensors are installed at fixed locations, in the surroundings of the home or workplace, where they are attached to the room, the furniture, the infrastructures, and fixtures [Begole *et al.* 2004; Begole & Tang 2007; Cohn *et al.* 2010; Tapia *et al.* 2004]. In the *user-centric* approach the individual users themselves or their electronic devices (e.g., mobile phones, laptop computers) are equipped with or

used as sensors [Bao & Intille 2004; Ho & Intille 2005; Horvitz *et al.* 2004; Ter Hofte 2007]. Beckwith [2003] uses the terms *personal systems* and *infrastructure systems* for referring to ubiquitous systems that rely on either one of the two approaches. In many settings however, the two approaches may occur in combination [Fogarty *et al.* 2004a], as each comes with its individual benefits and downsides. For example, in settings where the context of an individual user is of prime interest, the user-centric sensor data collection approach allows for an easier mapping of sensor data to the user while the environment-centric approach makes it necessary to infer the context of an individual user from the global data. On the other side the environment-centric approach makes it feasible to collect data on a greater number of users, and also on the state of the environment (e.g., is a meeting room currently occupied). Based on the notion of Voida *et al.* [2014, p. 295] who categorise three different research scopes for the usage of sensor data streams (i.e., egocentric, group-centric, and space-centric), the following Table 11 provides an overview on how appropriate the two sensing approaches are in respect to the effort for mapping the sensor data to entities in the research scope.

Table 11. Appropriateness of the two sensing approaches for different research scopes (++ = very appropriate, + = appropriate, 0 = neutral).

		Research Scope		
		Ego-Centric	**Group-Centric**	**Space-Centric**
Sensing	**User-Centric**	++	+	0
	Environment-Centric	0	+	++

Further, the environment-centric approach moves the sensing in the background, and does not effort the user to constantly carry a specific device or sensor. However, the environment-centric approach currently is often spatially limited, to a specific room, building or campus while the user-centric approach easily allows the user to collect sensor data at different whereabouts. To minimise the burden for the users in the user-centric approach, a suitable approach is to augment a device the user already carries with them, with sensing capabilities e.g., by installing software that collects the sensor data on the users' mobile phones.

Based on these premises the work here focuses on the *user-centric approach* for collecting sensor data. The application of the user-centric approach removes the need of mapping the collected data to individual users for our egocentric research scope. In order to collect the data, I designed an application that runs as a daemon process on a user's laptop computer. The design of this daemon is founded on a number of requirements that I identified in form of lessons

learned [Fetter *et al.* 2011b] during the development and testing of different approaches. Further, they are informed by the experience of other researchers in the field of context-sensing and adaptive systems (e.g., [Biehl *et al.* 2013; Gray & Salber 2001; Jameson 2006; Kurz & Ferscha 2010; Norman 1997]). In the Following I report on properties that are central when designing for sensor data collection on a user's device and comment on how I achieved satisfying them in my implementation of the *PRIMI Advanced Sensor Suite (PASS)*:

- *Robust*: In order to be able to continuously obtain valid data the first basic requirement is the robustness of the sensor data collection process to failure and outage. The software has to continuously collect data during the normal context of the device's use. Accordingly this includes robustness towards restarts, idle-times, system and application crashes, as well as reconfigurations of system or hardware settings. I therefore included mechanisms that automatically start and stop the daemon processes and mechanisms that automatically restart single sensors that stopped delivering data.
- *Unobtrusive*: A further requirement is the unobtrusiveness of the sensor data collection process for the users. In order to guarantee a continuous collection the process should not get in the way of the users normal utilisation of their devices. Therefore, sensors should not exclusively allocate some of the resources of the system (e.g., blocking camera or microphone) and in such way interfere with the users normal interactions (e.g., making a video chat). They also should not activate applications or services, which the users did not activate, or even did deliberately deactivate. For example, if the Bluetooth or WiFi functionality is turned off, a sensor should not turn this hardware on to retrieve values. The same is true, if an application is not currently running although a sensor needs it to access information (e.g., number of unread mails from the mail client). Vice versa, the sensor data collection should always be preemptable by users' tasks, which in any case should have a higher priority. My implementation of the sensors therefore only uses the underlying hardware, services or applications briefly for reading in a sensor value, and the releases the resources again between each reading. The pauses between the samplings would provide enough time for any user specific task to allocate the resource (e.g., start a video chat). If the sensor would find a resource already allocated, it would repeatedly try to obtain the resource and return NULL values, until the resource is obtainable again (e.g., after the user ended the video chat).
- *Resource-efficient*: In the same sense, the collection of sensor data for adaptation is a low-level task, which should not overly restrict the

normal usage of the device in terms of consumable resources. Consumable resources in this case are the battery life, the CPU load, the utilised network capacity or the memory usage. As it is clear, that any sensing activity will affect those resources, it is critical to reduce the effect on the users and their current tasks. In my implementation I therefore always focused on possibilities to minimise the overall CPU and memory usage. For example, the sensors that interpret video or audio data rely on algorithms that are computational inexpensive—although other algorithms promised to extract more information, but would have drastically impacted the degree of capacity utilisation of the computer. Further, the frequency of sensing greatly impacts the battery life. My approach therefore supports balancing the sampling-frequency based on the resource intensiveness of an individual sensor in comparison to the expected information gain from sampling more often.

- *Configurable*: In order to be flexible and applicable to a wide range of settings, the sensor data collection needs to be configurable. This is necessary to adapt the sensing to the individual circumstances of the user, the environment, the device, and of course the task, that needs to be automated. In my case the individual sensors could be individually activated and deactivated (cf. Figure 37). For each activated sensor, individual settings allow a finer adjustment of the sensing process (e.g., the adjustment of the sampling rate between 1 second and 24 hours). Some sensors further allow configuring domain-specific settings. For instance the voice activity sensor provided detailed configuration options to specify when the sensor should report voice activity (e.g., a threshold for the minimal length of continuous voice activity).
- *Controllable*: While the sensing should happen in the background, widely unnoticed by the user, the user at any time ultimately needs to stay in control of the sensing process. Therefore mechanisms, enabling the user to immediately pause or stop the sensor data collection, are needed. Whether the reasons for doing so are privacy related, due to the performance of the system, or otherwise. In my approach a single click of a button allowed to toggle between starting and stopping the sensing for all sensors.
- *Privacy-sensitive*: The collection of the data should happen in a privacy-preserving manner. Even if the data is not passed over to third parties for processing, the collection process should not gather more data than is necessary for the purpose and should not store the data longer than is necessary for the purpose. In my case, a hash-based privacy

algorithm in the style of the *Subtle* system [Fogarty & Hudson 2007] is used. Every collected nominal value (e.g., title of application in focus, name of the nearby WiFi access points) is converted into MD5 hash value. For example, the name of the application Firefox is converted into the Hash-Value 16c189cc195ea9b837bc53303588520b. This way a human cannot easily decipher the data. But as each hash-value is almost unique (i.e., collisions have a probability of roughly 1 in 10^{38} for 128-bit hash), the values still are meaningful when processed by machine learning algorithms. Also, while my solution currently offers different options to log and store the data for our experiments, the logging can be completely deactivated when the sensing approach is used for live adaptation.

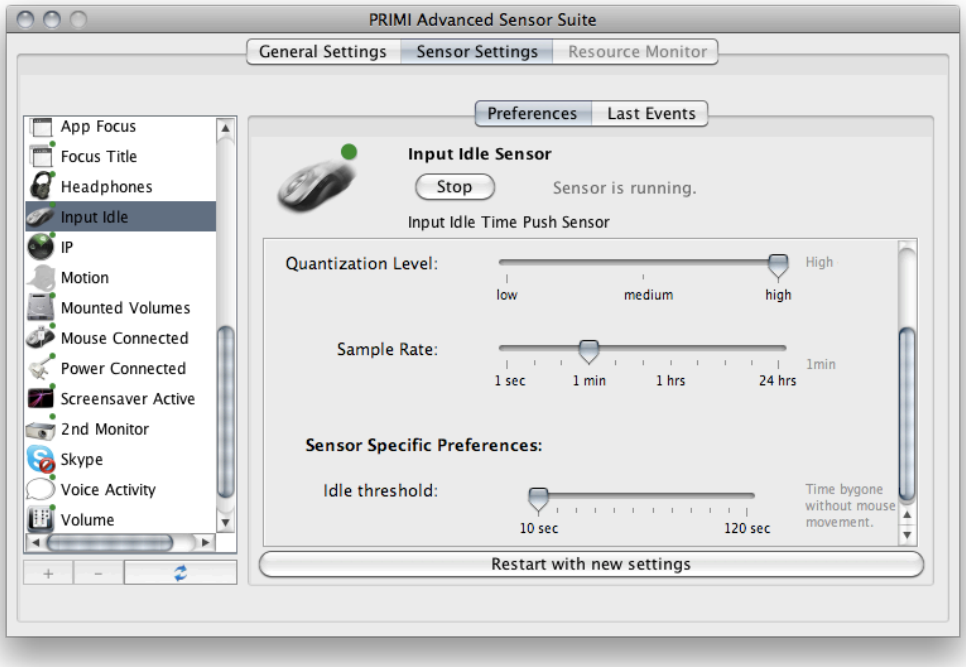

Figure 37. PRIMI Advanced Sensor Suite with the Sensor Settings Dialogue of the Input Idle Sensor open [Fetter *et al.* 2011b].

- *Extendable* and *Maintainable*: In order to be flexible for different settings and scenarios, it makes sense to design the sensor data collection system in a way, that it can be easily attuned to new circumstances. To accommodate for these issues in my approach I rely on an open and modular software design. I apply a plugin approach, encapsulating each sensor in an own self-contained file—comparable to the conception of sensors as wrappers [Kurz & Ferscha 2010] or the *Context Widgets* in the *Context Toolkit* [Salber *et al.* 1999]. This way each sensor can be maintained, extended, or updated separately and new

sensors can be easily deployed. Further, this allows providing a standardised access interface by abstracting information from multifarious sources [Lukowicz *et al.* 2012]. This way, if a sensor implementation does not work anymore, because the underlying hard- or software changed, it can be easily exchanged by a new implementation that provides similar results.

- *Adaptable:* The context-sensing layer is on the lowest level of a context-aware adaptive system. Ideally, the data is sensed once and disseminated to different services that interpret the sensed information to provide different forms of adaptation. Hence, the context-sensing layer should be adaptable to allow forwarding of the sensed information to new adaptation services. In our approach I provide several services to store, analyse or forward the data—most times in an easily interpretable XML-format. While one central element, is the forwarding of the sensor data to the Sens-ation platform (which among others provides ultimate freedom in further processing the data), I also offer means to register new services by extending a service contract.

To accommodate for all this requirements, I implemented the *PRIMI Advanced Sensor Suite* (*PASS*). *PASS* has three central elements the *Sensor Daemon* (*PASSDaemon*), the *Sensor Daemon GUI* (*PASSGUI*), and an extensive yet extendable collection of sensors.

The daemon process is started at the login and continuously runs in the background. The collection of sensor data can be activated and deactivated via inter-process communication (IPC). This is most comfortably done from the *PASSGUI*, which can be started as a separate application for configuring the daemon and the individual sensors. Besides offering general settings for the data collection by the sensor daemon, each sensor can be adjusted individually (e.g., quantisation level or sampling rate) via a preference panel (cf. Figure 37) in the *PASSGUI*. New sensors can be easily added dynamically via the plugin-mechanism. That is, by placing a new sensor module in a specific folder of the application in the file system and restarting the daemon (or GUI) process, the sensors are immediately available to the daemon. Each sensor is a self-contained module that can communicate via a standardised interface with the daemon and the GUI.

Finally, *PASS* differentiates between two basic sensor types: *Pull-* and *Push-Sensors* [Gross & Oemig 2006]. Each plugin needs to announce to *PASS* which type of sensor it represents. *Pull-Sensors* are often more basic sensors that monitor simple values. The reading of the sensor is triggered from outside, normally at a given sample rate. Therefore *Pull-Sensors* continually deliver sensor readings in a fixed interval. *Push-Sensors* on the other hand are often

more complex sensors, that already contain some lightweight inference. Therefore *Push-Sensors* deliver sensor readings at unsteady, variable intervals. Examples for *Pull-Sensors* are sensors that continuously read data from hardware like position or motion sensors (e.g., GPS, accelerometer) at a high frequency, constantly deliver the heart rate, or even at a slower pace deliver the number of unread Emails. *Push-Sensors* on the other hand often try to already detect changes in the data, and only send data when a state significantly changes. Examples are sensors that detect the connection or disconnection of some peripheral to the computer, infer from the microphone if a conversation started or ended, or if the users' computer activity changes to idle. Often *Push-Sensors* therefore only deliver a *true* or *false* value in respect to the specific state the sensor is monitoring.

The concept behind these two different representations is to reduce the amount of sensor data by using *Push-Sensors*. First, in cases where the observed state is not likely to change very often, and hence continuously sending the observations with a fixed sampling rate would lead to series of sensor events containing the same data. Second, when the sensed data is quite complex and extensive—as for example video or audio data—and therefore the analysis of the data locally and ad-hoc makes sense.

Hence, each sensor either contributes its data to a stream of sensor events in an interval-based (*Pull-Sensors*) or an event-driven (*Push-Sensors*) manner. The output of the *PASS* accordingly is a stream of data from multifarious sensors in form of sensor-events that arrive at the next processing step at fixed but differing intervals (*Pull-Sensors*) as well as in irregular patterns (*Push-Sensors)*.

5 LiLoLe—A Framework for Stream-based Active Learning

In the Following I introduce the LiLoLe Framework [Fetter & Gross 2014]. The focus of this chapter is the central building block of the LiLoLe Framework, the concept of Stream-based Active Learning. Further I provide details on the proof-of-concept implementation and highlight aspects of the validation of the Framework.

I have argued in 4.2 that machine learning is a promising approach for building personalised adaptive systems. Yet, I also mentioned the challenges involved with building such systems. One major challenge when building personalised models is the upfront need for a huge amount of learning examples in form of labelled data, to train the system before it can be utilised [Webb *et al.* 2001]. In order to maximise the utility of such an adaptive system for the users, of course the users' effort for providing such data should be kept minimal. From this perspective, two main approaches in user modelling can be distinguished [Zukerman & Albrecht 2001]: *content-based learning* and *collaborative learning*. Content-based learning utilises the users' individual past behaviour to predict their future behaviours. Collaborative learning uses the past behaviour of a group of similar users as the basis for predictions for an individual user. In both cases, the approaches rely on observing user behaviour. To provide an example: A music streaming service can easily observe what kind of music the users listen to in their clients. So either based on the meta-data of a song like artist, genre, beats per minute, etc. (content-based learning) or based on what other songs other similar users listen to (collaborative learning), the music streaming service can offer recommendation for new songs the user should listen to. The obstacle for both approaches is that the concept to be learned has to be observable for the machine for *in-stream supervision* [Kapoor & Horvitz 2007]. In the case of availability prediction however—as well as in many other cases in context-aware and ubiquitous systems (e.g., activity or mood recognition)—the concept to be learned is hidden for the machine. Hence, this information needs to be provided manually in form of labels.

The LiLoLe Framework—described in the Following— proposes an architecture that allows for continuous learning from sensor data streams for predictive user modelling. In this specific setting, the concept to be learned is the momentary preference for *Selective Availability* of a user. In this setting of learning personalised availability user models the following three earlier identified constraints demand for a novel approach:

- The individual availability in a given situation is a highly personalised construct and accordingly labelled data is sparse.

- Only the individual and only in the moment when it is experienced the data can be labelled—assigning labels later based on a presentation of collected sensor data is not feasible.
- As the labels, also the sensor data is highly personalised as it is grounded in the individual users' daily contexts, resulting in an individual evolving feature space based on a continuous sensor data stream.

5.1 A Rationale for Stream-based Active Learning

In the Following a Rationale for the utilisation of Stream-based Active Learning in this thesis is given juxtaposed with more conventional approaches to and concepts of machine learning as they are found in classic batch learning settings. While the whole architecture depends on several machine learning paradigms, here the two most central concepts—namely *Stream-Based Learning* (also Data Stream Mining) and *Active Learning* (also Query Learning)—are explained, discussed, and founded.

Mitchell [1997, p. 2] gives the following widely used, broad definition of machine-learning: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". For the last twenty-five years thereby machine learning was mostly concerned with batch learning [Gama 2008]. The premise of batch learning is, that the whole training data is available to a machine learning algorithm at a time. The algorithm processes this data and outputs a decision model, which then can be applied to new data. The underlying rationale is that the examples in the training data as well as the data the model is later applied to, are independent and identically distributed and therefore allow inferences on the new data. However, the premise that the training data is available upfront and representational for future data, may not always be given. In the Following I explore two settings in which the batch learning approaches fall short.

Data Stream Mining is a fast emerging field in the area of machine learning giving attention to the challenge of learning from a continuous flow of information. The basic premise is, that the development of information and communication technologies led us to a point were continually more and more data is generated. Sources for such data are for example highly distributed sensor networks in ubiquitous environments like digital factories or smart homes, network traffic in worldwide grids, or data on human activities that is collected with video cameras. Opposed to traditional data mining, the mining of data streams brings some unique characteristics that algorithms have to cope

with. These characteristics were identified in the *Data Stream Model* by Babcock *et al.* [2002], in the work of Bifet [2010, p. 5], Aggarwal [2007], and Gama [2008]:

- An extremely large (potentially infinite) amount of data continuously arrives, which is theoretically impossible to store it in its completeness. Accordingly, only small chunks can be processed and only summaries can be stored. Instead of multiple passes the data is processed in one-pass [Aggarwal 2007; Babcock *et al.* 2002; Bifet 2010; Gama 2008].
- This data arrives at great speed, making it necessary to process the items in real-time [Babcock *et al.* 2002; Bifet 2010; Gama 2008].
- The processing system has no control—and sometimes even no information— about the order in which the single elements are incoming. A coarse temporal order can be assumed, but is not guaranteed [Babcock *et al.* 2002; Bifet 2010].
- The concepts underlying the data can change over time, making data from the past partially irrelevant or even contradictory—a phenomenon referred to as concept drift [Widmer & Kubat 1996] or temporal locality [Aggarwal 2010].
- Finally, not only the concepts but also the data structures can change, as new data sources are added, or old ones are dismissed [Aggarwal 2007; Babcock *et al.* 2002; Bifet 2010; Gama 2008] features that were characteristic for specific classes can dissolve while others with more discriminative power can emerge.

Accordingly, the approaches for mining evolving data streams differ in many ways from the traditional batch learning settings, relying on new learning algorithms, which can as well classify as further adopt the model to incoming new data at the rate of data arrival and without accessing previous data [Gama 2008]. Further, while traditional batch learning departs from the assumption that the data has a static underlying probability distribution, in most real-world applications the underlying concepts are drifting, resulting in changing class-distributions over time, hence requiring algorithms that continually learn. And even beyond algorithms that are able to continuously learn from new data, also algorithms that are able to partially forget previous data are required, when the new data is not inline with the actual current distribution of the learned model. Further, those approaches also have to deal with an evolving feature space. That is, a feature space that is continually growing to infinite or at least unknown size—and accordingly—the entire feature set is not available at the beginning of the learning phase [Yu *et al.* 2015]. Consequently, also features can become irrelevant or redundant at one point in time, but may become relevant later [Gama 2008]. Finally, in classification tasks, also new classes might emerge over time and existing classes can loose their meaning and disappear [Mu *et al.* 2017] and sometime even reappear [Al-Khateeb *et al.* 2016].

Accordingly the requirements for data stream mining algorithm need to provide capabilities for incremental [Zhou & Chen 2002] and online learning, only requiring a single-pass over the training data. Such algorithms should offer constant and minimal processing time per example. They need to be able to detect and react to drifting concepts (e.g., through concept drift detection or decremental unlearning) and to adapt to an evolving feature space (e.g., through online feature selection) and emerging classes (e.g., through class-based ensembles).

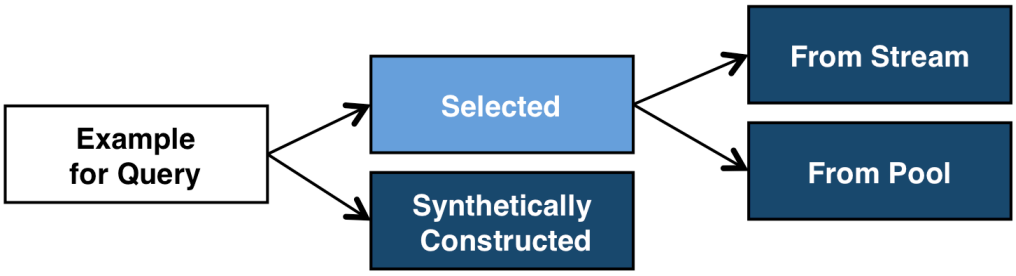

Figure 38. An illustration of the three active learning scenarios. The example for the query is either synthetically constructed or selected, either from the stream or a pool of examples.

Active learning [Cohn 2017; Cohn *et al.* 1994; Settles 2012] is also a machine learning approach that recently receives increased attention. It departs from the premise that obtaining labelled data for supervised learning is often very expensive. For example, in order to train a speech recognition system, it is quite easy to obtain unlabelled data of human speech (e.g., by recording the radio program or walking around with a microphone). However, to annotate this data with labels is a tedious and time-consuming task. The key idea behind *active learning* is according to Settles [2012, p. VI] that "a machine learning algorithm can perform better with less training if it is allowed to choose the data from which it learns". To accomplish this, an *active learning* algorithm selects examples from the unlabelled data set and presents those in form of a query to an oracle (e.g., a human annotator) to obtain labels for theses instances.

While the literature differentiates between three basic active learning scenarios (cf. Figure 38), all three scenarios include a step in which an algorithm decides whether or not a label for an unlabelled instance should be queried. Several ways for achieving this have been proposed, referred to as *active learning strategy* or query *strategy*. To give a brief understanding of what an *active learning strategy* can look like, in the Following three very basic examples are given. The first one will be referred to here as the Naïve Strategy. The second, called Random Strategy, and the third, called Fixed Uncertainty Strategy, are references to [Zliobaite *et al.* 2011].

Algorithm: Naïve Strategy **(I_t, B)**
Input: I_t - incoming instance, B - labelling budget.
Output: query ∈ {**true, false**} when true, query the true label $\mathbf{a}_t$.

return t mod 1/B = 0

Listing 7. Formal Description of Naïve Strategy (adapted based on [Zliobaite *et al.* 2011]).

The Naïve Strategy (cf. Listing 7) is the most basic and simplest strategy for requesting labels in an active learning setting with a determined labelling budget **B**. The labelling budget **B** thereby defines the portion of the incoming instances, for which a label can be requested from the human oracle, and accordingly can be seen as some form of a cost limit. A labelling budget of b = 1 would result in a label request for each incoming instance. A labelling budget of b = 0.1 would allow the active learner to request a label for 10% of the incoming instances. The Naïve Strategy simply requests a label for each $(1/b)^{th}$ incoming instance based on the available budget. For b = 0.05 a label would be requested for every 20^{th} incoming instance (e.g., $\mathbf{I}_{20}$, $\mathbf{I}_{40}$, $\mathbf{I}_{60}$, $\mathbf{I}_{80}$, ...).

Algorithm: Random Strategy **(I_t, B)**
Input: I_t - incoming instance, B - labelling budget.
Output: query ∈ {**true, false**} when true, query the true label $\mathbf{a}_t$.

generate a uniform random variable $\varepsilon_t \in [0,1]$
return $\varepsilon_t > B$

Listing 8. Formal Description of Random Strategy (based on [Zliobaite *et al.* 2011]).

The Random Strategy (cf. Listing 7)—also a baseline strategy with a naïve approach—randomly requests label from the human oracle based on a given labelling budget **B**. That is, the label is requested with a probability of **B**. For a budget of b = 0.05 the strategy would randomly select 5% of the instances, accordingly a possible sequence of instances to query could be (e.g., $\mathbf{I}_3$, $\mathbf{I}_{34}$, $\mathbf{I}_{42}$, $\mathbf{I}_{79}$, ...). Again, no knowledge about the data or learning process is used for choosing samples in this naïve approach.

Algorithm: Fixed Uncertainty Strategy **(I_t, θ, C)**
Input: I_t - incoming instance, **θ** - labelling threshold, **C** - trained classifier.
Output: query ∈ {true**,** false} when true, query the true label $\mathbf{a}_t$.

$\hat{y}_t = \arg\max_y P_C(y|I_t)$, where $y \in \{1, \ldots, c\}$ is one of the class labels.
return $P_C(\hat{y}_t|I_t) < \theta$

Listing 9. Formal Description of Fixed Uncertainty Strategy (based on [Zliobaite *et al.* 2011]).

Finally, the Fixed Uncertainty Strategy (cf. Listing 9) is the only of the three strategies mentioned here, that has a real active learning aspect to it and does not depart from a sole naïve assumption. The Fixed Uncertainty Strategy departs from the idea to request labels for instances that the classifier in its current trained state is the least confident about. The underlying algorithm works the following way: The classifier tries to classify the incoming instance. Based on the posterior probability distribution $\mathbf{P_C(y|I_t)}$, for this instance and a predefined, fixed threshold $\boldsymbol{\theta}$, the strategy actively decides whether a label should be requested from the human oracle or not. While the last approach is one of the most common [Zliobaite *et al.* 2011] active learning strategies, the labelling costs are not limited by a labelling budget, but only by manually determining a good threshold value upfront, that optimises between the labelling costs and the performance of the classifier.

From here, a lot of different strategies for deciding to query and for selecting the examples for querying are conceivable [Settles 2012]. Especially in respect to different goals, like detecting concept changes (i.e., concept drift, shift, or evolution), balancing prediction quality vs. labelling effort, etc. However, in the Following I want to move away from providing more details on different strategies, but focus on the combination of active learning and stream-based learning, the foundation of the LiLoLe Framework.

5.2 Stream-based Active Learning from Sensor Data Streams

In the beginning of this chapter I stated that for my setting collecting ground truth in form of labels is only possible in the moment when it is experienced and can be only provided by the individual user. By combining the approach of active learning with learning from sensor-data streams it becomes feasible to learn personalised user models for such ephermal, hidden concepts.

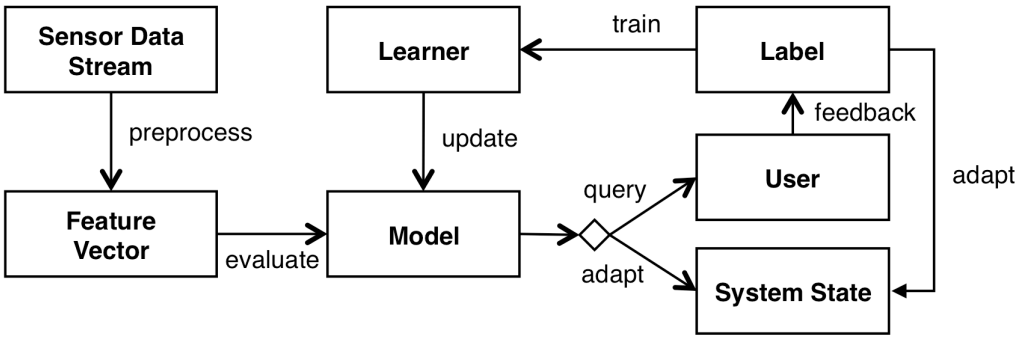

Figure 39. The basic principle of stream-based active learning from sensor data [Fetter & Gross 2014].

The basic principle of stream-based active learning from sensor data streams is outlined in Figure 39. In ubiquitous environments a plethora of sensors

captures data that allows inferences on a user's contexts, activities etc. This data is provided in form of a *sensor data stream*. The incoming data stream can now be *preprocessed* and transformed into *feature vector*s (also instances) that are processable by machine learning algorithms. Each feature vector is now evaluated against a predictive *model*. In the simplest case, this evaluation can happen by classifying the feature vector and checking whether the resulting classification confidence is above a specified threshold. Based on the outcome of this evaluation the system decides to either *adapt* the *system state* or to *query* an oracle (i.e., the *user*) to provide *feedback* in form of a *label*[1]. The label is used to *adapt* the *system state* but foremost to *train* a *learner* that will *update* the *model*. In Listing 10 the essence of this principle is shown in an algorithmic representation.

Algorithm: Embedded Stream-based Active Learning Approach

Input: strategy parameters
Output: o_t as output label for instance I_t
for each I_t - *incoming instance*, **do**
    if *ActiveLearningStrategy*(I_t,...) = **true then**
        request the true label a_t of instance It from human oracle
        train classifier C with (I_t; a_t)
        return true label a_t as output o_t
    else
        classify It with classifier C an obtain the predicted label p_t
        return classification result p_t as output o_t
    end if
end for

Listing 10. Basic principle underlying the Embedded Active Learning Approach.

Let $\mathbf{a}_t$ be the true label of the instance $\mathbf{I}_t$ where **t** indicates the time as the order of the arrival of the instances. Let an optimal data stream consist of instances arriving in sequential order ($\mathbf{I}_1$, $\mathbf{I}_2$, $\mathbf{I}_3$,...). For each incoming instance an *ActiveLearningStrategy* decides whether the system should ask for a label from the human oracle and train the classifier **C** or try to classify the incoming instance based on the current training-state of **C** to obtain the predicted label $\mathbf{p}_t$. The *embedded* active learning approach has an implementation perspective, where for each input instance $\mathbf{I}_t$ an associated label $\mathbf{o}_t$ is required as output. This is necessary in order to embed the algorithm into an application, where the output results in some form of adaptation of the application behaviour or state. Therefore in either case a label $\mathbf{o}_t$ is returned by the algorithm—either the true label $\mathbf{a}_t$ (as input by the human oracle) is passed through as output or the

[1] From the user interaction perspective, providing feedback in form of a label does not necessarily mean that the user needs to enter some text or select a label from a choice box. It is more likely that the user is asked via a dialogue to adapt the system state directly (e.g., adapting the availability status of the IM client), which is then used as a label.

predicted label $\mathbf{p}_t$. This way the algorithm can easily be embedded in an adaptive application.

While the previous algorithm showed how the approach of stream-based active learning could be integrated in an adaptive system, in Listing 11 the idea is extended towards a system that can deal with changes (i.e., concept shift, drift or evolution, or feature evolution). The embedded algorithm is extended with a mechanism, able to sense imminent and actual changes, and the ability to react to both. The change detection ideally would be designed as a second task of the *ActiveLearningStrategy*. If a change is imminent, a change warning is signalled and the algorithm starts training a new classifier $\mathbf{C}_n$. The new classifier $\mathbf{C}_n$ might for example use a different subset of features, including newly emerged features. In subsequent runs, the $\mathbf{C}_n$ classifier is trained whenever the *ActiveLearningStrategy* queries a new true label $\mathbf{a}_t$ from the user. The classifier receives the label $\mathbf{a}_t$ together with an instance $\mathbf{I}_{t'}$ - transformed into the feature space of $\mathbf{C}_n$. Alternatively $\mathbf{C}_n$ can also be co-trained [Blum & Mitchell 1998] based on the predicted label $\mathbf{p}_t$. At the time, the *ActiveLearningStrategy* is detecting an actual change (e.g., when $\mathbf{C}_n$ is performing better then $\mathbf{C}$), the algorithm replaces the old classifier $\mathbf{C}$ with then newly trained $\mathbf{C}_n$. The illustration of the LILOLE Framework in Figure 40 reflects this idea, were the model updating section of Figure 39 is extended with different approaches for handling change, including Ensemble Classifiers, Attribute Bagging, and Co-Training.

Algorithm: Embedded Stream-based Active Learning Approach with Change Detection

```
Input: strategy parameters
for each I_t - incoming instance, do
    if ActiveLearningStrategy(I_t,...) = true then
        request the true label a_t of instance I_t
        train classifier C with (I_t; a_t)
        if C_n exists then
            train classifier C_n with (I_t'; a_t)
        end
        if change warning is signalled then
            start a new classifier C_n++
        end
        if change is detected then
            replace classifier C with C_n
        end
    else //do co-training
        if C_n exists then
            train classifier C_n with (I_t'; p_t)
        end
    end
end
```

Listing 11. Embedded Stream-based Active Learning Approach with Change Detection.

Figure 40 also shows a more detailed view of the preprocessing steps that transforms the continuous stream of sensor data in a sequential flow of feature vectors. In order to be applicable in many settings (e.g., for nomadic users), the underlying requirement is that the preprocessing needs to work in an opportunistic matter where "no predefined sensor infrastructure has to be defined at design time" [Kurz & Ferscha 2010, p. 136] hence making it necessary to create a system that is able to react to "changes in the sensor network topology at runtime." [Kurz & Ferscha 2010, p. 136].

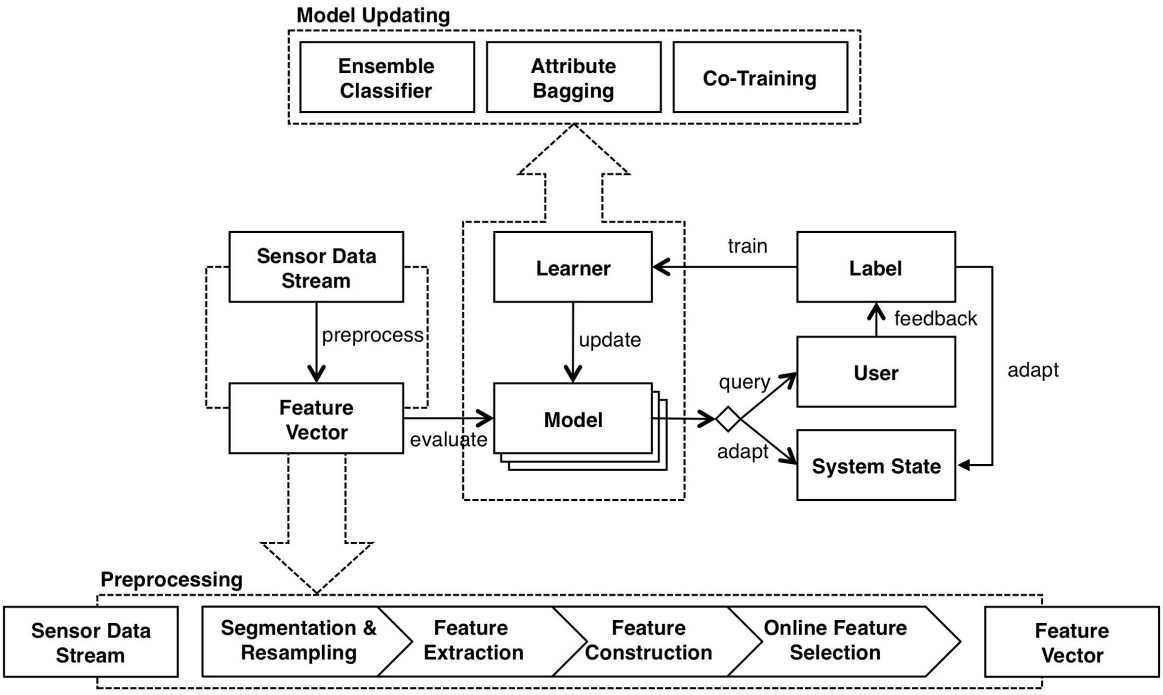

Figure 40. Overview of the LiLoLe Framework [Fetter & Gross 2014].

The whole process depicted in Figure 40 is shown in a more formalised, algorithmic representation in Listing 12.nd is described in the Following in more detail based on the concrete example of adapting *selective availability*. The subsequent description of the process relies on examples based on the study data as well as the proof-of-concept [Bardram & Friday 2009] implementation that was built in order to test the technical feasibility of the approach. Each of the steps needed to transform the incoming sensor events into an adaptation decision is now successively explained. The description is divided into four bigger steps, each aligned to the implementation in form of Inference Engines (IE) as well as Sensors and Actuators for *Sens-ation*.

Algorithm: Embedded Stream-based Active Learning Approach for LILOLE

Input: strategy parameters, strategies, sensor stream
Output: o_t as output label for instance I_t
for each s_t - *incoming SensorEvent*, **do**
  resample s_t to discrete timestamps t' and add to resampled data stream
  segment resampled data stream into segments $S_{t'}$=[$s_{t'}$, $s_{t'-1}$, $s_{t'-2,...}$ $s_{t'-w}$]
  for each *FeatureConstructionStrategy(...)* **do**
    for each *FeatureExtractionStrategy(...)* **do**
      constructFeatures(extractFeatures($S_{t'}$))
    end for
  end for
  build $I_{t'}$ - *instance* from Features
  if *FeatureSelectionStrategy* ($I_{t'}$,...) = **true then**
    start a new classifier C_n with selected features
  end if
  for each $I_{t'}$ - *incoming instance*, **do**
    if *ActiveLearningStrategy*($I_{t'}$,...) = **true then**
      query the true label $a_{t'}$ of instance $I_{t'}$ from the user
      train ensemble E with ($I_{t'}$; $a_{t'}$)
      return true label at' as output $o_{t'}$
    else
      classify $I_{t'}$ with ensemble E and obtain the predicted label $p_{t'}$
      return classification result $p_{t'}$ as output $o_{t'}$
    end if
    if C_n *exists* **then**
      co-train classifier C_n with $o_{t'}$
      if C_n *is trained* **then**
        add classifier C_n to Ensemble E
      end if
    end if
    adapt the system based on $o_{t'}$
  end for
end for

Listing 12. Algorithmic representation of the LILOLE Framework process.

5.2.1 Step One—Temporal Discretisation: Resampling and Segmentation

With a subsystem in place for collecting sensor data (compare 4.4) we receive a continuous stream of sensor data, in form of a series of repeated measures from multiple sensors (cf. Figure 40). Each measure is received in form of a sensor event including meta-data (e.g., a time-stamp). Hence, a sensor data stream basically represents a multivariate time series (MTS). A MTS is representing the observations of several variables **i** over a time **T** storing each observation at different points in time **t**. As the data from *Push-* and *Pull-Sensors* arrives with mixed sampling rates [Cachucho *et al.* 2014], the first step is to achieve some form of temporal discretisation [Moskovitch & Shahar 2015]. That is, to resample and segment the incoming sensor events to discrete timestamps. To provide an example: in the sensor data I obtained in the study, the *WiFi* sensor is a *Pull-Sensor*, collecting information on the network infrastructure every 150

seconds (sampling rate of 0,004 Hz). The *Input Idle* sensor—a *Push-Sensor*—only sends an event when a change occurred (i.e., the user stopped interacting with mouse and keyboard for a fixed time or started again after an idle period). Accordingly, there is a very high likelihood for each point in time, that only data from one of the two sensors is available (cf. Table 12).

Table 12. This example shows a segment of the sensor data stream as it could appear in the study data. The table is showing the times when values are available from the interval-based *WiFi sensor*—with a sampling of 0.004 Hz) and the event-driven *Input Idle sensor* (✓ denotes an available sensor event at a given time t_n).

	t_1 (0s)	t_2 (11s)	t_3 (150)	t_4(300)	t_7(421s)
WiFi sensor	✓	-	✓	✓	-
Input Idle sensor	-	✓	-	-	✓

As this out of phase data makes the data processing more difficult, the data is resampled to discrete time points resulting in a MTS with a fixed sample rate. This can either happen by upsampling the data (e.g., through interpolating the missing values) or downsampling the data (e.g., by calculating averages over a sliding window). The right strategy as well as the parameters needs to be fine-tuned for the concrete application [Cachucho *et al.* 2014]. In the proof-of-concept implementation of the LILOLE Framework I applied a naïve resampling strategy, called *LastValueStrategy* that simply stores the last value for each sensor. Each time a new sensor event comes in for a sensor, the *LastValueStrategy* algorithm would replace the value for this sensor in its internal storage. The algorithm then pushes all stored values in form of a set of sensor events to the next processing stage. A push-strategy-parameter allows configuring whether the set should for example be passed each time a new sensor value is received (*NewValuePush*) or based on a fixed time interval (*StaticTimerPush*). The result is an evenly (*StaticTimerPush*) or unevenly (*NewValuePush*) spaced times series, which at each observation time has a value for all sensors. For the variety of data types in our raw sensor data (i.e., string, number, and Boolean values) and the added complexity by the event-driven *Push-Sensors,* this was a straightforward approach, as the main idea of the implementation was to show the general feasibility and not aim for the best possible prediction results.

As a result of the resampling—foremost when upsampling to the sensor with the highest sampling rate or even above this rate with *NewValuePush strategy*—the amount of data to process strongly increases.

As the sensor data stream continually delivers new events, the resulting MTS is infinite and therefore cannot be processed in its whole. Hence, the next processing stage is to cut the now temporal discretised time series into even segments of finite length for further processing. Again, depending on the application and the expected data, many different approaches for segmenting

time-series [Keogh *et al.* 2003] are conceivable. In the feasibility implementation I utilised a simple fixed-size sliding window (*FixedWindowStrategy*) for the segmentation, with a configurable window size and overlap offset. Each segment is passed to the next step as a batch of sensor events.

5.2.2 Step Two—Feature Engineering: Online Feature Extraction, Construction and Selection

The resampled and segmented data is now ready to be processed by the feature engineering chain. As I already explained the three parts of the feature-engineering step (i.e., feature extraction, construction and selection) in some detail in section 4.1.2, the following section concentrates on the specifics of transferring the previously in batch and manually performed mechanisms into an automatic process for streaming data. That is, the whole feature engineering process needs to be done with online algorithms.

Online Feature Extraction: The first task—the *Feature Extraction* [Liu & Motoda 1998]—transforms each incoming segment into an instance. That is, an example that can be supplied to the predictive model to be used for classification or further training. This foremost transforms the data into simpler data types that can be typically processed by machine learning algorithms (i.e., numeric and nominal types [Witten *et al.* 2011a]) and flattens more complex sensors to simple *key-value* pairs of attributes (i.e., features).

For my implementation I reused many of the feature extraction mechanisms implemented for the batch processing (as described in 4.1.2). A single Inference Engine (IE) mapped the complexity of 1×1, 1×m, n×1, and n×m sensor events to the Weka's Instance format. From top down this means first transforming all 1×m and n×1 into 1×1 representations, as well as building all n possible 1×1 permutations for each of the m rows of an n×m sensor. The resulting set is completely flattened and only includes simple *key-value* pairs. Each *key* is automatically constructed from the sensor name as well as the '*item'* name from the sensor event (as was shown in 4.1.2.1 for the WiFi sensor: `WiFi.SSID.Uni-Bamberg802.1X.RSSI=179`). Numeric and Boolean values can now easily be transformed into Weka attributes, using the *key* as the name of the attribute. For nominal values (e.g., the name of the application in focus like `ApplicationFocus.AppName="Firefox"` or the SSID of a wireless network like `WiFi.SSID="Uni-Bamberg802.1X"`) the IE performs one additional processing step. Comparable to the *NominalToBinary* filter (cf. [Witten *et al.* 2011a, p. 439]) of the Weka Toolkit the inference engine generates and maintains a list of the length **l** for each key with nominal values (e.g., `ApplicationFocus`) with entries of all unique values it has seen so far (e.g., `Firefox, Finder, Terminal, Word`). If this algorithm processes a new nominal key-value pair with the same key name, it will hand back **l** new Boolean

attributes—one for each entry in the list, with all but one set to false (compare to 4.1.2.1).

As we can see, with this approach the feature space (i.e., the number of attributes) is constantly growing. Potentially with each newly arriving piece of sensor data—if the user for example used a new application for the first time, or logged into a network with an unknown SSID—new attributes emerge. Yet, for learning algorithms in general—and especially for their implementations in toolkits like Weka—the number of features needs to be constant. Hence, an additional task of this IE is to deal with this problem from an implementation perspective. More details on how this evolving feature space is handled technically, is given later.

One further, simple *FeatureExtractionStrategy* I implemented is the previously used (compare 4.1.2.1) list length feature for all n×1 and n×m, as well as the approach for calculating temporal features from the meta information for hour of the day, part of day, day of the week and weekend. The result are additional features like `Applications.ListLength = 34` or `DayOfWeek.mon = true`.

Online Feature Construction: As the data arrives in form of a time-series, the now extracted feature vector more or less only represents the very moment. A next logical task accordingly is to also take into account the recent history of sensor data with the means of *Feature Construction* [Liu & Motoda 1998]. The respective IE therefore maintains a history of the last 15 minutes for all features that were built in the *Feature Extraction* task. For each numeric and Boolean feature, three new features are calculated, taking into account the last 2, 5, and 15 minutes. For numeric features a simple mean value is calculated. For Boolean values the fraction of time the feature was true is computed. Possible features accordingly are `WiFi.SSID.eduroam.RSSI.2Min = 165` or `InputIdle.15Min=0.23`.

Online Feature Selection: The final task in the feature-engineering step is the *Feature Selection,* to reduce the dimensionality of the feature vector. While the first two tasks could be mostly automated straightforward, the task of *Online Feature Selection* (*OFS*) is less trivial—especially the more effective wrapper-based approaches. As we learned before, the aim is to reduce the number of features by selecting only the features with the overall most predictive power, as most classifiers perform better with a low dimensional feature space. In my case this is especially true, as the feature engineering step is automated and the resulting feature vector is not hand crafted by a domain expert. Hence, the result is a high dimensional feature space with many irrelevant and redundant features, which need to be removed in the feature selection process.

In batch supervised learning, the feature selection process starts with all possible features available upfront and is often based on a number of labelled

examples—the training data—which can be easily be used to evaluate the chosen subset in a wrapper-based approach. In my approach of stream-based active learning these two prerequisites are not met, hence special algorithms for *OFS* [Perkins & Theiler 2003; Wang *et al.* 2013; Wu *et al.* 2013; Wu *et al.* 2010] are needed. However, the research on algorithms for OFS is a relatively new field in machine learning. Accordingly, at the time of the implementation, no of out-of-the-box implementations of OFS algorithms for standard machine learning toolkits like Weka existed. Only recently a few implementations for the toolkit MOA [Ramorez-Gallego *et al.* 2017] and MATLAB [Yu *et al.* 2016] became available. As the focus of the implementation was to evaluate the end-to-end feasibility of the stream-based active learning approach, I combined a filter-based strategy and a naïve random strategy for selecting features. First, I simply applied the *RemoveUseless* filter, to remove all constant features, and those with too much variance. Afterwards a fixed percentage of the remaining features were randomly selected. While being aware that this implementation degrades the machine learning performance, the focus of this implementation is more on showing a complete and working implementation of the Framework, than reaching a high accuracy.

As with each single task in the feature-engineering step, also here the outcome is an evolving feature vector that potentially changes each time, new data arrives. The *FeatureSelectionStrategy* therefore also needs to include a mechanism that handles when and how to switch to a newly composed feature vector and accordingly to a new or updated model. Therefore, the implementation in form of a *Sens-ation* IE does two things:

First, it maintains one general representation of the feature vector including all features that were ever constructed and extracted in the feature-engineering step. For this *superset* feature vector the IE also maintains a history of the processed instances in form of a buffer. When a new feature is added to this feature vector, the IE also updates the buffer, which includes adding the new feature and filling up older instances with missing values for this feature. Second, it maintains representations of all feature vector *subsets*, which were used for an actual model. As each of these *subset* feature vectors only contain features that are elements of the *superset feature vector*, new arriving instances that are including all features of the *superset* can be converted to these *subset* representations on the fly through filtering.

In the current implementation, the IE only passes the instance with the most current feature subset to the next IE, which is concerned with the active learning step.

5.2.3 Step Three—Active Learning: Classification and Training

After the first two steps of Temporal Discretisation and Feature Engineering—both part of preprocessing the sensor data stream into a feature vector representation—the third step is concerned with *Active Learning* from the data. While from a conceptual perspective (cf. Figure 40) it involves a set of diverse tasks, from an implementation perspective, it more or less revolves around the Model as a central data structure. Accordingly the classification and training is done in one IE.

This IE constantly receives input in form of a single instance with a certain number of features. The interval and rhythm is determined by the update frequency of the sensors and the Temporal Discretisation step. The task of this IE is manifold: Initially, it needs to decide whether to adapt the system (i.e., the availability state) or whether to query for a label. Depending on this decision the IE either needs to propagate the predicted adaptation decision to set the correct system state, or query the user and handle the result. Handling the result involves updating the model through training and ideally, also setting the system state according to the query result. Based on the requirements it is obvious, that only incremental updatable classifier algorithms can be used. Examples for such algorithms would be, Hoeffding trees [Hulten *et al.* 2001] or Very Fast Decision Trees (VFDT) [Yang *et al.* 2012], online variants of kernel learners like LASVM [Bordes *et al.* 2005] as well as more simple algorithms like an updateable variant of a Naïve Bayes [John & Langley 1995] or the instance-based K* classifier [Cleary & Trigg 1995].

Finally, this step also needs to handle all aspects of the evolving data stream, with the most central prerequisite in my approach: the ability to deal with feature evolution. As most learning algorithms cannot deal with instances represented by a changing combination of features—i.e., different feature subsets—and the likelihood that an ideal and stable feature subset will not emerge at the very beginning, I propose three mechanisms for model updating in the LiLoLe Framework to deal with this issue (cf. Figure 40). All three mechanisms are based on the underlying assumption that starting a new model from scratch, and discarding all previous knowledge is not ideal. Hence, all three mechanisms have in common that they aim to combine new emerging models with older models. The suggested mechanisms are: Ensemble Classifier, Attribute Bagging, and Co-Training.

First, in the area of machine learning, ensemble classifiers (or more general ensemble learning) [Al-Khateeb *et al.* 2016; Bryll *et al.* 2003; Rokach 2010; Wang *et al.* 2003] in essence departs from the idea that a better predictive performance can be achieved, by combining a set of different models. The models are combined by different strategies (e.g., bagging or boosting), where each strategy determines how the classification result of each individual model should be

combined to reach an optimal overall classification. For example, it has been shown that ensemble approaches deliver superior result when learning from concept drifting data streams [Al-Khateeb *et al.* 2016; Rokach 2010; Wang *et al.* 2003]. While ensemble methods were used in batch learning settings, to improve the classification results by combining several models per se, they can clearly also be applied in an adapted manner, to combine older and newer models with some form of weighting or voting.

As a second approach, I suggests a specialised form of ensemble methods, namely attribute bagging [Bryll *et al.* 2003] (also feature subset ensembles or random subspace method [Ho 1998]). While the original purpose of this method again is to combine different models—here models based on different subsets of the feature vector—to optimise the overall classification performance, it also could be repurposed to deal with the challenges of an evolving feature space. That is, each time the feature subset is evolving, a new model is generated for the new features subset and added to the ensemble. In both cases (i.e., for ensemble classifiers and attribute bagging) a challenge is that each time a new model is added, at the beginning the model is untrained.

To overcome this issue, I propose a third mechanism, a phase of co-training [Blum & Mitchell 1998]. That is, the new model would be trained for a certain time, not only based on the answers to the active learning queries, but also based on the classification output of the current ensemble. This would allow training a model based on a new feature subset, until a certain training criterion is met and the model can be added to the ensemble and actively used for predictions. However, none of the three mechanisms was used in the proof-of-concept implementation, as the effort for implementing them with the possibilities of the Weka Toolkit was too high and beyond the scope of simply demonstrating the technical feasibility of a stream-based active learning system.

Hence, the implementation in form of an IE, focuses on the more central aspect of the framework, which is active learning with an updatable classifier. To provide a bit of flexibility to select a classification algorithm and adapt its settings, the IE accepts Weka command-line strings as a configuration parameter for specifying the classifier used in the IE. The IE accepts all Weka classifiers that implement the Interface *UpdateableClassifier* as parameter.

For the *ActiveLearningStrategy*, the IE currently realises two alternative approaches. The first, named *FixedUncertaintyStrategy*, allows to set a minimum and maximum threshold for the confidence value (i.e., in Weka the probability distribution for the predicted class). As described in [Zliobaite *et al.* 2011], if the confidence value for the classification is below the minimum threshold, the active learning strategy decides to query the user for a label, as the algorithms seems unsure about the prediction. Additionally, I also introduce a maximum threshold. If the classifier is too confident, the strategy also requests a label, to

prevent over fitting the model. If the confidence value is between minimum and maximum, the system performs the adaptation.

As an alternative strategy—an explorative *ClusterStrategy*—based on the COWEB [Fisher 1987] clustering algorithm was also implemented. The idea of this strategy is to detect and react to changes in the underlying data. If the strategy detects such changes, as new clusters emerge from the sensor data, it also queries for a label.

5.2.4 Step Four—Querying and Adapting

The last step—although from a purely temporal perspective it is tightly interwoven with the previous step—is the interaction with the user. This step focuses on keeping the human-in-the-loop. This happens in two ways: First, to improve the model the active learning algorithms queries the user (i.e., the *oracle)* for a label. Hence, requesting some sort of reaction from the user. Second, also the adaptation is affecting the user. By changing the state of the system, the adaptation impacts the current and future interaction of the user.

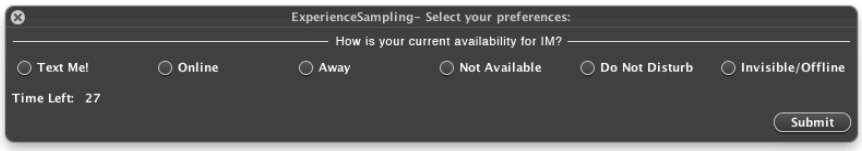

Figure 41. Screenshot of an exemplary query dialogue for the availability scenario as it would be generated by Listing 13 [Fetter & Gross 2014].

When focussing on the query, it seems logically that the user in someway must be actively notified about the query and needs to be provided with a possibility for answering the query. While many modalities are conceivable, depending on the application, here I focus on the two possibilities offered in the prototypical implementation.

The first possibility is rather explicit and direct via a simple dialogue. In a GUI based system, it is possible to simply present the query direct to the user in form of a dialogue, together with some sort of form to answer the question. For a classification task, this could be a simple presentation of options as menu or radio-buttons. The dialogue then very much resamples the dialogue I used in the ESM study [Fetter *et al.* 2011b]. In the implementation of this step a combination of actuator and sensor is used for presenting the dialogue (cf. Figure 41). The dialogue can be easily configured via an XML-formatted String (cf. Listing 13). The IE of the previous step triggers the actuator to present the query. The answer to the query flows back as a sensor event to the IE, which uses it for training. The XML configuration more over offers the possibility to set a time-out for the dialogue (in seconds), as the user normally is required to answer the query very promptly. Otherwise, the label looses its meaning for the

current instance of sensor data, that triggered the query. Additionally, an interval can be specified that defines a minimum time (in minutes) that needs to pass before a new dialogue can be presented. With this pacing mechanism it is possible to regulate how often the system is allowed to query the user during the day. In both cases, whether a time-out occurs or the actuator did not present the dialogue because of the interval, an empty answer is returned to the IE.

```
<esmconfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation='esmconfig.xsd'>
    <sid>ExperienceSampling</sid>
    <requesttimeout>30</requesttimeout>
    <requestinterval>33</requestinterval>
    <question>
        <query>Available</query>
        <description>Available Status</description>
        <single>
          <value code="0">Text Me!</value>
          <value code="1">Online</value>
          <value code="2">Away</value>
          <value code="3">Not Available</value>
          <value code="4">Do Not Disturb</value>
          <value code="5">Invisible/Offline</value>
        </single>
    </question>
</esmconfig>
```

Listing 13. XML example for specifying the query-dialogue

The second possibility is a little less direct and more implicit. The idea is to present a notification to the user, to manually update the system state of the application, that is the target of the adaptation. When the users adapts the system, the selected state is also send to the active learning algorithm as label. For the proof-of-concept implementation I integrated this concept into the PRIMI Mundane client. PRIMI Mundane is a lightweight IM client based on the PRIMI infrastructure [Gross & Oemig 2005a] that supports *Selective Availability*—normally via manual adaptation—and allows specifying individual *Availability Categories*. That is, users can create and label individual categories (e.g., *"soccer team"*), add contacts to this category, and then regulate for this group of users manually the availability.

In the proof-of-concept implementation, now the user is queried via a desktop notification to adapt the availability states of all availability categories in PRIMI Mundane. The according changes are sent to the IE as label information for training a binary model (i.e., *available* or *unavailable*) for the respective category. In a second, simpler, and explorative variant of this implicit approach I realised for the proof-of-concept, the users are asked via a notification to adapt the volume settings of their computer to the currently preferred level. As the volume sensor easily can observe manual changes to the volume, the

implementation maps the absolute volume settings to the three labels *soft*, *medium*, and *loud*. For both implemented variants of the implicit approach, it was also possible to specify an interval and a time-out, similar to the dialogue solution.

In terms of adaptation, it should be clear that for the last two examples (PRIMI Mundane and the volume setting) the active learning algorithm instead of sending a query logically also would adapt the system state based on the classification result— that is, if the *ActiveLearningStrategy* is confident that the prediction is correct. For the ESM-like dialogue, there was no adaptation made, but the adaptation decision were simply stored in a log-file. This is the case, as the dialogue was mainly implemented to use it for further studies based on a real, stream-based active learning system. The volume example on the other hand, was used as a nice demonstrator for the whole system, when combined with two or three easily manipulable sensors.

5.3 Proof-of-Concept Implementation of the Framework

In the Following, a basic overview of the architecture of the proof-of-concept implementation is given, including a detailed look on some of the individual components. The aim is primarily to provide a basic understanding of the technical details, by offering a coarse overview of the interaction of the individual components. For each component, I additionally provide insights in how specific aspects were implemented, as far as they are not already covered in the conceptual description before.

5.3.1 Overview of the Software Architecture

The overall architecture follows the general layout of context-aware and *Ubiquitous Computing* environments, consisting of components for input in form of sensors, components for processing in form of inference engines, and components for output in form of actuators [Bardram & Friday 2009; Dey *et al.* 2001; Gross *et al.* 2006; Salber *et al.* 1999].

Therefore I rely on *Sens-ation* (developed by Gross, Egla, and Marquardt [2006]) as a central mediator, for orchestrating the data flow between sensors, inference engines, and actuators. As a service-oriented middleware for distributed *Ubiquitous Computing* environments, Sens-ation is able to handle the input from a multitude of distributed sensors. Sensor implementations can connect to Sens-ation via various ways, as for example via XML-RPC, which is also primarily used in this work for remote communication. Sensors can register and unregister themselves to the platform and send sensed data in form of `SensorEvents` to *Sens-ation*. *Sens-ation* also functions as a run-time environment for built-in and custom `InferenceEngine` implementations

and provides means to trigger distributed actuator components that are registered at the middleware. While in the largest part I relied on the existing *Sens-ation* implementation, I also partially extended its capabilities for example with the possibility to configure, instantiate, and connect inference engines at run-time and by introducing the *SensVal*-format. I provide details on both extensions later in this chapter.

An additional third-party component in this architecture is the Weka library (in version 3.5.7) [Hall *et al.* 2009], which is used for its Java implementations of various machine learning algorithms. The library is accordingly mainly utilised for building the feature vector and instance representations of the data and for the training and classification.

An overview of all systems, subsystems and components is given in the diagram in Figure 42. In the centre of the diagram is the proof-of-concept implementation of the *LiLoLeServer*. It uses *Sens-ation* as platform and service for running the custom implementations of `ConfigurableInference Engine`, namely the *IETemporalDiscretisation*, *IEFeatureEngineering*, *IEActive Learning*. The chain of these three IEs fulfils the tasks described in detail in the previous chapter and utilises the Weka library for its work.

The *LiLoLeServer* receives input in form of sensor events from the currently 29 sensors plugins in the *PRIMIAdvancedSensorSuite*. The *PASSDaemon* offers a run-time environment for `SensorPlugin` implementations and dispatches the sensed information to *Sens-ation*'s *SensorPort* via XML-RPC. The *PASSGUI* allows users to configure the *PASSDaemon* as well as individual sensors.

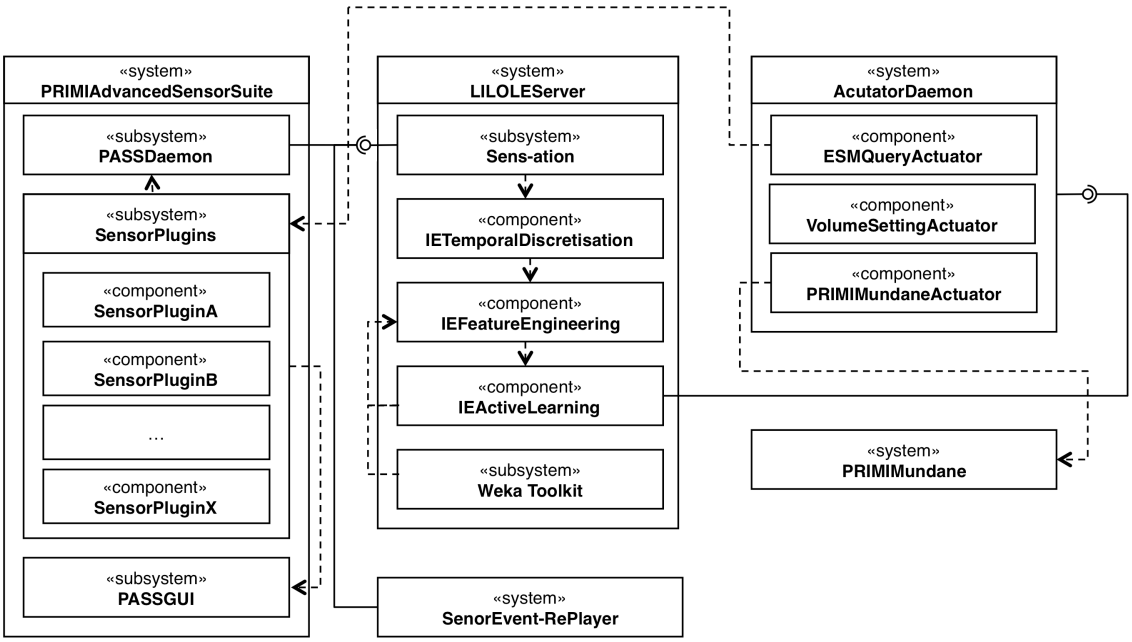

Figure 42. An overview of the main systems, subsystems, and components of the proof-of concept implementation.

The incoming `SensorEvent`s from all Sensors are routed through the three IEs, where they are gradually transformed and in the end passed to the *ActuatorDaemon*. The *ActuatorDaemon*—much alike the *PASSDaemon*—is a central run-time and dispatching system, that runs on a client-computer for routing commands to individual Actuator components. In the specific case of the proof-of-concept implementation these three actuators are the *ESMQueryActuator*, the *VolumeSettingsActuator*, and the *PRIMIMundane Actuator*.

The *SensorEvent-RePlayer* is a stand-alone Java Swing application that substitutes the *PASSDaemon* in simulations. *Sens-ation* receives the sensor events in the exact same way from the *SensorEvent-RePlayer* as from the *PASSDaemon*—hence live and simulated sensor data are indistinguishable for *Sens-ation*.

In the Following, I provide more details on the individual building blocks of the architecture in the order of processing.

5.3.2 Implementation of the PRIMI Advanced Sensor Suite

Collecting and providing sensor data is the foundation for context-aware applications (see 4.4). In the proof-of-concept implementation, the *PRIMI Advanced Sensor Suite* (*PASS*) carries out this basic task. The *PASS* consists of three parts: the *PASSDaemon*, the *PASSGUI*, and an extendable collection of Sensor Plugins. The smallest building block however is the sensor data produced by the sensors, and hence will be in the focus before the three parts are discussed.

In the *PRIMI Advanced Sensor Suite* I established the `SensVal` format for *Sens-ation*[1], to allow the comfortable and generic storage and transmission of more complex sensor data. Each time the sensors reads information from the environment, the collected data of this sensor event is converted into a `SensVal`. The *SensVal* format (cf. Listing 14) is designed as an extensible sensor data format. It allows computing and processing information that is collected by different types of sensors in a flexible manner. Further the data can be easily stored through its XML representation. It also can be easily wrapped inside the *Sens-ation* specific `SensorEvent` as a String value and thus can be easily processed by *Sens-ation*. As we briefly learned in 4.1.1, the information collected by one specific sensor can either be a single datum or consist of multiples pieces of information. Accordingly, the *SensVal* schema accounts for this by storing the data in a nested structure of `row`- and `item`-elements

---

[1] The `SensorEvent` implementation is *Sens-ation* did only foresee events with one single value and the data types Integer, Float, and String. More complex sensor data can be expressed via the *Sens-ation* concept of relationships and ingredients, but would have introduced an immense processing overhead.

Thereby each `item` stores one concrete datum with a label and data-type (either String, Integer, Float or Boolean) as attributes. Based on how the data is structured, we can distinguish between 1×1-, 1×m-, n×1-, and n×m-sensors. From an implementation perspective, the structure of a single sensor thereby is the result of one of four reasons:

```
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="sensVal">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="row" nillable="true" minOccurs="0"
             maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="item" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="type"
                         type="xs:string"/>
                      <xs:attribute name="label"
                         type="xs:string"/>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="date" type="xs:dateTime"/>
      <xs:attribute name="sid" type="xs:string"/>
      <xs:attribute name="dimensionality">
        <xs:simpleType>
          <xs:restriction base="xs:string">
           <xs:enumeration value="1x1"/>
           <xs:enumeration value="1xm"/>
           <xs:enumeration value="nx1"/>
           <xs:enumeration value="nxm"/>
          </xs:restriction>
        </xs:simpleType>
       </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing 14. XML Schema of the *SensVal*-format.

- First, the construct measured by the sensor calls for a specific structure. For example, the 3D physical motion as measured by the hardware accelerometer is represented as acceleration in x-, y-, and z-

direction. For the sensor it made sense to report this data together and thus implement it as a 1×m-sensor.

- Second, many sensed properties are in form of a list of repeated and reoccurring items of a similar type—like the list of names of the currently connected USB devices—it made sense to express those as a list of variable length. Hence, the n×1-sensor was chosen for such data. Much the same, when several items belonging together (like for example the SSID, BSSID, and RSSI of a WiFi-access point) repeatedly occurred, a table-like structure was chosen—i.e., the n×m-sensor.
- Third, in some cases the way the sensor was implemented was the determining factor for choosing the structure. For some sensors—for example such, that checked the state of a hardware component—it made sense from a resource perspective, to report the data together, often in form of a 1×m-sensor.
- And finally, sometimes a sensor is implemented to measure a couple of values that semantically belong together, like different figures and characteristics of the state of the email application. Such structures are the result of an implementation decision for the specific sensor—and not so much established structures like the x-, y-, and z-acceleration—and were also stored in form of a 1×m-sensor.

Besides the collected sensor data, the *SensVal* format also allows to store meta-data about the sensor producing the sensor event, namely the sensor ID (sID) as well as the date and the time of the sensor event. This self-contained format allowed the easy logging of sensor information in the *PRIMI Advanced Sensor Suite*, but also allowed to maintain the compatibility with *Sens-ation*.

5.3.2.1 Implementation of Sensor Plugins

The next building block is the implementation of the individual sensors. For the sensors, *PASS* reuses a small part of the *PRIMI* infrastructure [Gross & Oemig 2005a; Gross & Oemig 2006] as departing point. *PRIMI*—the *Platform for Research on Instant Messaging Infrastructures*—uses an elegant Plugin mechanism to allow easy reconfiguration by adding or removing plugins. *PASS* reuses the *PRIMI* Java interfaces `Plugin` and `Sensor`, and supplements them with new capabilities. By implementing the `Plugin` interface, a sensor can be loaded at runtime, and provide a `Sensor` implementation via the Plugins `getComponent()` method. By implementing the `Plugin` and `Sensor` interface, each sensor realisation can be easily packaged into one self-contained jar-file, which can be dynamically detected and loaded by *PASS* at run-time through the Java ClassLoader mechanism.

As we learned before, *PASS* distinguishes between two main sensor types: *Pull-* and *Push-Sensors*. The `getType()` method of the `Sensor` interface allows

to infer the type for a concrete sensor implementation. Depending on the type, PASS is now able to obtain new sensor readings by actively calling the `Sensor`'s `getValue()` method (for *Pull-Sensors*) in a fixed interval; or by registering via a publish-subscribe mechanism as a `SensorListener` using the `addSensorListener(SensorListener sl)` method (for *Push-Sensors)*. A *Push-Sensor's* main functionally is typically implemented in a continuously running Thread, that informs registered listeners via their implementation of the `notify(SensorMessage message)`.

While there is some more functionality in the `Sensor` interface, I reduce my description here to only one further function, which is of importance later. Each sensor realisation also provides its own means to allow the configuration of sensor specific settings. Therefore each `Sensor` implementation offers a configuration dialogue in form of a `JComponent` that can be requested with the `getPreferencesPane()` method. The `JComponent` needs to be implemented in a self-contained manner that it can remember (i.e., store and load) the sensor specific settings. For all 29 implemented sensors this storage is achieved via the *Java Preference API*, that allows to save the sensor settings in the default location for user preferences of the underlying OS.

Of course, each sensor also needs to implement its own specific logic for sensing and eventually making some inference on the sensed properties, to provide meaningful sensor data to *PASS*. From a technical perspective the sensors can be coarsely distinguished in the way they gather the needed information, into four distinct groups:

- A small number of sensors infer the needed information using standard Java classes (e.g., IP Address sensor).
- Most of the sensors use the `exec(String command)` method of the Java `Runtime` class, to execute command line utilities of the OS and then filter and parse out needed information form the output (e.g., Battery sensor).
- Another group of sensors executes one or more AppleScripts using the `osascript` command based on Apple's *Open* Scripting *Architecture (OSA)*, to retrieve the required information (e.g., Active Chats sensor).
- And finally, a number of sensors also use specific Java libraries—often based on Java Native Interface (JNI) to access some hardware components (e.g., Ambient Light sensor).

In the same way, the sensors can also be grouped by their basic functionality. Seven sensors monitor whether specific hardware is currently connected to the computer (e.g., Headphones Connected sensor). Six sensors monitor the state of internal or external hardware (e.g., Motion sensor). Five sensors are concerned with network connectivity (e.g., WiFi sensor), and five other sensors monitor different processes on the level of the operating system (e.g., Focus

Application sensor). Four sensors are monitor specific applications (e.g., Calendar sensor) and two sensors try to infer information on the physical presence of persons (e.g., Voice Activity sensor). In the Following detailed information is given on the implementation for each of the 29 sensors:

- *Active Access Point (1×1, Pull):* This sensor returns the Basic Service Set Identification (BSSID) i.e., the MAC address of the wireless access point the computer is currently connected to. The sensor is realised by invoking the Mac OS X command line utility `airport` utility with the option `–I`.
- *Active Chats (1×1, Pull):* This sensor returns the cumulative number of active chats for the three predominant chat applications under OSX: iChat [Apple Inc. 2011], Skype [Skype Limited 2011], and Adium [Adium Team 2011]. It is realised in form of an AppleScript that retrieves and sums up the number of active chats for each chat running application.
- *Active Network Interfaces (n×1, Pull):* This sensor uses the `ifconfig` command to sense which of the following network interfaces is currently active: ethernet interface (en0), airport interface (en1), and firewire interface (fw0). It returns a n×1 list of the active interfaces.
- *Ambient Light (1×1, Pull):* This sensor returns the intensity level of ambient light as sensed by the Apple Ambient Light Sensor (i.e., a photodetector intended for adjusting keyboard and display brightness) build into some models of Apple's PowerBooks, MacBooks, MacBook Pros, and iMacs. It relies on a JNI-based (Java Native Interface) library to access this value.
- *Applications (n×1, Pull):* This sensor delivers insight into the currently running applications by delivering a list of currently running processes. The information is obtained by executing the *process status* command `ps –xc`.
- *Application Focus (1×1, Pull):* This sensor returns the name of the application that is currently in focus. It is realised via an AppleScript that returns the name of the frontmost application process.
- *Battery (1×m, Pull):* This sensor collects information on the status of the battery for current Apple laptop computers including the battery's capacity, the charging state, and whether it is fully charged. It utilises the `ioreg` command to get this information from Max OS X I/O Kit registry. It specifically uses the `–c` option to retrieve only object properties of instances of the AppleSmartBattery class and parses the necessary information from these properties.
- *Bluetooth Devices (n×1, Push):* This sensor provides a list of the currently nearby and discoverable Bluetooth devices. The n×1 list

includes the Bluetooth device name (also referred to as user-friendly name) or if not available the Bluetooth device address for each discovered device. The information is obtained via the BlueCove library [BlueCove Team 2008], which implements the JSR 82 Java APIs for Bluetooth [SUN Microsystems Inc. 2010].

- *Calendar (1×m, Pull):* This sensor delivers information whether there are ongoing and upcoming events entered in the user's iCal application. It is realised via an AppleScript that checks all user-selected calendars for current events (ongoing) or events that start in the next 15 minutes (upcoming).
- *Connected FireWire Devices (n×1, Pull):* This sensor lists the IDs of all currently connected FireWire devices. It utilises the `ioreg` command to get this information from Mac OS X I/O Kit registry in combination with the `grep` command to filter for entries concerning FireWire devices.
- *Connected USB Devices (n×1, Pull):* This sensor lists the IDs of all currently connected USB devices. It utilises the `ioreg` command to get this information from Mac OS X I/O Kit registry in combination with the `grep` command to filter for entries concerning USB devices. Built-in USB devices like the iSight or FaceTime HD camera, Bluetooth USB Host Controller, etc. which are permanently connected, are ignored by this sensor.
- *CPU (1×m, Pull):* This sensor samples the CPU load in per cent by using the summary information from the `top -l 4` command. It senses the user CPU load in per cent, the system CPU load in per cent, and the idleness of the CPU in per cent.
- *Email (1×m, Pull):* This sensor provides information about the email activity of the user in the last minute. It returns the number of received, unread and sent emails from the OS X Mail.app for this period and is realised in form of an AppleScript.
- *Ethernet Connected (1×1, Push):* This sensor parses the output of the command `ifconfig en0` for the status *inactive* to check if the computer is currently being connected to or disconnected from a network via an Ethernet cable.
- *Face Detection (1×1, Pull):* This sensor aims at sensing the physical proximity of a user to the computer by checking whether a face is detectable in the computer's webcam. It was primarily designed to work with Apple computers with a built-in iSight or FaceTime HD camera like Apple MacBooks or iMacs. It is based on the OpenCV (Open Source Computer Vision) library [Willow Garage 2011]—respectively the partial port of the library for Java and Processing

[Cousot & Stanley 2011]. The usage of this sensor therefore requires the upfront installation of the OpenCV Framework. The detection of frontal faces in the camera image relies on the approach proposed by Viola and Jones [2001] as it is implemented in the library's cascade classifier. The code for the computer vision is separated from the normal code of the sensor in an own separate Java archive and started in a own virtual machine. This way exceptions that are thrown during face detection (e.g., if the code fails because the camera resource is already occupied) do not affect the robustness of the overall sensor. In order to minimise the error, the face detection is not executed once but six times on the camera stream and a face detection is only reported if in more than half of the executions a face is detected.

- *Focus Title (1×1, Pull):* This sensor returns the title of the frontmost window that is currently in focus. It is realised via an AppleScript that returns the title of the frontmost window. For some applications it additionally uses simple heuristics in order to get for example the document name for a word processor or the name of the current web page for the browser.
- *Headphones Connected (1×1, Push):* This sensor detects when an external headphone or speaker jack is currently plugged in or removed from the headphone socket of an Apple computer and returns the according state. It utilises the `ioreg` command to get this information from Mac OS X I/O Kit registry in combination with the grep command to filter for a *Headphones* entry.
- *Input Idle (1×1, Push):* This sensor records if the mouse, keyboard, trackpad or other input devices have been idle for a pre-specified period. The period can be set via an idle time threshold in the sensor's preferences. The sensor uses the input device with shortest idle time for comparison with this threshold. Idle times are retrieved via the command `ioreg -c IOHIDSystem` from Mac OS X I/O Kit registry. The entries are filtered for `HIDIdleTime` to get the idle times for key strokes, mouse movement, mouse buttons, trackpad interaction etc.
- *IP Address (1×m, Pull):* This sensor collects information on the current local and external (assigned by the Internet Service Provider) IP address of the computer. The local IP address is retrieved via the standard Java class `java.net.InetAddress`. The external IP address is retrieved by parsing the returned website of an IP lookup service (http://checkip.dyndns.org) that is accessed with the command line tool `curl`.

- *Motion (1×m, Pull):* This sensor can give insights if the laptop computer the software is installed on is moved physically. It returns three float values for the acceleration along the x-, y-, and z-axis as sensed by the Apple Sudden Motion Sensor (i.e., a triaxial accelerometer intended for detecting sudden acceleration, like it would occur when the device was dropped, in order to take measures that prevent a disk head crash) build into some models of Apple's PowerBooks, MacBooks, and MacBook Pros. It relies on a JNI-based (Java Native Interface) library to access this value.
- *Mounted Volumes (n×1, Pull):* This sensor provides a list of the currently mounted volumes including among others internal and external hard drives, network shares, or USB flash drives. The information is gathered by executing the list command `ls /Volumes/`.
- *Mouse Connected (1×1, Push):* This sensor returns whether an USB mouse is currently being connected or disconnected from the computer. It utilises the `ioreg` command to get information for objects that are instances of `IOUSBDevice` from Mac OS X I/O Kit registry in combination with the `grep` command to filter for a *Mouse* entry.
- *Power Connected (1×1, Push):* This sensor monitors if the power chord is currently being attached to or detached from the computer and returns the according state. It utilises the `ioreg` command to get information from Mac OS X I/O Kit registry in combination with the `grep` command to filter for the entry `ExternalConnected`.
- *Screensaver Active (1×1, Push):* This sensor reports if the screensaver is currently activated or deactivated. It is realised via an AppleScript that checks the OS X System Events suite for the existence of a process named ScreenSaverEngine.
- *Second Monitor (1×1, Push):* This sensor returns the number of connected displays in multi-screen environments each time a new display is attached or detached. The information is obtained via the standard Java class `java.awt.GraphicsEnvironment`.
- *Skype (1×m, Pull):* This sensor collects information around the usage of the Skype [Skype Limited 2011] application including the number of contacts currently online, the mood message, the online status and if a call is ongoing. It is realised in form of an AppleScript.
- *Voice Activity (1×1, Push):* This sensor reports if human voice activity in form of speech is currently detectable via the built-in microphone. It therefore utilises the Java based CMU Sphinx 4 speech recognition toolkit [Carnegie Mellon University 2011] to detect speech start and

speech end events in the audio stream. Some further basic parameters control how the sensor decides to report the start or end of a speech activity. These parameters are: *speechDurationThreshold* (10.000 ms), which defines the minimum length of a speech segment to be reported as speech activity; *speechTrailer* (5.000 ms), which defines the maximum length of a break between words and sentences in order to decide the speech segment is still ongoing; and *speechEndThreshold* (6.000 ms), which defines the length of time after the last word is said, to decide the speech segment has ended. The pre-set values for these parameters have been determined in an explorative manner and worked well for the tested settings, but can be adjusted via the sensor's preferences.

- *Volume Settings (1×n, Pull):* This sensor collects information on the state of the audio hardware of the computer. In detail that is the current set output, input, and alert volume (in a range from 0 to 100) and whether the output is currently muted or not. It is realised via an AppleScript that invokes the expression `get volume settings`.
- *WiFi (n×m, Pull):* This sensors returns a table of all nearby, discoverable WiFi networks, each with an entry of the Service Set Identifier (SSID), Basic Service Set Identification (BSSID), and Received Signal Strength Indication (RSSI). The sensor is realised by invoking the Mac OS X command line utility `airport` with the option `–xs` which returns the needed information in form of an XML-formatted String that is parsed by the sensor.

5.3.2.2 The PASS Daemon and GUI

The *PRIMI Advanced Sensor Suite Daemon* (*PASSDaemon)* and PRIMI Advanced Sensor Suite GUI (*PASSGUI*) together form the infrastructure for deploying the previously described sensors to enable *user-centric* sensing on laptop computers (cf. Figure 43).

The *PASSDaemon* is implemented to run as a background process by utilising the Apache Commons Daemon [Apache Software Foundation 2013] library. The task of the daemon is threefold: discovering newly deployed sensor plugins, providing a run-time environment for all deployed sensor plugins, and offering different outlets for the collected sensor data.

The `PASSDaemonPluginClassLoader` discovers all sensors that have been placed in form of a jar-file into the plugins folder. The discovery happens at start-up but can also be initiated manually. The `PASSSensorService` acts as the run-time environment, by registering listeners to *Push-Sensors* and collecting data from *Pull-Sensors* in a customisable interval per sensor. Every time a new sensor reading is available, it is passed to the `PASSPersistence`,

which implements various outlets for the data. The two of interest for this work are the `SimpleXMLPersistence` and the `SensationPersistence`. The first simply writes out all data into XML files stored on the local hard drive. This was used for logging the data in the study. The `SensationPersistence` on the other hand allows sending each sensed *SensVal*-datum packaged into a Sensation `SensorEvent` to a Sens-ation instance via XML-RPC. This of course was used for building the proof-of-concept implementation.

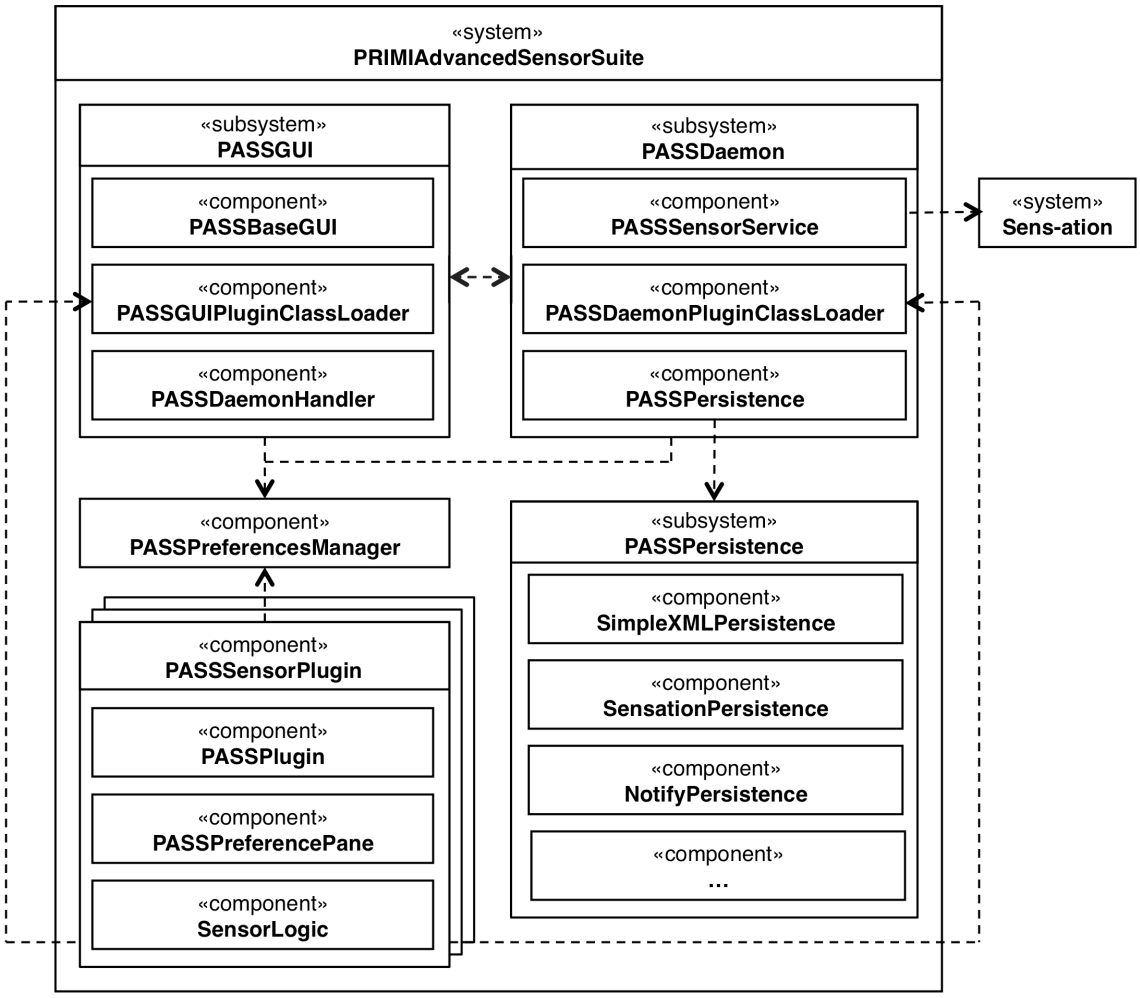

Figure 43. Overview of the PRIMI Advanced Sensor Suite implementation.

The *PASSDaemon* is a self-contained application that is distributed with an Installer for Mac OS X systems. Once installed, it runs via *launchd* as a background process and automatically starts with the OS. It uses a TCP Socket for inter-process communication, as for example with the GUI.

Additionally, the GUI allows configuring the daemons general settings (cf. Figure 44), including the configuration for the different persistence services, as well as each individual sensor (cf. Figure 37). For configuring the sensors, the GUI simply loads and displays the individual sensors' preferences dialogue retrieved via its `getPreferencesPane()` method.

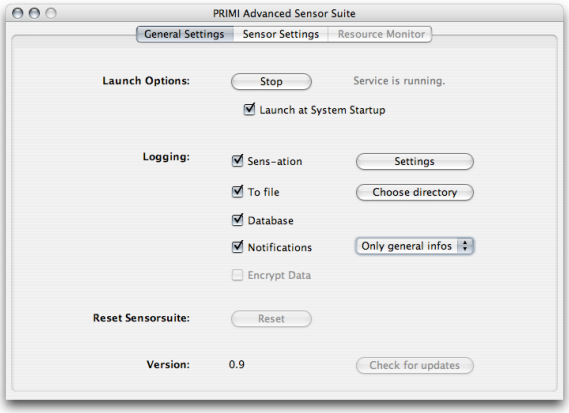

Figure 44. PRIMI Advanced Sensor Suite GUI with the General Settings dialogue open.

The `PASSPreferencesManager` functions as a centralised entry point for the GUI, the daemon, and all Sensor Plugins for storing and retrieving settings and configurations. For example, when configuration changes are made in the GUI to a specific sensor via the preference pane, the changes are simply stored in a preference file. When the daemon restarts, it automatically loads the updated preferences. To force the restart after changing the configuration for a sensor, the GUI offers the "Restart with new settings"-Button (see Figure 37).

5.3.3 Implementation of the Stream-Based Active Learning

The basic functioning of the Stream-Based Active Learning approach as well as many details of the concrete proof-of-concept implementation have been already laid out in the previous section. In this section I share the details surrounding the implementation of the approach as *Inference Engines* for *Sens-ation*.

Sens-ation was developed by Gross, Egla, and Marquardt [2006] as an event-driven architecture based on the Observer Pattern [Gamma *et al*. 1994]. As previously noted, the main components from a conceptual view are *Sensors*, *Inference Engines* and *Actuators* (cf. Figure 45). Communication between these components is based on the publish-subscribe paradigm that allows the message passing between the components in form of `SensorEvents`. While the *Sensors* are the subjects, and *Actuators* are the observers, *Inference Engines* always take the role of subject and observer at the same time. Hence, implementation of the abstract class `InferenceEngine` receive new input in form of a `SensorEvent` via the `notify(Object event)` method, process the input, and send the inferred result in form of a `SensorEvent` to all registered observers. As implementations of the `InferenceEngine` class are quite restricted in changing their functionality, the class `ConfigurableInferenceEngine` extends the possibilities of IE

implementations by providing a standardised way to customise the settings. Technically, `InferenceEngines` and `ConfigurableInferenceEngines` run inside *Sens-ation*—hence in the same virtual machine.

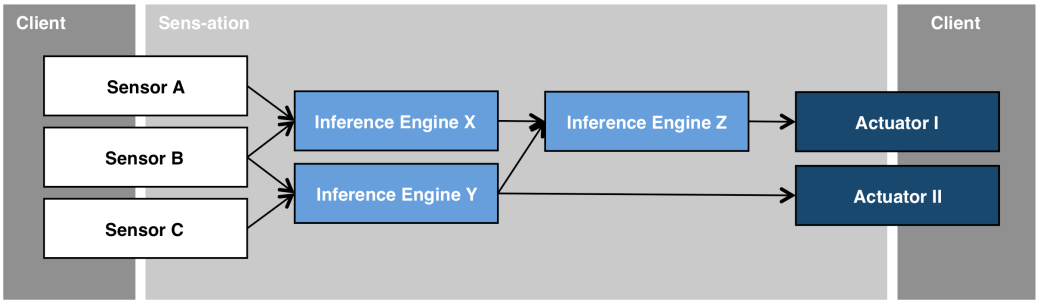

Figure 45. A conceptual depiction of the event flow from Sensors via Inference Engines to the Actuators.

Sensors and *Actuators* on the other side only fulfil the functions of a remote proxy inside *Sens-ation*. Their actual implementations usually run in an own virtual machine on the various clients and send their data to *Sens-ation* (e.g., via XML-RPC).

In the proof-of-concept implementation, all three implemented IEs make use of the configuration possibilities. For example, the `IEActiveLearning` class provides two configuration parameters. The first parameter is the classifier, which can be set by using Weka's command line approach for specifying a classifier implementation followed by the options as String (e.g., "weka.classifiers.trees.HoeffdingTree -L 2 -S 1 -E 1.0E-7 -H 0.05 -M 0.01 -G 200.0 -N 0.0"). The second parameter is the selection of the *ActiveLearningStrategy*, which can be chosen from a set of provided options (i.e., the *FixedUncertaintyStrategy* and the *ClusterStrategy*). For the inference engine `IETemporalDiscretisation` the push strategy can be configured (i.e., *NewValuePush* and *StaticTimerPush*). The parameters can either be set via XML-RPC when instantiating the IEs programmatically or via the Inspector dialogues of the *CollaborationBus Aqua* editor developed by Schirmer and Gross [2011] (cf. Figure 47).

A further addition to *Sens-ation* was the introduction of *SensBatch* and *SensSet*. While the introduction of the *SensVal* format provided more freedom for describing more complex sensor data, the data processing often required passing several sensor readings at once as *SensVal* values. The introduction of *SensSet* and *SensBatch* now allows passing bulky data between inference engines. As can be seen in the Schema description (cf. Appendix C —) *SensSet* is basically a collection of *SensVal* elements, while *SensBatch* is a collection of *SensSet* elements. The introduction of these data formats among other things allowed handing over segments of the time series from the

`IETemporalDiscretisation` to the `IEFeatureEngineering` for further processing.

The `IEFeatureEngineering` and `IEActiveLearning` make use of the Weka library's filter- and classifier-algorithms. Hence, the implementations also internally use the Weka classes for building feature-vector representations of the data (i.e., `Instance` and `Attribute`). For transferring the instances between the inference engines a Utility class is able to convert them into 1×m *SensVal* representations and back.

Finally, all three IEs constantly serialise their internal state to disk, to be robust to restarts or crashes of the *Sens-ation* instance. As all these IEs maintain several variables, lists, and hash tables that are continuously updated, it is crucial that they can unmarshal their state from disk, once the server restarts. I gave an example for such a list in section 5.2.2. To recapture, in the feature extraction process, nominal values are converted into binary representations. The `IEFeatureEngineering` therefore maintains for each nominal value (e.g., *ApplicationFocus*) a list with entries of all unique values it has seen so far (e.g., Firefox, Finder, Terminal, Word). As this information would be lost after a restart, the list is serialised to disk, every time a new entry is added. In the same way also the learned models are stored, each time they are updated using Weka's `SerializationHelper` to write a `Classifier` instance to disk by the `IEActiveLearning`.

For validating the performance, the `IEActiveLearning` also produces a log-file, collecting various figures on the classifier performance, which can be easily rendered into a chart as shown in Figure 49.

5.3.4 Implementation of the ActuatorDaemon and the Adaptation

Much alike the *PASSDaemon*, I also implemented the *ActuatorDaemon* to bundle the communication of *Actuator* clients with *Sens-ation* into a single instance running on the users' computers. The daemon provides a runtime environment for different actuator implementations in form of a constantly running background process. Via *Sens-ation*'s `ActuatorXMLRPCGateway` it is possible to register *Actuators* as observers of the output of certain *Sensors* or *InferenceEngines*. For each *Actuator*, it is possible to connect one or more actuator client implementations via the gateway, by specifying an URL (i.e., IP-address and Port) together with a remote callback method, which is reachable via XML-RPC. Registered actuator client implementations are then informed via a long polling mechanism about every new sensor event the Actuator observes. The *ActuatorDaemon* provides convenient methods to centrally handle this communication with the `ActuatorXMLRPCGateway` and respectively notify the implemented actuator. In the context of this work the implemented

actuators *ESMQueryActuator*, *VolumeSettingActuator*, and *PRIMIMundane Actuator* are of interest (cf. Figure 42).

The *ESMQueryActuator* is implmeneted to provide a straightforward query mechanism for the `IEActiveLearning` in form of a dialogue. An example screenshot is depicted in Figure 41. When triggered by a `SensorEvent`, the XML configuration transmitted in the event is used to present the query in form of a dialogue. The XML therefore needs to comply with the schema description in Appendix D —. Besides the presentation of several options with radio buttons, also other answer formats are possible (e.g., Number, String, or Boolean). The answer provided by the human oracle is sent back in form of a `SensorEvent` with the sensor ID (`sid`) that was retrieved from the configuration XML. The actuator also uses the values set for `requesttimeout` and `requestinterval` from the XML configuration for timing the presentation of the dialogue.

The *VolumeSettingActuator* is a very basic actuator that is able to control the OS X System Volume via an AppleScript. When the actuator is triggered, it expects an Integer value in the range from 0 to 100 inside the sensor event. This value (v) is used to adapt the volume accordingly with the simple AppleScript "`set volume output volume %v.`"

The *PRIMIMundaneActuator* uses XML-RPC to communicate with the PRIMI Mundane application. The actuator is triggered by Sens-ation with an n×m SensVal, containing a partial or complete list of all *Availability Categories* and a Boolean value for each category, encoding whether the user is available or unavailable for the contacts in this category. Based on this list, the implemented actuator adapts the PRIMI Mundane client accordingly.

5.4 Validation of the Stream-Based Active Learning Concept

In order to validate the concept behind the Stream-Based Active Learning Concept, a layered approach is necessary, as it is depicted in Figure 46.

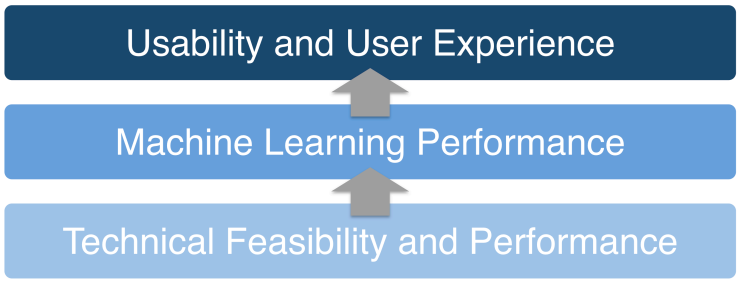

Figure 46. Layered validation scheme for the Stream-Based Active Learning concept.

Starting from the top, the ultimate validation goal would be to assess the *Usability and User Experience* (UX) [International Organization

for Standardization 2017] of the proposed approach of stream-based active learning, based on its actual manifestation in form of a concrete implementation. The usability aspect thereby comprises the effectiveness and efficiency of, and satisfaction with the system, in its ability to support the users in achieving their goals. The UX adds a slightly more qualitative notion and extends over the pure utility of the product, when analysing the users perceptions and responses from the use (or anticipated use). One important influencing factor for Usability and UX, is the quality of the adaptation and the effort for the user. This can be directly linked to the accuracy of the classification and the amount of needed training. Accordingly, the *machine learning performance* of the active learning approach is a key factor for the Usability and UX of the system. And conclusively, a sound and stable implementation with good performance is the foundation for a good machine learning performance and a usable system, placing the *technical feasibility and performance* at the foundation of all validation layers.

5.4.1 Validating the Usability and User Experience

Concerning the assessment of the Usability and UX, it needs to be clear that such an assessment could only be done in field in a meaningful way [Kray *et al.* 2007] not in a lab experiment [Duh *et al.* 2006; Kjeldskov & Skov 2014; Sun & May 2013]. This requires a number of real users actively using the system for an extended period of time (> 4 weeks), training the system, and actively using it for communication. Especially Groupware is difficult to evaluate [Grudin 1994]. It must be noted that such an extensive evaluation was out of the scope of this thesis. Yet, the aim of this thesis was to lay the technical foundations for enabling such studies in future work.

This said, at least it was possible to get some feedback on the Usability and UX. First, as parts of the overall system (e.g., the *PRIMI Advanced Sensor Suite*) were used in the experience sampling study, it was possible to collect informal feedback on parts of the proof-of-concept implementation. Further, based on the small demonstrator for adapting the volume setting, I was able to explain the underlying idea of the system to users and discuss it with them. This was another source to collect some natural responses on usability and UX factors, in terms of general desirability, for example during two open house events of ours research lab. Beyond that, no further structured evaluation of the system's usability was undertaken.

5.4.2 Validating the Machine Learning Performance

Concerning the validation of the machine learning performance of the online learning system, it must be noted that most evaluation methods and metrics in machine learning are designed for static case (i.e. Offline Learning) [Gama

2008]. Further, evaluation methods, which are common in batch settings [Kohavi 1995], like holdout, cross-validation, leave-one-out, or stratification, are no perfect fits for learning from data streams [Bifet *et al.* 2011]. These methods often depart from a setting were training and test data is sparse, therefore trying to optimise the balance of data that is used for training and data that is used for testing. Data stream mining usually is applied to problems with a great amount of data, where it is often unproblematic to get enough training and test data. On the other side the main interest is not necessarily the overall quality in terms of the accuracy (i.e., the percentage of all test examples that have been classified correct), but often how the quality of the predictions evolve over time [Bifet *et al.* 2011]. Phenomena like concept drift and shift make it even harder to find good metrics, as common metrics like accuracy, precision, and recall are not sufficient to measure such aspects [Gama & Rodriques 2007].

Additional, in the field of context-aware computing also new measures are brought into the mix. For example, for the recognition of activities in continuous data, also the temporal correlation of the prediction with the ground truth becomes important, therefore new methods are suggested [Ward *et al.* 2006]. While they suggest measures that are essentially comparable to false positive and negative errors (i.e., Insertion and Deletion) they also introduce the notion of Merge and Fragmentation, which better illustrate the challenges with the time-domain. The measures are:

- Insertion: An activity is predicted while there is no activity in ground truth at this time (or depending on the detection method, is substituted for a NULL class).
- Deletion: An activity in ground truth is not predicted by the machine learning algorithm (or depending on the detection method, is substituted by a NULL class).
- Merge: Only one on-going activity for a time span is predicted, while there are several activities in the ground truth in this time span.
- Fragmentation: One on-going activity in ground truth for a certain time span is predicted as several distinct activities.

Further Ward et al. define Overfill and Underfill [Ward *et al.* 2005] as complementary measures to Insertion, Deletion, Merge and Fragmentation for the evaluation of continuous recognition problems. Both are concerned with the length of the predicted activity event and the length of the activity in ground truth:

- Overfill: An activity is generally predicted correctly, the activity is predicted longer then in ground truth. This can either be the start time or end time of the activity, or both.
- Underfill: Also here the classification is generally correct, but the activity is detected shorter then it actually happened in the ground

truth, leading to some period of False Negatives at the beginning or end.

Of courses such metrics can only be produced if ground truth data is available, as for example annotated video observations of people performing gestural input, which can be compared to the predictions. In the case of predicting the availability of people however, such fine-grained annotated ground truth data is basically inaccessible as it is not directly observable. Annotating video observations by third persons for example is not an option, as humans are not specifically good at estimating the interruptibility of others based on video observations [Avrahami *et al.* 2007]. Requiring study participants to continuously produce such data themselves (e.g., through a much more fine-grained ESM) would likely greatly influence their behaviour and the according availability. Hence, in the case of this work, the only data for evaluation accordingly are the sparsely sampled labels from the ESM study.

In respect to active learning, there also not really is an agreed on procedure for empirical evaluations [Ramirez-Loaiza *et al.* 2017]. Further, only few people so far conceived approaches for testing adaptive systems and intelligent agents [Kulesza *et al.* 2011; Stumpf *et al.* 2014] that take into account that only one person—the user—can really evaluate if the system is working correctly for them.

With all this in mind, there is one relative straightforward method for evaluating the machine learning performance in data stream mining settings: the *Interleaved Test-Then-Train* (also *Prequential*) evaluation [Gama 2010; Stefanowski & Brzezinski 2017]. In this method, each new unseen instance is first used for testing, and then for training. This normally does not happen per instance, but rather on a small number of instances as a block that is generated by a sliding window of a fixed size. Thereby a cumulative moving average for the accuracy is calculated.

For evaluating stream-based active learning the testing of each instance is done anyhow by the *FixedUncertaintyStrategy*. Accordingly, it is easy to simply store this result. However, to evaluate the active learning part, the method of *Interleaved Test-Then-Train* needs to be slightly adopted. Instead of using every instance for training after testing, the adopted method would rely on the applied *ActiveLearningStrategy*, and only uses those instances for training, for which the *ActiveLearningStrategy* requests a label. Besides calculating a moving average of the accuracy, also the number of label requests is stored as a second metric. Ideally the two numbers could be used to calculate some sort of ratio between the accuracy and the number of requests, for the evaluation of stream-based active learning. However, currently no such measure for comparison exists.

To conclude, an evaluation of the machine learning performance for the proposed approach is problematic for two reasons. First, the lack of adequate

measures and metrics for the approach of active stream-based learning makes it hard to compute meaningful numbers, which can be compared to some baselines or benchmarks. Second, as I repeatedly noted, the focus of the proof-of-concept implementation is on the end-to-end feasibility, and therefore I made several compromises that certainly degrade the learning performance. In other words, the current implementation is not optimised for reaching a competitive machine learning performance.

However, in line with the validation of the technical feasibility, I present a chart that allows a basic assessment of the machine learning performance by presenting the values from the modified *Interleaved Test-Then-Train* over time for one dataset in 5.4.4.

5.4.3 Validating the Technical Feasibility and Performance

On the lowest level, the technical feasibility and performance need to be validated through adequate tests. Software testing "is the process of executing a program with the intent of finding errors" [Myers *et al.* 2004, p.6]. Thereby functional as well as on non-functional aspects of the system are tested against different criteria.

Again, also on the level of pure software testing adaptive context-aware applications are specifically challenging [Ahmed 2011; Bardram & Friday 2009]. Many systems are distributed, and parallel, and therefore bedevil software testing [Maddox 2015]. The systems are deployed in environments with many uncertain aspects, as for example network connectivity, sensor failures, etc. The testing of functional aspects in most cases also requires either field tests *in context*, where the conditions for testing are barely controllable, or, as an alternative extensive simulation. In the case of this work, also software components and systems were used (e.g., *Sens-ation* [Gross *et al.* 2006]), that are in itself research prototypes and therefore not completely formally verified.

The main aim however was to successfully demonstrate the end-to-end functionality [Maddox 2015] of the implementation. This was achieved in three different ways on a system test level. That is, through live testing of the functionality with a simple demonstrator, through a real-time simulation of context based on replaying sensor log-files, and through a non-real-time, accelerated simulation.

The live testing was basically done while multiple times demonstrating the adaptation of the volume settings to users. By using two sensors that can be influenced easily—as for example *Application Focus* and *Connected USB Devices*—I was able to repeatedly demonstrate how a specific combination (e.g., PowerPoint as focussed application and detaching the USB mouse) could lead to an adaptation of the volume to a specific level. The visual editor for

configuring the Sens-ation platform [Schirmer & Gross 2011] allowed to easily explore different configurations (cf. Figure 47).

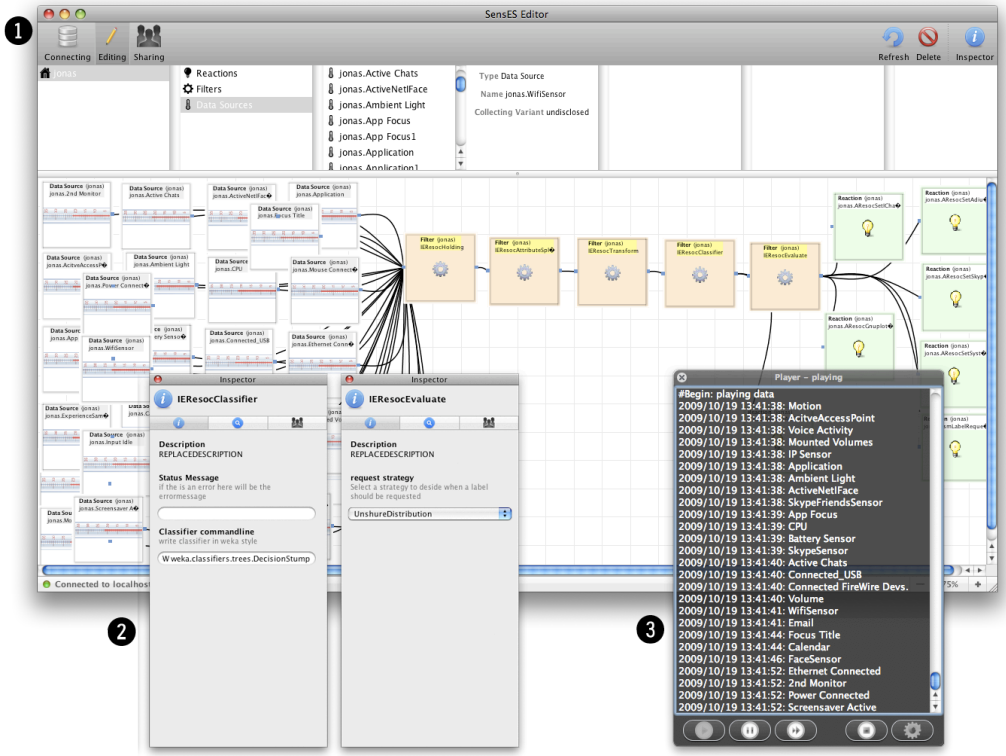

Figure 47. Screenshot of a test setup configured inside the *CollaborationBus Aqua* [Schirmer & Gross 2011] editor (1) with two opened inspector dialogues (2) for configuring parameters of two inference engines and the *SensorEvent-RePlayer* (3). The editor shows the configured event flow starting from the left with the 29 proof-of-concept sensors (white; with scale icon), routing into a chain of inference engines (beige; with gear icon) that on the right are connected to different actuators (lime; with light bulb icon) [Fetter & Gross 2014].

For the simulation the *SensorEvent-RePlayer* (Figure 48) was implemented. This standalone application allows loading a folder full of sensor event log-files and basically permits replaying the sensor data to a *Sens-ation* instance. Therefore it can be seen as a context-simulator. The application basically has two modes for replay. In real-time mode, the time-stamps from the events are used, and the events are sent in the correct order and by preserving the pacing in terms of intervals between the events. A fast-forward mode allows to playback the sensor data in correct order, but with accelerated speed (i.e., playback speed is as fast as the data can be read from the files and processed). Naturally, the player allowed in both cases to stop or pause the execution at any time.

With the means of the simulator it was feasible to use the collected sensor data from the ESM study for validating the proof-of-concept implementation. By replaying the data in real-time mode, it was observable that the *ActiveLearningStrategy* promptly starts requesting a label from the user by asking for the current availability via a dialogue. However, the test in real-time mode

only allows verifying the basic end-to-end functionality and the computing performance. As the context is only simulated, a meaningful input of the availability by a real user or tester is not possible this way. Also, replaying the whole data in real-time would have taken the same amount of time as the study (i.e., four weeks or more).

Figure 48. Screenshot of the SensorEvent-RePlayer application replaying a folder with sensor data log-files for testing the implementation.

The fast-forward mode on the other hand allowed me using the complete study data for the simulation in a reasonable time. One challenge however is the provisioning of labels. While generally the *ActiveLearningStrategy* can decide anytime to request a label from the user, in the study data there was only a label for roughly every 30 minutes. Accordingly, I adapted the IE for active learning for the simulation in a way, that it only evaluated the model at times, when a label is available in the data. The rest of the time, the sensor data is not used for evaluation. Yet, of course all the sensor data did go through the pre-processing, as any time when a label is available, there is also an instance needed to evaluate. The results of this simulation are presented and discussed in the next section.

In needs to be added, that in this verification of the end-to-end functionality with simulated sensor data, the PRIMI Advanced Sensor Suite (*PASS*) was not tested. PASS was only tested in the end-to-end functionality in the demonstrator for the adaptation of the volume setting. However, the PRIMI Advanced Sensor Suite and the individual sensors were heavily tested to guarantee the successful collection of sensor data. The good and uniform data quality from the study, together with the fact that the logged sensor data could be used without

problems in the simulation, also allow a positive assessment of the performance and stability of this specific subsystem.

5.4.4 Final Result and Discussion

In the previous sections I highlighted some results but also discussed the challenges with the validation of the proposed approach. Here, I present the main validation result, the demonstration of the general feasibility through the prototypical implementation in form of a proof-of-concept. This proof-of-concept implementation was verified in tests to be stable, with good computational performance, and delivering consistent and reproducible results. While the implementation is not optimised for machine learning performance, simulated tests based on the previously collected study data show promising results. Figure 49 depicts the results as a trend over time, based on the sensor data and availability estimates of one participant. The chart exemplary shows the performance of predicting the six classes for one user's General Availability based on the collected sensor data and labels. The data was played back with the *SensorEvent-RePlayer* in the fast-forward mode, and went through all previously described preprocessing steps. The *FixedUncertaintyStrategy* was utilised as the *ActiveLearningStrategy*, with a setting of 0.6 for the minimum and 0.99 as the maximum threshold for the confidence value. That is, if the confidence for a prediction was either below the 60% mark or above the 99% mark, a label was requested. As classifier Weka's incremental Naïve Bayes implementation *NaiveBayesUpdateable* was used with default settings.

The blue line in the chart shows the trend of the overall accuracy in form of a cumulative moving average. At the beginning, the accuracy quickly rises and then dithers around the 70% mark until the last third, where it rapidly falls under the 20% mark for a short time before going back up. The sudden drop is likely caused by some changes in the routines of the user during a holiday break, leading to a drift or shift in the concept or the underlying data: Different conditions, a different situation or also different availability preferences for this time interval.

From the vertical pink lines in the chart—which indicate each time the *ActiveLearningStrategy* requested a label—we can see that after an initial intense training phase, the number of label requests went down. In the same way we can observe how the probable concept drift around the end of the year, also leads to an increased request for labels towards the end of January.

Overall, the approach clearly does not reach the average accuracy of 81.35% that was demonstrated in the manual batch setting (compare 4.1.4). But this was also expected, as the offline evaluation used a superior feature selection mechanism in combination with a better-tuned classifier algorithm. The proof-of-concept implementation on the opposite, in many parts relied on naïve

implementations for several of the proposed steps. As argued before, this was the consequence of demonstrating the general feasibility of a lifelong learning system with currently available implementations of machine learning algorithms.

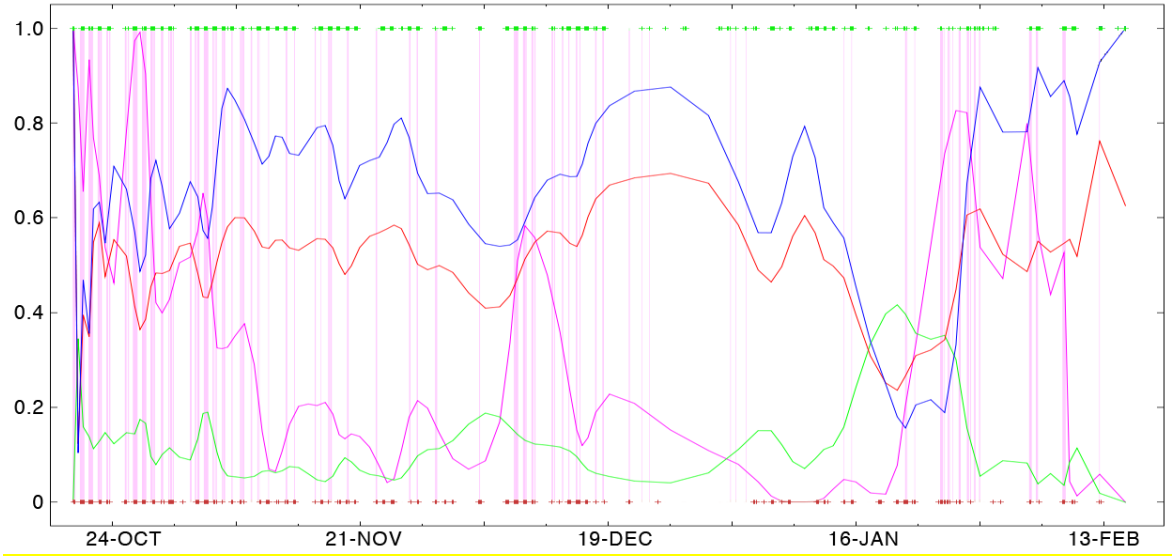

Figure 49. The chart shows the results over time using the *NaïveBayesUpdateable* classifier to predict the General Availability of one participant (P1). The red and green cross marks the times when a label in the data set is available, that was either evaluated correct (=1.0) or incorrect (=0) by the classifier. The pink vertical lines show when the *ActiveLearningStrategy* decided to request a label. The blue line shows the trend of the overall accuracy of the classifier. The red, green and pink line show the confidence values for the real class (red) the predicted class (green), and the normalised absolute difference between real and predicted class (green)) [Fetter & Gross 2014].

Accordingly, the validation shows that the approach is feasible in general. It can be fairly assumed, that novel algorithms for online feature selection as well as more advanced algorithms for classification like kernel learners or tree ensembles can further boost the performance. A better performance would technical manifest in either a better accuracy or a lower number of queries. Ultimately of course, only a study with real users can assess the benefit of the approach. A further discussion of such aspects along with a reflection on future work can be found in the final chapter of this thesis.

6 Concluding Remarks and Outlook to Future Work

In this final chapter I first provide an résumé, summarising the contributions of this thesis leading over to a short discussion. At the end, I present an outlook to areas for future work.

6.1 Summary and Conclusions

In this thesis, I first provided a definition of the terms *presence* and *availability* grounded in literature. I defined it, as the positively connoted advertisement of ones' interruptibility in technology-mediated communication towards others. It is the way in which we are in view or at hand, and receptive for technology-mediated communication. By connecting these terms to concepts like privacy, impression management, or interruptibility and reflecting on the interpretation of these concepts in different disciplines like sociology or psychology, I provided a glimpse at the multifarious functions *presence* and *availability* have to fulfil, and how complex and entangled this can be. This way, I laid the theoretical and conceptual foundations [Stolterman & Wiberg 2010] for further exploration in the following steps.

I argued that the current possibility for expressing availability (e.g., in Instant Messaging clients) with one single online state that needs to be manually adapted, does not account for the social reality of the users and the *dynamic* and *selective* nature of *availability*. I further provided insights into patterns of selective information disclosure [Fetter & Gross 2008; Gross & Fetter 2010] and the dynamic nature of availability [Fetter *et al.* 2010a] based on two studies.

Hence, I analysed whether the repetitive and manual task of availability management can be automated. In an explorative machine learning approach I was able to successfully demonstrate that it is possible to use sensor data to predict the *selective availability* of individual users [Fetter *et al.* 2010b; Fetter *et al.* 2011b]. Thereby I achieved a classification performance that is at least on par with those of others [Fogarty *et al.* 2004a; Horvitz *et al.* 2004; Ter Hofte 2007], although they analysed the predictability only for less complex settings. I also presented several lessons learned, whereby the most central insight is that prediction models for selective availability have to be learned for each user individually.

To come up with a solution, I explored the opportunity space of online learning and real-time data mining for the automation of privacy and availability configurations [Fetter & Gross 2009b; Gross *et al.* 2010]. As a result I propose the concept of *active stream-based learning* of availability preferences from sensor data [Fetter & Gross 2014]. By combining two machine learning paradigms it

becomes possible to learn personalised user models for unobservable user preferences. With a proof-of-concept implementation I also demonstrated the general feasibility of the approach.

While the proposed approach generally works, it needs to be understood that the training of the system will require some effort and attention from the user. The clear limitation of this work is, that the effort more or less is done, in order to automate only one single task for one single user: managing the personal availability. However, this limitation—to only learn the highest-level concept—is designedly introduced in this work. This approach is not meant to be a standalone solution, but might be later combined with other solutions. For example, to reduce the overall effort, it is very likely that a system in the same way (i.e., active stream-based learning) could upfront learn lower-level constructs, like the personal significance of a location or the detection of a specific tasks or activities from sensor data. When these lower-level constructs then are fed back into the system, to predict the higher-level constructs (e.g., availability) it is likely that this will improve the classification performance while overall reducing the number of queries for the user. This also could allow combinations of personal models with general models and so forth.

Of course, the perfect adaptive system would come without any extra effort for the user. Yet, what we tend to forget is, that also with humans, we also have that extra layer of teaching and training. For example, we invest time upfront in a new employee, an assistant, a secretary, to teach them how a task is done, in order to benefit later from this person carrying out such tasks for us. The major difference is, that this interaction with humans to teach and train them is tending to be overlooked, as it is one of the roots of our co-existence and comes naturally. Training a machine through clicking through dialogues however is not.

6.2 Future Work

Accordingly, there are different opportunities for continuing, deepening, further validating or simply complementing this work. While a few examples and possibilities for follow-up work were already mentioned in the different chapters, in this final section I highlight a few ideas for future research.

6.2.1 Opportunities for Technological Improvements

By focussing only on the proposed adaptive system, there are already some clear opportunities for extending and improving the work.

First, the proof-of-concept system makes several compromises in the implementation. An optimisation of the learning performance could be achieved by evaluating novel learning algorithms that promise to be a better fit

for learning from data streams. For example, implementing mechanisms for optimising the feature selection in an online setting can probably improve the results significantly. Further, for the model updating process, some of the proposed strategies like attribute bagging and co-training should be realised.

Second, some alternative approaches for learning the concept of availability should be explored. Currently the learning task is treated as a classification problem. As the classification does not take into account an ordinal order of the classes, the error of classifying *Do Not Disturb* as *Not Available* is not treated differently then classifying *Do Not Disturb* as *Text Me!* For the user, the result however is very different. If the availability level—or selective availability level towards a group or person—can be seen as a gradual scale from being highly unavailable to being highly available, learning a regression instead of a classification might lead to improved results for the user.

Third, in the current solution the different availability states for the different availability categories are treated independently—each is handled as a separate classification problem. We have seen in section 4.1.4.3 that people tend to have different patterns of configurations. For example, being highly available for private contacts for some users coincides with being highly unavailable for work contacts. Currently, these correlations are not used for mutually improving the classification for the different categories.

While these are only a few suggestions for technological improvements of the implementation, there are certainly several more.

6.2.2 Opportunities for Understanding and Improving the User Experience

Studying privacy preferences for new technologies is a difficult task as it deals with phenomena that manifest over time "within repeated and socially established use" [Barkhuus 2012, p. 372]. Most studies based on prototypes therefore often lack the grounding in the actual lived situations of the subjects. For researchers that investigate the predictability of availability or interruptibility, this is even more critical. In the related work, researchers often stopped with analysing the collected data, and rarely implemented a real prototype of an adaptive system, let alone evaluated a real system in practise.

With my proof-of-concept implementation I created the possibility to deploy such systems in the field and study them. This allows pushing our understanding of the real benefits of such systems beyond simple metrics like the classification accuracy. In future studies, we might be able to answer questions as: What classification accuracy is good enough for a satisfying user experience [Kay *et al.* 2015]? How much effort are people willing to invest in the training? How can people balance training effort with the desired level of accuracy?

We further can reason on how to improve the usability of the system. For example, offering different modalities for answering the queries, with voice, gestures, gaze, etc. would allow the usage in novel contexts. A further possibility lays in investigating, how additional meta-information about the machine learning process could enrich the presentation of the availability information. For example, the degree of uncertainty for the prediction could help the interrupter in their decision. Further, the intelligibility and scrutability [Bellotti & Edwards 2001; Lim & Dey 2010] could be improved, by adding information on what criteria had the most impact on the classification result. For example, when hovering over the availability state in the IM client, a popup could show a text like: "The user is currently unavailable based of the information of the sensors *Active Access Point* and *Application Focus*."

6.2.3 Opportunities for Deepening our Understanding of Availability

Finally, the Experience Sampling Method (ESM) has proven to be a good candidate for getting insights in the availability preferences of individual users. To further deepen our understanding of the concept of availability—which ultimately would allow improving current systems—more studies in different context are needed.

In the study conducted in this work, I used random sampling with a relatively fixed interval. While this allowed to get a good overview of an individual's availability over the course of a day, it is very likely that this way I also missed a lot of shorter phases of high availability or unavailability. This way it is hard to tell, at what pace the availability changes. A future study could aim at identifying the moments at which the availability changes, by applying Context-Aware Experience Sampling. A simple Accelerometer sensor in a smart watch could be used to detect changes in the activity of tasks and then trigger a sampling, querying the user from which to which state the availability changed.

However, also improved tools to conduct such studies are needed. I have already provided some initial concepts and ideas for such tools [Fetter & Gross 2011b; Fetter *et al*. 2011a] that can be further explored and validated.

References

Abowd, G.D. What Next, Ubicomp?: Celebrating an Intellectual Disappearing Act. In *Proceedings of 14th International Conference on Ubiquitous Computing - UbiComp 2012* (Sep. 5-8, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 2012. pp. 31-40.

Abowd, G.D. and Mynatt, E.D. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction (TOCHI) 7*, 1 (2000). pp. 29-58.

Ackerman, M.S. and Mainwaring, S.D. *Privacy Issues and Human-Computer Interaction*. In Cranor, L.F. and Garfinkel, S., eds. Security and Usability - Designing Secure Systems that People Can Use. (1st ed.). O'Reilly Media, Sebastopol, CA, USA, 2005. pp. 381-401.

Adams, A. Users' Perception of Privacy in Multimedia Communication. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 1999* (May 15-20, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 1999. pp. 53-54.

Adium Team. Adium. http://adium.im/, 2011. (Last accessed: 10/12/2017).

Agamolis, S. New Technologies for Human Connectedness. *interactions 12*, 4 (July - August 2005). pp. 33-37.

Aggarwal, C. *An Introduction to Data Streams*. In Aggarwal, C., ed. Data Streams - Models and Algorithms. Springer, Berlin/Heidelberg, Germany, 2007. pp. 1-8.

Aggarwal, C. *Data Streams: An Overview and Scientific Applications*. In Gaber, M.M., ed. Scientific Data Mining and Knowledge Discovery - Principles and Foundations. Springer, Berlin/Heidelberg, Germany, 2010. pp. 377-397.

Ahmed, R.E. *Challenges in Testing Context-Aware Applications*. In Shah, S.A., Ilyas, M. and Mouftah, H.T., eds. Pervasive Communications Handbook. CRC Press, Boca Raton, FL, USA, 2011.

Al-Khateeb, T., Masud, M.M., Al-Naami, K.M., Seker, S.E., Mustafa, A.M., Khan, L., Trabelsi, Z., Aggarwal, C. and Han, J. Recurring and Novel Class Detection Using Class-Based Ensemble for Evolving Data Stream. *IEEE Transactions on Knowledge and Data Engineering 28*, 10 (October 2016). pp. 2752-2764.

Alpaydin, E. *Classification*. In Introduction to Machine Learning. (2nd ed.). MIT Press, Cambridge, MA, USA, 2010a. pp. 5-9.

Alpaydin, E. *Supervised Learning*. In Introduction to Machine Learning. (2nd ed.). MIT Press, Cambridge, MA, USA, 2010b. pp. 21-45.

Altman, I. *The Environment and Social Behavior: Privacy, Personal Space, Territory, Crowding*. Brooks/Cole Publishing Company, Monterey, CA, USA, 1975.

Altman, I. Privacy - A Conceptual Analysis. *Environment and Behavior 8*, 1 (March 1976). pp. 7-29.

Altman, I. and Chemers, M.M. *Culture and Environment (Environment and Behavior)*. Cambridge University Press, Cambridge, UK, 1984.

Altman, I. and Taylor, D. *Social Penetration: The Development of Interpersonal Relationships*. Holt, Rinehart, and Winston, New York, NY, USA, 1973.

Angluin, D. Queries and Concept Learning. *Machine Learning 2*, 4 (Apr. 1988). pp. 2:319–342.

Aoki, P.M. and Woodruff, A. Making Space for Stories: Ambiguity in the Design of Personal Communication Systems. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 181-190.

AOL Inc. AIM - Chat, Share, Connect. http://www.aim.com/, 2014. (Last accessed: 12/07/2017).

Apache Software Foundation. Daemon: Java based daemons or services. https://commons.apache.org/proper/commons-daemon/, 2013. (Last accessed: 12/07/2017).

Apple Inc. iChat. http://www.apple.com/macosx/apps/all.html#ichat, 2011. (Last accessed: 28/09/2017).

Apple Inc. Apple - iOS 7 - Messages. http://www.apple.com/ios/messages/, 2014. (Last accessed: 22/02/2017).

Asch, S.E. Forming Impressions of Personality. *The Journal of Abnormal and Social Psychology 41*, 4 (1946). pp. 258-290.

Ashbrook, D. Learning Significant Locations and Predicting User Movement with GPS. In *Proceedings of the 6th IEEE International Symposium on Wearable Computers - ISWC 2002* (October, 07-10, 2002, Seattle, WA, USA). IEEE Computer Society, Washington, D.C., USA, 2002. pp. 101-108.

Ashbrook, D. and Starner, T. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. *Personal and Ubiquitous Computing Journal 7*, 5 (October 2003). pp. 275-286.

Atlas, L.E., Cohn, D.A. and Ladner, R.E. Training Connectionist Networks with Queries and Selective Sampling. In *Neural Information Processing Systems* (Nov., 27-30, Denver, CO, USA). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 1989. pp. 566-573.

Avrahami, D., Fogarty, J. and Hudson, S.E. Biases in Human Estimation of Interruptibility: Effects and Implications for Practice. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2007* (Apr. 28 - May 3, San Jose, CA, USA). ACM Press, New York, NY, USA, 2007. pp. 51-60.

Avrahami, D., Fussell, S.R. and Hudson, S.E. IM Waiting: Timing and Responsiveness in Semi-synchronous Communication. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2008* (Nov. 8–12, San Diego, CA, USA). ACM Press, New York, NY, USA, 2008. pp. 285-294.

Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. 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* (Jun. 3-5, Madison, WI, USA). ACM Press, New York, NY, USA, 2002. pp. 1-16.

Bailey, B.P. and Konstan, J.A. On the Need for Attention-Aware Systems: Measuring Effects of Interruption on Task Performance, Error Rate, and Affective State. *Computers in Human Behavior 22*, 4 (July 2006). pp. 685-708.

Bailey, B.P., Konstan, J.A. and Carlis, J.V. The Effects of Interruptions on Task Performance, Annoyance, and Anxiety in the User Interface. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction - INTERACT 2001* (Jul.9-13, Tokyo, Japan). IOS Press, Amsterdam, The Netherlands, 2001. pp. 593-601.

Bao, L. and Intille, S.S. Activity Recognition from User-annotated Acceleration Data. In *Second International Conference on Pervasive Computing - Pervasive 2004* (Apr. 18-23, Linz/Vienna, Austria). Springer, Berlin/Heidelberg, Germany, 2004. pp. 1-17.

Bardram, J.E. and Friday, A. *Ubiquitous Computing Systems*. In Krumm, J., ed. Ubiquitous Computing Fundamentals. CRC Press, Boca Raton, FL, USA, 2009. pp. 37-94.

Bardram, J.E. and Hansen, T.R. The AWARE Architecture: Supporting Context-Mediated Social Awareness in Mobile Cooperation. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, IL, USA). ACM Press, New York, NY, USA, 2004. pp. 192-201.

Barkhuus, L. The Mismeasurement of Privacy: Using Contextual Integrity to Reconsider Privacy in HCI. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2012* (May 5-10, Austin, TX, USA). ACM Press, New York, NY, USA, 2012. pp. 367-376.

Barkhuus, L., Brown, B., Bell, M., Sherwood, S., Hall, M. and Chalmers, M. From Awareness to Repartee: Sharing Location within Social Groups. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2008* (Apr. 5-10, Florence, Italy). ACM Press, New York, NY, USA, 2008. pp. 497-506.

Baron, N.S. *Instant Messaging*. In Herring, S., Stein, D. and Virtanen, T., eds. Pragmatics of Computer-Mediated Communication. De Gruyter, Berlin, Germany, 2013. pp. 135-162.

Barrett, D.J. and Barrett, L.F. An Introduction to Computerized Experience Sampling in Psychology. *Social Science Computer Review 19*, 2 (Summer 2001). pp. 175-185.

Barrett, L.F. and Barrett, D.J. ESP: The Experience Sampling Program. http://www.experience-sampling.org/, 2014. (Last accessed: 26/08/2017).

Beckwirth, R. Designing for Ubiquity: The Perception of Privacy. *Pervasive Computing 2*, 2 (April 2003). pp. 40-46.

Begole, J.B., Matsakis, N.E. and Tang, J.C. Lilsys: Sensing Unavailability. In *Proceedings of the 2004 ACM Conference on Computer-Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, IL, USA). ACM Press, New York, NY, USA, 2004. pp. 511 - 514.

Begole, J.B. and Tang, J.C. Incorporating Human and Machine Interpretation of Unavailability and Rhythm Awareness Into the Design of Collaborative Applications. *Human-Computer Interaction 22*, 1 (May 2007). pp. 7-45.

Begole, J.B., Tang, J.C., Smith, R.B. and Yankelovich, N. Work Rhythms: Analyzing Visualizations of Awareness Histories of Distributed Groups. In *Proceedings of the 2002 ACM Conference on Computer-Supported Cooperative Work - CSCW 2002* (Nov. 16-20, New Orleans, LA, USA). ACM Press, New York, NY, USA, 2002. pp. 334-343.

Bellotti, V. *Design for Privacy in Multimedia Computing and Communication Environments*. In Agre, P.E. and Rotenberg, M., eds. Technology and Privacy: The New Landscape. MIT Press, Cambridge, MA, USA, 1998. pp. 63-98.

Bellotti, V. and Edwards, K. Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction 16*, 2 (December 2001). pp. 193-212.

Bellotti, V. and Sellen, A. Design for Privacy in Ubiquitous Computing Environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work - ECSCW 1993* (Sep. 13-17, Milan, Italy). Kluwer Academic Publishers, Norwell, MA, USA, 1993. pp. 77-92.

Berlage, T. and Sohlenkamp, M. Visualizing Common Artefacts to Support Awareness in Computer-Mediated Cooperation. *Computer Supported Cooperative Work (CSCW) 8*, 3 (September 1999). pp. 207-238.

Biehl, J., Turner, T., van Melle, W. and Girgensohn, A. myUnity: A New Platform to Support Communication in the Modern Workplace. In *Proceedings of the 19th ACM International Conference on Multimedia - MM 2014* (Nov. 28 – Dec. 1, Scottsdale, AZ, USA). ACM Press, New York, NY, USA, 2011. pp. 801-802.

Biehl, J.T., Rieffel, E.G. and Lee, A.J. When Privacy and Utility are in Harmony: Towards Better Design of Presence Technologies. *Personal and Ubiquitous Computing 17*, 3 (March 2013). pp. 503-518.

Bifet, A. *Adaptive Stream Mining - Pattern Learning and Mining from Evolving Data Streams*. IOS Press, Amsterdam, Netherlands, 2010.

Bifet, A., Holmes, G., Kirkby, R. and Pfahringer, B. Data Stream Mining: A Practical Approach http://sourceforge.net/projects/moa-datastream/files/documentation, 2011. (Last accessed: 20/07/2017).

Birnholtz, J., Dixon, G. and Hancock, J. Distance, Ambiguity and Appropriation: Structures Affording Impression Management in a Collocated Organization. *Computers in Human Behavior 28*, 3 (May 2012a). pp. 1028–1035.

Birnholtz, J., Hancock, J., Smith, M. and Reynolds, L. Understanding Unavailability in a World of Constant Connection. *interactions 19*, 5 (September/October 2012b). pp. 32-35

BlueCove Team. BlueCove. http://bluecove.org/, 2008. (Last accessed: 20/07/2017).

Blum, A. and Mitchell, T. Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory - COLT 1998* (Jul. 24-26, Madison, WI, USA). ACM Press, 1998. pp. 92-100.

Bly, S.A., Harrison, S.R. and Irwin, S. Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment. *Communications of the ACM 36*, 1 (January 1993). pp. 28-46.

Bordes, A., Ertekin, S., Weston, J. and Bottou, L. Fast Kernel Classifiers with Online and Active Learning. *Journal of Machine Learning Research 6* (September 2005). pp. 1579-1619.

Borning, A. and Travers, M. Two Approaches to Casual Interaction over Computer and Video Networks. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1991* (Apr. 28 - June 5, New Orleans, LA, USA). ACM Press, New York, NY, USA, 1991. pp. 13-19.

Borriello, G. The Challenges to Invisible Computing. *IEEE Computer 33*, 11 (November 2000). pp. 123-125.

Boyle, M. and Greenberg, S. The Language of Privacy: Learning from Video Media Space Analysis and Design. *ACM Transactions on Computer-Human Interaction (TOCHI) 12*, 2 (June 2005). pp. 328-370.

Boyle, M., Neustaedter, C. and Greenberg, S. *Privacy Factors in Video-Based Media Spaces*. In Harrison, S., ed. Media Space 20 + Years of Mediated Life. Springer-Verlag London Limited, London, UK, 2009. pp. 97-122.

Bradner, E., Kellogg, W.A. and Erickson, T. The Adoption and Use of 'BABBLE': A Field Study of Chat in the Workplace. In *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW 1999* (Sep. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Norwell, MA, USA, 1999. pp. 139-158.

Breiman, L. Random Forests. *Machine Learning 45*, 1 (October 2001). pp. 5-32.

Brown, B. and Randell, R. Building a Context Sensitive Telephone: Some Hopes and Pitfalls for Context Sensitive Computing. *Computer Supported Cooperative Work (CSCW) 13*, 3-4 (August 2004). pp. 329–345.

Bryll, R., Gutierrez-Osuna, R. and Quek, F. Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets. *Pattern Recognition 36*, 6 (June 2003). pp. 1291-1302.

Cachucho, R., Meeng, M., Vespier, U., Nijssen, S. and Knobbe, A. Mining Multivariate Time Series with Mixed Sampling Rates. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 2014* (Sep. 13-17, Seattle, WA, USA). ACM Press, New York, NY, USA, 2014. pp. 413-423.

Cambridge Dictionaries Online. "-ible". http://dictionary.cambridge.org/dictionary/british/ible, 2014. (Last accessed: 04/08/2017).

Camp, J. and Connelly, K. *Beyond Consent: Privacy in Ubiquitous Computing (Ubicomp)*. In Acquisti, A., Gritzalis, S., Lambrinoudakis, C. and di Vimercati, S., eds. Digital Privacy: Theory, Technologies, and Practices. Auerbach Publications, Boca Raton, FL, USA, 2007. pp. 327-343.

Carew, P.J. and Stapleton, L. Towards a Privacy Framework for Information Systems Development. In *Proceedings of the Thirteenth International Conference on Information Systems Development - ISD 2004* (Sep. 9-11, Vilnius, Lithuania). Springer, Berlin/Heidelberg, Germany, 2004. pp. 77-88.

Carnegie Mellon University. CMU Sphinx - Open Source Toolkit For Speech Recognition. http://cmusphinx.sourceforge.net/, 2011. (Last accessed: 28/09/2017).

Cerulean Studios. Trilian. https://www.trillian.im/, 2014. (Last accessed: 22/02/2017).

Chalmers, D. *Sensing and Systems in Pervasive Computing—Engineering Context Aware Systems*. Springer, Berlin/Heidelberg, Germany, 2011.

Chalmers, M. A Historical View of Context. *Computer Supported Cooperative Work (CSCW) 13*, 3-4 (August 2004). pp. 223-247.

Chen, C.-Y., Forlizzi, J. and Jennings, P. ComSlipper: An Expressive Design to Support Awareness and Availability In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2006* (Apr. 22-27, Montreal, Canada). ACM Press, New York, NY, USA, 2006. pp. 369-374.

Chen, S.Y., Macredie, R.D., Liu, X. and Sutcliffe, A. Data Mining for Understanding User Needs. *ACM Transactions on Computer-Human Interaction 17*, 1 (March 2010). pp. 1:1-1:6.

Cheverst, K., Davies, N., Mitchell, K. and Efstratiou, C. Using Context as a Crystal Ball: Rewards and Pitfalls. *Personal Ubiquitous Computing 5*, 1 (February 2001). pp. 8-11.

Cleary, J.G. and Trigg, L.E. K*: An Instance-based Learner Using an Entropic Distance Measure. In *Proceedings of the 12th International Conference on Machine Learning - ICML 1995* (Jul. 9-12, Tahoe City, CA, USA). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 1995. pp. 108-114.

Cohn, D. *Active Learning*. In Sammut, C. and Webb, G.I., eds. Encyclopedia of Machine Learning and Data Mining. Springer US, New York, NY, USA, 2017. pp. 9-14.

Cohn, D., Atlas, L. and Ladner, R. Improving Generalization with Active Learning. *Machine Learning 15*, 2 (May 1994). pp. 201-221.

Cohn, G., Stuntebeck, E.P., Pandey, J.N., Otis, B.P., Abowd, G.D. and Patel, S.N. SNUPI: Sensor Nodes Utilizing Powerline Infrastructure. In *Proceedings of 12th International Conference on Ubiquitous Computing - UbiComp 2010* (Sep. 26-29, Copenhagen, Denmark). ACM Press, New York, NY, USA, 2010. pp. 159-168.

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

Consolvo, S. and Walker, M. Using the Experience Sampling Method to Evaluate Ubicomp Applications. *Pervasive Computing 2*, 2 (April 2003). pp. 24 -31.

Cousot, S. and Stanley, D.E. OpenCV Processing and Java Library. http://www.ubaa.net/shared/processing/opencv/, 2011. (Last accessed: 27/07/2017).

Crafty Apps EU. Tasker - Total Automation for Android. http://tasker.dinglisch.net, 2016 (Last accessed: 25/08/2017).

Crisman, P.A., ed. *The Compatible Time-Sharing System - A Programmer's Guide*. (2nd ed.). MIT Press, Cambridge, MA, USA, 1969.

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

Culnan, M.J. Protecting Privacy Online: Is Self-regulation Working? *Journal of Public Policy & Marketing 19*, 1 (Spring 2000). pp. 20-26.

Cutrell, E.B., Czerwinski, M. and Horvitz, E. Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction - INTERACT 2001* (Jul.9-13, Tokyo, Japan). IOS Press, Amsterdam, The Netherlands, 2001. pp. 263–269.

Cutrell, E.B., Czerwinski, M. and Horvitz, E. Effects of Instant Messaging Interruptions on Computing Tasks. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2000* (Apr. 1-6, The Hague, The Netherlands). ACM Press, New York, NY, USA, 2000. pp. 99-100.

Czerwinski, M., Cutrell, E.B, and Horvitz, E. Instant Messaging and Interruption: Influence of Task Type on Performance. In *Proceedings of OZCHI 2000* (Dec. 4-8, Sydney, Australia). CHISIG, North Ryde, NSW. Australia, 2000a. pp. 356-361.

Czerwinski, M., Cutrell, E.B. and Horvitz, E. Instant Messaging: Effects of Relevance and Timing. In *Proceedings of the 18th British HCI Group Annual Conference - HCI 2000* (Sep. 5-8, Sunderland, UK). Springer, London , UK, 2000b. pp. 71-76.

Dargie, W. and Poellabauer, C. *Fundamentals of Wireless Sensor Networks*. John Wiley & Sons Ltd, Chichester, UK, 2010.

Davis, S. and Gutwin, C. Using Relationship to Control Disclosure in Awareness Servers. In *Proceedings of Graphics Interface 2005 - GI 2005* (May 9-11, Victoria, Canada). A K Peters Ltd., Wellesley, MA, 2005. pp. 145-152.

Day, M., Rosenberg, J. and Sugano, H. A Model for Presence and Instant Messaging (RFC 2778). http://www.ietf.org/rfc/rfc2778.txt, 2000. (Last accessed: 7/07/2017).

DellaFera, A., Eichin, M.W., French, R.S., Jedlinsky, D.C., Kohl, J.T. and Sommerfeld, W.E. The Zephyr Notification Service. In *Proceedings of Usenix Winter 1988 Technical Conference* (Feb. 9-12, Dallas, TX, USA). 1988. pp. 213-219.

Dempster, A.P., Laird, N.M. and Rubin, D.B. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39*, 1 (1977). pp. 1-38.

Derlaga, V.J. and Berg, J.H., eds. *Self-Disclosure: Theory, Research and Therapy*. Plenum Press, New York, NY, USA, 1987.

Dey, A.K., Abowd, G.D. and Salber, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction 16*, 2-4 (December 2001). pp. 97-166.

Dey, A.K. and Häkkilä, J. *Context-Awareness and Mobile Devices*. In Lumsden, J., ed. Handbook of Research on User Interface Design and Evaluation for Mobile Technology. IGI Global, Hershey, PA, USA, 2008. pp. 205-217.

Diacakis, A. and Cohen, D. Patent US 20020116461 A1 - Presence and Availability Management System. http://www.google.com/patents/US20020116461, 2002. (Last accessed: 13/08/2017).

Dillon, A. and Watson, C. User Analysis in HCI—The Historical Lessons from Individual Differences Research. *International Journal of Human-Computer Studies 45*, 6 (December 1996). pp. 619-637.

Dourish, P. and Anderson, K. Collective Information Practice: Exploring Privacy and Security as Social and Cultural Phenomena. *Human-Computer Interaction 21*, 3 (December 2006). pp. 319-342.

Dourish, P. and Bell, G. Yesterday's Tomorrows: Notes on Ubiquitous Computing's Dominant Vision. *Personal and Ubiquitous Computing 11*, 2 (February 2007). pp. 133-143.

Dourish, P. and Bellotti, V. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work - CSCW 1992* (Oct. 31 - Nov. 4, Toronto, Canada). ACM Press, New York, NY, USA, 1992. pp. 107-114.

Dourish, P. and Bly, S. Portholes: Supporting Awareness in a Distributed Work Group. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1992* (May 3-7, Monterey, CA, USA). ACM Press, New York, NY, USA, 1992. pp. 541-547.

Draper, J.V., Kaber, D.B. and Usher, J.M. Telepresence. *Human Factors: The Journal of the Human Factors and Ergonomics Society 40*, 3 (September 1998). pp. 354-375.

Duh, H.B.-L., Tan, G.C.B. and Chen, V.H.-h. Usability Evaluation for Mobile Device: A Comparison of Laboratory and Field Tests. In *Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2006* (Sep. 19-22, Helsinki, Finland). ACM Press, New York, NY, USA, 2006. pp. 181-186.

Encarnação, J.L. and Aarts, E. *Into Ambient Intelligence*. In Encarnação, J.L. and Aarts, E., eds. True Visions - The Emergence of Ambient Intelligence. Springer, Berlin/Heidelberg, Germany, 2006. pp. 1-6.

EPIC - Electronic Privacy Information Center. *Privacy and Human Rights 2001: An International Survey of Privacy Laws and Developments* EPIC - Electronic Privacy Information Center, 2001.

Erickson, T. and Kellogg, W.A. Social Translucence: An Approach to Designing Systems That Support Social Processes. *ACM Transactions on Computer-Human Interaction (TOCHI) 7*, 1 (March 2000). pp. 59 - 83.

Facebook Inc. Facebook Chat. https://www.facebook.com/sitetour/chat.php, 2016. (Last accessed: 22/02/2016).

Facebook Inc. Facebook. http://www.facebook.com, 2014. (Last accessed: 12/09/2017).

Farhoomand, A.F. and Drury, D.H. Managerial Information Overload. *Communications of the ACM 45*, 10 (October 2002). pp. 127-131.

Fetter, M. and Gross, T. Time, Space, Connection: Scaling Ambient Intelligence. *UPGRADE - The European Journal for the Informatics Professional VIII*, 4 (August 2007). pp. 44-49.

Fetter, M. and Gross, T. Contact Management on the Wall: A Card-Game Metaphor for Large Displays. In *Proceedings of the Second International Conference on Tangible Embedded Interaction - TEI 2008* (Feb. 18-20, Bonn, Germany). ACM Press, New York, NY, USA, 2008. pp. 247-250.

Fetter, M. and Gross, T. Beyond the Dyad: Understanding Sharing in Instant Messaging. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2009* (Apr. 4-9, Boston, MA, USA). ACM Press, New York, NY, USA, 2009a. pp. 4243-4248.

Fetter, M. and Gross, T. LocaRhythms: Real-Time Data Mining for Continuous Detection and Prediction of Stays. In *Proceedings of the 35th EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2009* (Aug. 27-29, Patras, Greece). IEEE Computer Society Press, Los Alamitos, CA, USA, 2009b. pp. 64-71.

Fetter, M. and Gross, T. PILS: Advanced Instant Messaging in e-Learning Based on an Open Implementation. In *Proceedings of the Seventeenth Euromicro Conference on Parallel, Distributed, and Network-Based Processing - PDP 2009* (Feb. 18-20, Weimar, Germany). IEEE Computer Society Press, Los Alamitos, CA, USA, 2009c. pp. 347-354.

Fetter, M. and Gross, T. Neue Werkzeuge fuer die Experience Sampling Methode. In *Mensch & Computer - 11. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2011* (Sep. 11-14, Chemnitz, Germany). Oldenbourg, Munich, Germany, 2011a. pp. 383-386.

Fetter, M. and Gross, T. PRIMIExperience: Experience Sampling via Instant Messaging. In *Proceedings of the 2011 ACM Conference on Computer-Supported Cooperative Work - CSCW 2011* (Mar. 19-23, Hangzhou, China). ACM Press, New York, NY, USA, 2011b. pp. 629-632.

Fetter, M. and Gross, T. LiLoLe —A Framework for Lifelong Learning from Sensor Data Streams for Predictive User Modelling. In *Proceedings of the 5th International Conference on Human-Centered Software Engineering - HCSE 2014* (Sep. 16-18, Paderborn, Germany). Springer, Heidelberg, Germany, 2014. pp. 126-143.

Fetter, M., Gross, T. and Zeller, B. Disclosure Templates: Vorlagen fuer die selektive Freigabe personelicher Informationen (Disclosure Templates: Templates for the Selective Disclosure of Personal Information; in German). In *Mensch & Computer - 8. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2008* (Sep. 7-10, Luebeck, Germany). Oldenbourg, Munich, 2008. pp. 57-66.

Fetter, M., Schirmer, M. and Gross, T. CAESSA: Visual Authoring of Context-Aware Experience Sampling Studies. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2011* (May 7-12, Vancouver, Canada). ACM Press, New York, NY, USA, 2011a. pp. 2341-2346.

Fetter, M., Seifert, J. and Gross, T. Lightweight Selective Availability in Instant Messaging. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2010* (Apr. 10-15, Atlanta, GA, USA). ACM Press, New York, NY, USA, 2010a. pp. 3817-3822.

Fetter, M., Seifert, J. and Gross, T. Vorhersagbarkeit von Selektiver Verfügbarkeit im Instant Messaging (Predictability of Selective Availability in Instant Messaging; in German). In *Mensch & Computer - 10. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2010* (Sep. 12-15, Duisburg, Germany). Oldenbourg, Munich, 2010b. pp. 119-128.

Fetter, M., Seifert, J. and Gross, T. Predicting Selective Availability for Instant Messaging. In *Proceedings of the IFIP TC.13 International Conference on Human-Computer Interaction - INTERACT 2011* (Sep. 5-9, Lisbon, Portugal). Springer, Heidelberg, 2011b. pp. 503-520.

Fischer, J.E., Greenhalgh, C. and Benford, S. Investigating Episodes of Mobile Phone Activity As Indicators of Opportune Moments to Deliver Notifications. In *Proceedings of the 13th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2011* (Aug 30 - Sep. 2, Stockholm, Sweden). ACM Press, New York, NY, USA, 2011. pp. 181-190.

Fischer, J.E., Yee, N., Bellotti, V., Good, N., Benford, S. and Greenhalgh, C. Effects of Content and Time of Delivery on Receptivity to Mobile Interruptions. In *Proceedings of the 12th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2010* (Sep. 7-10, Lisbon, Portugal). ACM Press, New York, NY, USA, 2010. pp. 103-112.

Fisher, D.H. Knowledge Acquisition Via Incremental Conceptual Clustering. *Machine Learning 2*, 2 (September 1987). pp. 139-172.

Fisher, R. and Simmons, R. Smartphone Interruptibility Using Density-Weighted Uncertainty Sampling with Reinforcement Learning. In *10th International Conference on Machine Learning and Applications and Workshops - ICMLA 2011* (Dec. 18-21, Honolulu, HI, USA). IEEE Computer Society Press, Los Alamitos, CA, USA, 2011. pp. 436-441.

Fitzpatrick, G., Kaplan, S., Mansfield, T., Arnold, D. and Segall, B. Supporting Public Availability and Accessibility with Elvin: Experiences and Reflections. *Computer Supported Cooperative Work (CSCW) 11*, 3-4 (September 2002). pp. 447-474.

Fitzpatrick, G., Mansfield, T., Kaplan, S., Arnold, D., Phelps, T. and Segall, B. Augmenting the Workaday World with Elvin. In *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW 1999* (Sep. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Norwell, MA, USA, 1999. pp. 431-450.

Fitzpatrick, G., Parsonwith, S., Segall, B. and Kaplan, S. Tickertape: Awareness in a Single Line. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1998* (Apr. 18-23, Los Angeles, CA, USA). ACM Press, New York, NY, USA, 1998. pp. 281-282.

Fogarty, J. and Hudson, S.E. Toolkit Support for Developing and Deploying Sensor-Based Statistical Models of Human Situations. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2007* (Apr. 28 - May 3, San Jose, CA, USA). ACM Press, New York, NY, USA, 2007. pp. 135-144.

Fogarty, J., Hudson, S.E. and Lai, J. Examining the Robustness of Sensor-Based Statistical Models of Human Interruptibility. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2004* (Apr. 24-29, Vienna, Austria). ACM Press, New York, NY, USA, 2004a. pp. 207-214.

Fogarty, J., Lai, J. and Christensen, J. Presence Versus Availability: The Design and Evaluation of a Context-aware Communication Client. *International Journal of Human-Computer Studies 61*, 3 (September 2004b). pp. 299 - 317.

Fonner, K.L. and Roloff, M.E. Testing the Connectivity Paradox: Linking Teleworkers' Communication Media Use to Social Presence, Stress from Interruptions, and Organizational Identification. *Communication Monographs 79*, 2 (June 2012). pp. 205-231.

Foursquare Labs Inc. foursquare. http://foursquare.com/, 2015. (Last accessed: 11/09/2017).

Fraden, J. *Handbook of Modern Sensors - Physics, Designs, and Applications*. Springer, Berlin/Heidelberg, Germany, 2010.

François, J.-M. Jahmm - An Implementation of HMM in Java. https://code.google.com/archive/p/jahmm/, 2006. (Last accessed: 06/06/2017).

Frederic, S. and Woodrow, H. Boundary Regulation in Social Media. In *Proceedings of the 2012 ACM Conference on Computer-Supported Cooperative Work - CSCW 2012* (Feb. 11-15, Seattle, WA, USA). ACM Press, New York, NY, USA, 2012. pp. 769-778.

Froehlich, J. The myExperience Tool. http://myexperience.sourceforge.net/, 2009. (Last accessed: 26/08/2017).

Froehlich, J., Chen, M.Y., Consolvo, S., Harrison, B. and Landay, J.A. 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 2009* (Jun. 11-13, San Juan, Puerto Rico). ACM Press, New York, NY, USA, 2007. pp. 57-70.

Fuchs, L., Pankoke-Babatz, U. and Prinz, W. Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work - ECSCW 1995* (Sep. 10-14, Stockholm, Sweden). Kluwer Academic Publishers, Norwell, MA, USA, 1995. pp. 247-262.

Fussell, S.R., Kiesler, S., Setlock, L.D. and Scupelli, P. Effects of Instant Messaging on the Management of Multiple Project Trajectories. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2004* (Apr. 24-29, Vienna, Austria). ACM Press, New York, NY, USA, 2004. pp. 191-198.

Fussell, S.R., Kraut, R.E., Lerch, F.J., Scherlis, W.L., McNally, M.M. and Cadiz, J.J. Coordination, Overload and Team Performance: Effects of Team Communication Strategies. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 1998* (Nov. 14-18, Seattle, WA, USA). ACM Press, New York, NY, USA, 1998. pp. 275-284.

Gama, J. *Issues and Challenges in Learning from Data Streams*. In Kargupta, H., Han, J., Yu, P.S., Motwani, R. and Kumar, V., eds. Next Generation of Data Mining. Chapman & Hall/CRC, Tailor & Francis Group, Boca Raton, FL, USA, 2008.

Gama, J. *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC, Boca Raton, FL, USA, 2010.

Gama, J. and Rodriques, P.P. *Data Stream Processing*. In Gama, J. and Gaber, M.M., eds. Learning from Data Streams - Processing Techniques in Sensor Networks. Springer, Berlin/Heidelberg, Germany, 2007. pp. 25-39.

Gambs, S., Killijian, M.-O. and Nunez del Prado Cortez, M. Show Me How You Move and I Will Tell You Who You Are. *Transactions on Data Privacy 4*, 2 (August 2011). pp. 103-126.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1994.

Garrett, R.K. and Danzinger, J.N. IM=Interruption Management? Instant Messaging and Disruption in the Workplace. *Journal of Computer-Mediated Communication 13*, 1 (October 2007). pp. 23-42.

Gaver, W. Sound Support For Collaboration. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW 1991* (Sep. 24-27, Amsterdam, The Netherlands). Kluwer Academic Publishers, Norwell, MA, USA, 1991. pp. 293-308.

Gaver, W., Moran, T., MacLean, A., Lovstrand, L., Dourish, P., Carter, K. and Buxton, B. Realizing a Video Environment: EuroPARC's RAVE System. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1992* (May 3-7, Monterey, CA, USA). ACM Press, New York, NY, USA, 1992. pp. 27-35.

Gellersen, H.-W. and Beigl, M. Ambient Telepresence: Colleague Awareness in Smart Environments. In *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments - MANSE 1999* (Dec. 13, Dublin, Ireland). Springer, Berlin/Heidelberg, Germany, 1999. pp. 80-88.

Gellersen, H.-W., Beigl, M. and Krull, H. The MediaCup: Awareness Technology Embedded in a Everyday Object. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing - HUC 1999* (Sep. 27-29, Karlsruhe, Germany). Springer-Verlag, London, UK, 1999. pp. 308-310.

Goffman, E. On Face-Work: An Analysis of Ritual Elements in Social Interaction. *Psychiatry: Journal for the Study of Interpersonal Processes 18*, 3 (Aug. 1955). pp. 213-231.

Goffman, E. *The Presentation of Self in Everyday Life*. Doubleday Anchor Books, Garden City, NY, USA, 1959.

Goffman, E. *Relations in Public - Microstudies of the Public Order*. Basic Books, New York, NY, USA, 1971.

Gonzalez, V.M. and Mark, G. "Constant, Constant, Multi-tasking Craziness": Managing Multiple Working Spheres. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2004* (Apr. 24-29, Vienna, Austria). ACM Press, New York, NY, USA, 2004. pp. 113-120.

Google. Google+ Hangouts. http://www.google.com/hangouts/, 2014. (Last accessed: 22/02/2017).

Google+. Google+. https://plus.google.com, 2015. (Last accessed: 12/09/2017).

Gray, P. and Salber, D. Modelling and Using Sensed Context Information in the Design of Interactive Applications. In *Proceddings of the IFIP International Conference on Engineering for Human-Computer Interaction - EHCI 2001* (May 11-13, Toronto, Canada). Springer, Berlin/Heidelberg, Germany, 2001. pp. 317-335.

Greenwald, A.G. and Breckler, S.J. *To Whom is the Self Presented*. In Schlenker, B.R., ed. The Self and Social Life. McGraw-Hill, New York, NY, USA, 1985. pp. 126-145.

Grinter, R.E. and Palen, L. Instant Messaging in Teen Life. In *Proceedings of the 2002 ACM Conference on Computer-Supported Cooperative Work - CSCW 2002* (Nov. 16-20, New Orleans, LA, USA). ACM Press, New York, NY, USA, 2002. pp. 21-30.

Grinter, R.E., Palen, L. and Eldridge, M. Chatting with Teenagers: Considering the Place of Chat Technologies in Teen Life. *ACM Transactions on Computer-Human Interaction (TOCHI) 13*, 4 (December 2006). pp. 423-447.

Gross, T. Supporting Effortless Coordination: 25 Years of Awareness Research. *Computer Supported Cooperative Work: The Journal of Collaborative Computing and Work Practices 22*, 4-6 (August 2013). pp. 425-474.

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

Gross, T. and Fetter, M. CoDaMine: Communication Data Mining for Feedback and Control in Ubiquitous Environments. In *Proceedings of the Sixteenth Euromicro Conference on Parallel, Distributed, and Network-Based Processing - PDP 2008* (Feb. 13-15, Toulouse, France). IEEE Computer Society Press, Los Alamitos, CA, USA, 2008. pp. 539-546.

Gross, T. and Fetter, M. PRIMIUnite: Gemeinsame Benutzerkennungen fürs Instant Messaging (PRIMIUnite: Shared User Logins for Instant Messaging; in German). In *Mensch & Computer - 9. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2009* (Sep. 6-9, Berlin, Germany). Oldenbourg, Munich, 2009. pp. 479-482.

Gross, T. and Fetter, M. Disclosure Templates for Selective Information Disclosure. In *Proceedings of the IADIS International Conference on Collaborative Technologies - CT 2010* (July 26-28, Freiburg, Germany). IADIS Press, 2010. pp. 101-108.

Gross, T., Fetter, M. and Paul-Stueve, T. Towards Advanced Social TV in a Cooperative Media Space. *Human-Computer Interaction 24,* 2 (February 2008). pp. 155-173.

Gross, T., Fetter, M. and Seifert, J. CoDaMine: Effiziente soziale Interaktion durch die Analyse der Online-Kommunikation im Instant Text Messaging (CoDaMine: Efficient Social Interaction by Analysing Online Communication in Instant Text Messaging; in German). In *Usability Day - Eintauchen in Medienwelten - uDay VIII 2010* (May 21, Dornbirn, Austria). Pabst Science Publishers, Lengerich, Germany, 2010. pp. 106-112.

Gross, T. and Oemig, C. PRIMI: An Open Platform for the Rapid and Easy Development of Instant Messaging Infrastructures. In *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2005* (Aug. 30-Sep. 3, 2005, Oporto, Portugal). IEEE Computer Society Press, Los Alamitos, CA, USA, 2005a. pp. 460-467.

Gross, T. and Oemig, C. PRIMInality:Towards Human-Centred Instant Messaging Infrastructures. In *Mensch & Computer - 5. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2005* (Sep. 4-7, Linz, Austria). Oldenbourg, Munich, Germany, 2005b. pp. 71-80.

Gross, T. and Oemig, C. From PRIMI to PRIMIFaces: Technical Concepts for Selective Information Disclosure. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications - SEAA 2006* (Aug. 29-Sep.1, Cavtat, Dubrovnik, Croatia). IEEE Computer Society Press, Los Alamitos, CA, USA, 2006. pp. 480-487.

Gross, T., Paul-Stueve, T. and Fetter, M. *Social TV from a Computer-Supported Cooperative Work Perspective*. In Cesar, P., Geerts, D. and Chorianopoulos, K., eds. Social Interactive Television: Immersive Shared Experiences and Perspectives. IGI Global Publishing, Hershey, PA, USA, 2009. pp. 50-66.

Gross, T. and Prinz, W. Awareness in Context: A Light-Weight Approach. In *Proceedings of the Eigth European Conference on Computer-Supported Cooperative Work - ECSCW 2003* (Sep. 14-18, Helsinki, Finland). Kluwer Academic Publishers, Netherlands, 2003. pp. 295-314.

Gross, T. and Specht, M. Awareness in Context-Aware Information Systems. In *Mensch & Computer - 1. Fachuebergreifende Konferenz fuer interaktive und kooperative Medien - M&C 2001* (Mar. 5-8, Bad Honnef, Germany). Oldenbourg, Munich, Germany, 2001. pp. 173-181.

Gross, T., Stary, C. and Totter, A. User-Centered Awareness in Computer-Supported Cooperative Work-Systems: Structured Embedding of Findings from Social Sciences. *International Journal of Human-Computer Interaction 18,* 3 (June 2005). pp. 323-360.

Grudin, J. Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM 37,* 1 (January 1994). pp. 92-105.

Grudin, J. CSCW: History and Focus. *IEEE Computer 27*, 5 (May 1999). pp. 19-26.

Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. Internet of Things (Iot): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems 29*, 7 (September 2013). pp. 1645–1660.

Gutwin, C., Greenberg, S. and Roseman, M. Supporting Awareness of Others in Groupware. In *Companion Proceedings of the Conference on Human Factors in Computing Systems - CHI 1996* (Apr. 13-18, Vancouver, Canada). ACM Press, New York, NY, USA, 1996. pp. 205.

Guyon, I. and Elisseeff, A. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research 3* (March 2003). pp. 1157-1182.

Guyon, I., Nikravesh, M., Gunn, S. and Zadeh, L.A. *Feature Extraction - Foundations and Applications*. Springer, Berlin-Heidelberg, Germany, 2006.

Hall, E.T. *The Hidden Dimension*. Anchor Books, Garden City, NY, USA, 1966.

Hall, M. Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning. In *Proceedings of International Conference on Machine Learning - ICML 2000* (Jun. 29 - Jul. 2, Stanford, CA, USA). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 2000. pp. 359–366.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. The WEKA Data Mining Software: An Update. *SIGKDD Explorations 11*, 1 (July 2009). pp. 10-18.

Hall, M. and Holmes, G. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. *IEEE Transactions on Knowledge and Data Engineering 15*, 3 (May/June 2003). pp. 1437-1447.

Hancock, J., Birnholtz, J., Bazarova, N., Guillory, J., Perlin, J. and Amos, B. Butler Lies: Awareness, Deception, and Design. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2009* (Apr. 4-9, Boston, MA, USA). ACM Press, New York, NY, USA, 2009. pp. 517-526.

Handel, M. and Herbsleb, J.D. What Is Chat Doing in the Workplace? In *Proceedings of the 2002 ACM Conference on Computer-Supported Cooperative Work - CSCW 2002* (Nov. 16-20, New Orleans, LA, USA). ACM Press, New York, NY, USA, 2002. pp. 1-10.

Harper, R.H.R. *Texture. Human Expression in the Age of Communications Overload*. MIT Press, Cambridge, MA, USA, 2010.

Harper, R.H.R. and Hughes, J.A. *"What a F-Ing System! Send 'Em All to the Same Place and then Expect Us to Stop 'Em Hitting": Making Technology Work in Air Traffic Control*. In Button, G., ed. Technology in Working Order: Studies of Work, Interaction, and Technology. Routledge, New York, NY, USA, 1993. pp. 127-144.

Harper, R.H.R., Hughes, J.A. and Shapiro, D.Z. Working in Harmony: An Examination of Computer Technology in Air Traffic Control. In *Proceedings of the First European Conference on Computer-Supported Cooperative Work - ECSCW 1989* (Sep. 13-15, London, UK). 1989. pp. 73-86.

Harr, R. and Kaptelinin, V. Unpacking the Social Dimension of External Interruptions. In *Proceedings of the 2007 International ACM SIGGROUP Conference on Supporting Group Work* - GROUP 2007 (Nov. 4-7, Sanibel Island, USA). ACM Press, New York, NY, USA, 2007. pp. 399-408.

Harr, R. and Wiberg, M. Lost in Translation: Investigating the Ambiguity of Availability Cues in an Online Media Space. *Behaviour & Information Technology 27*, 3 (May-June 2008). pp. 243-262.

Harrison, S. *A Brief History of Media Space Research and Mediated Life*. In Harrison, S., ed. Media Space 20 + Years of Mediated Life. Springer-Verlag London Limited, London, UK, 2009a. pp. 9-16.

Harrison, S. *An Introduction to Media Space*. In Harrison, S., ed. Media Space 20 + Years of Mediated Life. Springer-Verlag London Limited, London, UK, 2009b. pp. 1-8.

Hashimoto, S., Tanaka, T., Aoki, K. and Fujita, K. Estimation of Interruptibility during Office Work Based on PC Activity and Conversation. In *Part III of the Proceedings of the 15th International Conference - HCI International 2013* (Jul. 21-26, Las Vegas, NV, USA). Springer, Berlin/Heidelberg, Germany, 2013. pp. 297-306.

Hausen, D., Boring, S., Lueling, C., Rodestock, S. and Butz, A. StaTube: Facilitating State Management in Instant Messaging Systems In *Proceedings of the Sixth International Conference on Tangible Embedded Interaction - TEI 2012* (Feb.19-22, Kingston, Canada). ACM Press, New York, NY, USA, 2012. pp. 283-290.

Heath, C.C. and Luff, P. Collaboration and Control - Crisis Management and Multimedia Technology in London Underground Line Control Rooms *Computer Supported Cooperative Work (CSCW) 1*, 1-2 (May 1992). pp. 69-94.

Heath, C.C. and Luff, P. *Convergent Activities: Line Control and Passenger Information on the London Underground*. In Engeström, Y. and Middleton, D., eds. Cognition and Communication at Work. Cambridge University Press, Cambridge, UK, 1996. pp. 96-129.

Hektner, J.M., Csikszentmihalyi, M. and Schmidt, J.A. *Experience Sampling Method: Measuring the Quality of Everyday Life*. Sage Publications, Inc., Thousand Oaks, CA, USA, 2006.

Herbsleb, J.D., Atkins, D.L., Boyer, D.G., Handel, M. and Finholt, T.A. Introducing Instant Messaging and Chat in the Workplace. In *Proceedings of the 2002 SIGCHI Conference on Human Factors in Computing Systems - CHI 2002* (Apr. 20-25, Minneapolis, MN, USA). ACM Press, New York, NY, USA, 2002. pp. 171-178.

Hightower, J., Consolvo, S., LaMarca, A., Smith, I. and Hughes, J. Learning and Recognizing the Places We Go. In *Proceedings of 7th International Conference on Ubiquitous Computing - UbiComp 2005* (Sep. 11-14, Tokyo, Japan). Springer, Berlin / Heidelberg, 2005. pp. 159-176.

Hincapié-Ramos, J.D., Voida, S. and Mark, G. A Design Space Analysis of Availability-Sharing Systems. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology - UIST 2011* (Oct. 16-19, Santa Barbara, CA, USA). ACM Press, New York, NY, USA, 2011. pp. 85-96.

Ho, J. and Intille, S.S. Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 909-918.

Ho, T.K. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence 20*, 8 (August 1998). pp. 832-844.

Hong, J.I. and Landay, J.A. An Architecture for Privacy-Sensitive Ubiquitous Computing. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services - MobiSys 04* (Jun. 6-9, Boston, MA, USA). ACM Press, New York, NY, USA, 2004. pp. 177-189.

Höök, K. Steps to Take Before IUIs Become Real. *Interacting with Computers 12*, 4 (February 2000). pp. 409-426.

Horvitz, E. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1999* (May 15-20, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 1999. pp. 159-166.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D. and Rommelse, K. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence - UAI 1998* (Jul.24-26, Madison, WI, USA). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 1998. pp. 256-265.

Horvitz, E., Jacobs, A. and Hovel, D. Attention-Sensitive Alerting. In *Proceedings of the Fifteenth Conference on Uncertainty and Artificial Intelligence - UAI 1999* (Jul. 30 - Aug 1, Stockholm, Sweden). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 1999. pp. 305-313.

Horvitz, E., Kadie, C.M., Paek, T. and Hovel, D. Models of Attention in Computing and Communication: From Principles to Applications. *Communications of the ACM 46*, 3 (March 2003). pp. 52-59.

Horvitz, E. and Kapoor, A. Experience Sampling for Building Predictive User Models: A Comparative Study. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2008* (Apr. 5-10, Florence, Italy). ACM Press, New York, NY, USA, 2008. pp. 657-666.

Horvitz, E., Koch, P. and Apacible, J. BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, IL, USA). ACM Press, New York, NY, USA, 2004. pp. 507 - 510.

Horvitz, E., Koch, P., Kadie, C.M. and Jacobs, A. Coordinate: Probabilistic Forecasting of Presence and Availability. In *Proceedings of the Eighteenth Conference on Uncertainty and Artificial Intelligence - UAI 2002* (Jul. 30 - Aug. 1, Edmonton, Alberta). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 2002. pp. 224-233.

Hsieh, G., Tang, K.P., Yong Low, W. and Hong, J.I. Field Deployment of IMBuddy: A Study of Privacy Control and Feedback Mechanisms for Contextual IM. In *Proceedings of 9th International Conference on Ubiquitous Computing - UbiComp 2007* (Sep. 16-19, Innsbruck, Austria). Springer, Berlin / Heidelberg, 2007. pp. 91-108.

Hudson, J.M., Christensen, J., Kellogg, W.A. and Erickson, T. "I'd Be Overwhelmed, But It's Just One More Thing To Do": Availability and Interruption in Research Management. In *Proceedings of the 2002 SIGCHI Conference on Human Factors in Computing Systems - CHI 2002* (Apr. 20-25, Minneapolis, MN, USA). ACM Press, New York, NY, USA, 2002. pp. 97 - 104.

Hudson, S.E. and Smith, I. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proceedings of the 1996 ACM Conference on Computer-Supported Cooperative Work - CSCW 1996* (Nov. 16-20, Boston, MA, USA). ACM Press, New York, NY, USA, 1996. pp. 248-257.

Hull, R., Neaves, P. and Bedford-Roberts, J. Towards Situated Computing. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers - ISWC 1997* (Oct. 13-14, Cambridge, MA, USA). IEEE Computer Society, Los Alamitos, CA, USA, 1997. pp. 146-153.

Hulten, G., Spencer, L. and Domingos, P. Mining Time-Changing Data Streams. In *Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining - KDD 2001* (Aug. 26-29, San Francisco, CA, USA). ACM Press, New York, NY, USA, 2001. pp. 97-106.

Iachello, G. and Hong, J. End-User Privacy in Human-Computer Interaction. *Foundations and Trends in Human-Computer Interaction 1*, 1 (January 2007). pp. 1-137.

ICQ LLC. ICQ. Everybody, Everywhere. http://www.icq.com/info/, 2013. (Last accessed: 01/07/2017).

Igoe, T. and O'Sullivan, D. *Physical Computing: Sensing and Controlling the Physical World with Computers*. Thomson Course Technology, Boston, MA, USA, 2004.

Ingelbrecht, N., Simpson, R., Milanesi, C., Zimmerman, T. and Jones, N. Gartner Research Glossary of Mobile and Wireless Communications Terminology. https://www.gartner.com/doc/1089314/glossary-mobile-wireless-communications-terminology, 2009. (Last accessed: 22/10/2017).

International Organization for Standardization. DIN EN ISO 924: Ergonomics of Human-System Interaction, 2006. http://www.iso.org/, 2017. (Last accessed: 07/07/2017).

Internet Engineering Task Force. SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE). http://datatracker.ietf.org/wg/simple, 2011. (Last accessed: 07/07/2017).

Intille, S.S., Rondoni, J., Kukla, C., Ancona, I. and Bao, L. A Context-Aware Experience Sampling Tool. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2003* (Apr. 5-10, Fort Lauderdale, FL, USA). ACM Press, New York, NY, USA, 2003. pp. 972-973.

Iqbal, S.T., Adamczyk, P.D., Zheng, X.S. and Bailey, B.P. Towards an Index of Opportunity: Understanding Changes in Mental Workload during Task Execution. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 311-320.

Iqbal, S.T. and Horvitz, E. Disruption and Recovery of Computing Tasks: Field Study, Analysis, and Directions. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2007* (Apr. 28 - May 3, San Jose, CA, USA). ACM Press, New York, NY, USA, 2007. pp. 677-686.

Isaacs, E., Walendowski, A. and Ranganthan, D. Hubbub: A Sound-Enhanced Mobile Instant Messenger that Supports Awareness and Opportunistic Interactions. In *Proceedings of the 2002 SIGCHI Conference on Human Factors in Computing Systems - CHI 2002* (Apr. 20-25, Minneapolis, MN, USA). ACM Press, New York, NY, USA, 2002a. pp. 179-186.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D.J. and Kamm, C. The Character, Functions, and Styles of Instant Messaging in the Workplace. In *Proceedings of the 2002 ACM Conference on Computer-Supported Cooperative Work - CSCW 2002* (Nov. 16-20, New Orleans, LA, USA). ACM Press, New York, NY, USA, 2002b. pp. 11-20.

Ishii, H. and Ullmer, B. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1997* (Mar. 22-27, Atlanta, GA, USA). ACM Press, New York, NY, USA, 1997. pp. 234-241.

Ishii, H., Wisneski, C., Brave, S., Dahley, A., Gorbet, M., Ullmer, B. and Yarin, P. ambientROOM: Integrating Ambient Media with Architectural Space. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1998* (Apr. 18-23, Los Angeles, CA, USA). ACM Press, New York, Ny, USA, 1998. pp. 173-174.

Jameson, A. *Adaptive Interfaces and Agents*. In Sears, A. and Jacko, J.A., eds. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. (2nd ed.). Lawrence Erlbaum Associates, New York, NY, USA, 2006. pp. 433 - 459.

Jett, Q.R. and George, J.M. Work Interrupted: A Closer Look at the Role of Interruptions in Organizational Life. *Academy of Management Review 28*, 3 (July 2003). pp. 494-507.

John, G.H. and Langley, P. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence - UAI 1995* (Aug. 18-20, Montreal, Canada). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995. pp. 338-345.

Johri, A. From a Distance - Impression Formation and Impression Accuracy Among Geographically Distributed Coworkers. *Computers in Human Behavior 28*, 6 (November 2012). pp. 1997–2006.

Jones, M. and Marsden, G. *Mobile Interaction Design*. John Wiley & Sons, Sussex, UK, 2006.

Jourard, S.M. *Self-Disclosure: An Experimental Analysis of the Transparent Self*. Wiley-Interscience, Hoboken, NJ, USA, 1971.

Kalisch, B.J. and Aebersold, M. Interruptions and Multitasking in Nursing Care. *Journal on Quality and Patient Safety 36*, 3 (March 2010). pp. 126-32.

Kang, J.H., Welbourne, W., Stewart, B. and Borriello, G. Extracting Places from Traces of Locations. *ACM SIGMOBILE Mobile Computing and Communications Review 9*, 3 (July 2005). pp. 58-68.

Kapoor, A. and Horvitz, E. Principles of Lifelong Learning for Predictive User Modeling. In *Proceedings of the 11th International Conference on User Modeling - UM 2007* (Jul. 25-29, Corfu, Greece). Springer, Berlin / Heidelberg , Germany, 2007. pp. 37-46.

Kapoor, A., Horvitz, E. and Basu, S. Selective Supervision: Guiding Supervised Learning with Decision-Theoretic Active Learning. In *Proceedings of Twentieth International Joint Conference on Artificial Intelligence - IJCAI 2007* (Jan. 6-12, Hyderabad, India). AAAI Press, Menlo Park, CA, USA, 2007.

Karat, J., Karat, C.-M. and Brodie, C. *Human-Computer Interaction Viewed from the Intersection of Privacy, Security, and Trust*. In Sears, A. and Jacko, J.A., eds. The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. (2nd ed.). Lawrence Erlbaum Associates, New York, NY, USA, 2006. pp. 639-658.

Kay, M., Patel, S.N. and Kientz, J.A. How Good is 85%?: A Survey Tool to Connect Classifier Evaluation to Acceptability of Accuracy. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2015* (Apr. 18-22, Seoul, Republic of Korea). ACM Press, New York, NY, USA, 2015. pp. 347-356.

Keogh, E., Selina, C., Hart, D. and Pazzani, M. *Segmenting Time Series: A Survey and Novel Approach*. In Last, M., Kandel, A. and Bunke, H., eds. Data Mining in Time Series Databases. World Scientific Publishing Co, Singapore, 2003. pp. 1-22.

Kern, N., Antifakos, S., Schiele, B. and Schwaninger, A. A Model for Human Interruptability: Experimental Evaluation and Automatic Estimation from Wearable Sensors. In *Proceedings of the 8th IEEE International Symposium on Wearable Computers - ISWC 04* (Oct. 31- Nov. 3, Arlington, VA, USA). IEEE Computer Society, Los Alamitos, CA, USA, 2004. pp. 158-165

Kern, N. and Schiele, B. Towards Personalized Mobile Interruptibility Estimation. In *Proceedings of the 2nd International Workshop on Location- and Context-Awareness - LoCA 2006* (May 10-11, Dublin, Ireland). Springer, Berlin/Heidelberg, Germany, 2006. pp. 134-150.

Khalil, A. and Connelly, K. Improving Cell Phone Awareness by Using Calendar Information. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction - INTERACT 2005* (Sep. 12-16, Rome, Italy). Springer, Berlin/Heidelberg, Germany, 2005. pp. 588-600.

Kienzle, W. and Chellapilla, K. Personalized Handwriting Recognition via Biased Regularization. In *Proceedings of the 23rd International Conference on Machine Learning - ICML 2006* (Jun. 25-29, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 2006. pp. 457-464.

Kjeldskov, J. *Mobile Interactions in Context: A Designerly Way Toward Digital Ecology*. Morgan & Claypool Publishers, San Rafael, CA, USA, 2014.

Kjeldskov, J. and Skov, M.B. Was It Worth the Hassle?: Ten Years of Mobile HCI Research Discussions on Lab and Field Evaluations. In *Proceedings of the 16th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2014* (Sep. 23-26, Toronto, Canada). ACM Press, New York, NY, USA, 2014. pp. 43-52.

Kleinrock, L. Nomadicity: Anytime, Anywhere in a Disconnected World. *Mobile Networks and Applications 1*, 4 (December 1996). pp. 351-357.

Kleinrock, L. Breaking Loose. *Communications of the ACM 44*, 9 (September 2001). pp. 41-46.

Knijnenburg, B.P., Kobsa, A. and Jin, H. Dimensionality of Information Disclosure Behavior. *International Journal of Human-Computer Studies 71*, 12 (December 2013). pp. 1144-1162.

Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - IJCAI 95* (Aug. 20-25, Montreal, Canada). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995. pp. 1137-1143

Kranzberg, M. Technology and History: "Kranzberg's Laws". *Technology and Culture 27*, 3 (July 1986). pp. 544-560.

Kray, C., Larsen, L.B., Olivier, P., Biemans, M., van Bunningen, A., Fetter, M., Jay, T., Khan, V.-J., Leitner, G., Mulder, I., Mueller, J., Ploetz, T. and de Vallejo, I.L. *Evaluating Ubiquitous Systems with Users (Workshop Summary)*. Presented at Evaluating Ubiquitous Systems with Users at European Conference on Ambient Intelligence - AmI 2007 (Nov. 7-10, Darmstadt, Germany). Springer, 2007. pp. 63-74.

Kubey, R.W. and Csikszentmihalyi, M. *Television and the Quality of Life: How Viewing Shapes Everyday Experience*. Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1990.

Kulesza, T., Burnett, M., Stumpf, S., Wong, W.-K., Das, S., Groce, A., Shinsel, A., Bice, F. and McIntosh, K. Where Are My Intelligent Assistant's Mistakes? A Systematic Testing Approach In *Proceedings of the Third International Symposium on End-User Development - IS-EUD 2011* (Jun. 7-10, Torre Canne (BR), Italy). Springer, Berlin/Heidelberg, Germany, 2011. pp. 171-186.

Kurz, M. and Ferscha, A. Sensor Abstractions for Opportunistic Activity and Context Recognition Systems. In *5th European Conference on Smart Sensing and Context - EuroSSC 2010* (Nov. 14-16, Passau, Germany). Springer, Berlin-Heidelberg, 2010. pp. 135-149.

Lai, J., Levas, A., Chou, P., Pinhanez, C. and Viveros, M. BlueSpace: Personalizing Workspace Through Awareness and Adaptability. *International Journal Human-Computer Studies 57*, 5 (November 2002). pp. 415-428.

Lai, J., Yoshihama, S., Bridgman, T., Podlaseck, M., Chou, P. and Wong, D. MyTeam: Availability Awareness through the use of Sensor Data. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction - INTERACT 2003* (Sep. 1-5, Zurich, Switzerland). IOS Press, Amsterdam, The Netherlands, 2003. pp. 503-510.

Langheinrich, M. Privacy by Design-Principles of Privacy-Aware Ubiquitous Systems. In *Proceedings of Third International Conference on Ubiquitous Computing - UbiComp 2001* (Sep. 30 - Oct. 2, Atlanta, GA, USA). Springer, Berlin / Heidelberg, 2001. pp. 273-291.

Langheinrich, M. *Privacy in Ubiquitous Computing*. In Krumm, J., ed. Ubiquitous Computing Fundamentals. CRC Press, Boca Raton, FL, USA, 2009. pp. 95-160.

Lashkari, Y., Metral, M. and Maes, P. Collaborative Interface Agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence - AAAI 94* (Jul. 31–Aug. 4, Seattle, WA, USA). American Association for Artificial Intelligence, Menlo Park, CA, USA, 1994. pp. 444-449.

Latorella, K.A. Effects of Modality on Interrupted Flight Deck Performance: Implications for Data Link. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting - 1998* (Oct. 5-9, Chicago, IL, USA). Human Factors and Ergonomics Society, 1998. pp. 87-91.

Leary, M.R. and Kowalski, R.M. Impression Management: A Literature Review and Two-Component Model. *Psychological Bulletin 107*, 1 (January 1990). pp. 34-47.

Lederer, S., Hong, J.I., Dey, A.K. and Landay, J.A. Personal Privacy Through Understanding and Action: Five Pitfalls for Designers. *Journal of Personal and Ubiquitous Computing 8*, 6 (May 2004). pp. 440-454.

Lee, A., Girgensohn, A. and Schlueter, K. NYNEX Portholes: Initial User Reactions and Redesign Implications. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work - GROUP 1997* (Nov. 16-19, Phoenix, AZ, USA). ACM Press, New York, NY, USA, 1997. pp. 385-394.

Leonardi, P.M., Treem, J.W. and Jackson, M.H. The Connectivity Paradox: Using Technology to Both Decrease and Increase Perceptions of Distance in Distributed Work Arrangements. *Journal of Applied Communication Research 38*, 1 (2010). pp. 85-105.

Lewis, D.D. and Gale, W.A. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval - SIGIR 1994* (Jul. 3-6, Dublin, Ireland). Springer London, London, UK, 1994. pp. 3-12.

Li, D., Chau, P.Y.K. and Lu, H. Understanding Individual Adoption of Instant Messaging: An Empirical Investigation. *Journal of the Association for Information Systems 6*, 4 (2005). pp. 102–129.

Liebowitz, J. Interruption Management: A Review and Implications for IT Professionals. *IEEE IT Professional 13*, 2 (March/April 2011). pp. 44-48.

Lim, B.Y. and Dey, A.K. Toolkit to Support Intelligibility in Context-Aware Applications. In *Proceedings of 12th International Conference on Ubiquitous Computing - UbiComp 2010* (Sep. 26-29, Copenhagen, Denmark). ACM Press, New York, NY, USA, 2010. pp. 13-22.

Lindley, S.E., Meek, S., Sellen, A. and Harper, R. "It's Simply Integral to What I Do": Enquiries into How the Web is Weaved into Everyday Life. In *Proceedings of the 21st International Conference on World Wide Web* (Apr. 16-20, Lyon, France). ACM Press, New York, NY, USA, 2012. pp. 1067-1076.

Liu, H. and Motoda, H., eds. *Feature Extraction, Construction and Selection - A Data Mining Perspective*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

Liu, H. and Setiono, R. A Probabilistic Approach to Feature Selection: A Filter Solution. In *Proceedings of International Conference on Machine Learning - ICML 1996* (Jul. 3-6, Bari, Italy). Morgan Kaufmann Publishers Inc., San Fransisco, CA, USA, 1996. pp. 319-327.

Loevstrand, L. Being Selectively Aware with the Khronika System. In *Proceedings of the Second European Conference on Computer-Supported Cooperative Work - ECSCW 1991* (Sep. 24-27, Amsterdam, The Netherlands). Kluwer Academic Publishers, Norwell, MA, USA, 1991. pp. 265-277.

Lowry, P., Cao, J. and Everard, A. Privacy Concerns Versus Desire for Interpersonal Awareness in Driving the Use of Self-Disclosure Technologies: The Case of Instant Messaging in Two Cultures. *Journal of Management Information Systems 27*, 4 (Spring 2001). pp. 163-200.

Lukowicz, P., Pentland, A.S. and Ferscha, A. From Context Awareness to Socially Aware Computing. *Pervasive Computing 11*, 1 (Jan-Mar 2012). pp. 32-41.

Lyon, D. and Zureik, E. *Surveillance, Privacy and the New Technology*. In Lyon, D. and Zureik, E., eds. Computers, Surveillance, and Privacy. University of Minesota Press, Mineapolis, MN, USA, 1996. pp. 1-20.

Maddox, P. Testing a Distributed System. *Communications of the ACM 58*, 9 (September 2015). pp. 54-58.

Maes, P. Agents That Reduce Work and Information Overload. *Communications of the ACM 37*, 7 (July 1994). pp. 30-40.

Makimoto, T. and Manners, D. *Digital Nomad*. John Wiley & Sons Ltd, Chichester, UK, 1997.

Mann, S. *Wearable Computing*. In Soegaard, M. and Dam, R.F., eds. The Encyclopedia of Human-Computer Interaction. The Interaction Design Foundation, Aarhus, Denmark, 2013.

Manning, P. *Impression Management*. In Ritzer, G., ed. Encyclopedia of Social Theory Sage Publications Inc. , Thousands Oaks, CA, USA, 2005. pp. 397-398.

Margulis, S.T. Conceptions of Privacy: Current Status and Next Steps. *Journal of Social Issues 33*, 3 (Summer 1977). pp. 5-21.

Mark, G., Gonzalez, V.M. and Harris, J. No Task Left Behind? Examining the Nature of Fragmented Work. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 321-330.

Mark, G., Gudith, D. and Klocke, U. The Cost of Interrupted Work: More Speed and Stress. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2008* (Apr. 5-10, Florence, Italy). ACM Press, New York, NY, USA, 2008. pp. 107-110.

Markopoulos, P., de Ruyter, B. and Mackay, W. *Awareness Systems: Advances in Theory, Methodology and Design*. Springer, Berlin/Heidelberg, Germany, 2009.

Markovitch, S. and Rosenstein, D. Feature Generation Using General Constructor Functions. *Machine Learning 49*, 1 (October 2002). pp. 59-98.

Marmasse, N. and Schmandt, C. A User-Centered Location Model. *Personal and Ubiquitous Computing 6*, 5-6 (December 2002). pp. 318 - 321.

McCrickard, D.S., Catrambone, R., Chewar, C.M. and Stasko, J.T. Establishing Tradeoffs that Leverage Attention for Utility: Empirically Evaluating Information Display in Notification Systems. *International Journal of Human-Computer Studies 58*, 5 (May 2003a). pp. 547-582.

McCrickard, D.S., Chewar, C.M., Somervell, J.P. and Ndiwalana, A. A Model for Notification Systems Evaluation—Assessing User Goals for Multitasking Activity. *ACM Transactions on Computer-Human Interaction (TOCHI) 10*, 4 (December 2003b). pp. 312-338.

McFarlane, D.C. and Latorella, K.A. The Scope and Importance of Human Interruption in Human–Computer InteractionDesign. *Human-Computer Interaction 17*, 1 (March 2002). pp. 1–61.

Merriam-Webster.com. "availability". http://www.merriamwebster.com/dictionary/availability, 2012a. (Last accessed: 24/07/2017).

Merriam-Webster.com. "available". http://www.merriam-webster.com/dictionary/available, 2012b. (Last accessed: 24/07/2017).

Merriam-Webster.com. "presence". http://www.merriam-webster.com/dictionary/presence, 2012c. (Last accessed: 24/07/2017).

Merriam-Webster.com. "present". http://www.merriam-webster.com/dictionary/present, 2012d. (Last accessed: 24/07/2017).

Merriam-Webster.com. "privacy". http://www.merriam-webster.com/dictionary/privacy, 2012e. (Last accessed: 24/07/2017).

Merriam-Webster.com. "instant messaging". http://www.merriamwebster.com/dictionary/instant messaging, 2013a. (Last accessed: 24/07/2017).

Merriam-Webster.com. "interrupt". http://www.merriam-webster.com/dictionary/interrupt, 2013b. (Last accessed: 24/07/2017).

Merriam-Webster.com. "ubiquitous". http://www.merriam-webster.com/dictionary/ubiquitous, 2014. (Last accessed: 24/07/2017).

Metts, S. and Grohskopf, E. *Impression Management: Goals, Strategies, and Skills*. In Greene, J.O. and Burleson, B.R., eds. Handbook of Communication and Social Interaction Skills. Lawrence Erlbaum Associates, Mahwah, NJ, USA, 2003.

Microsoft. MSN.com. http://www.msn.com, 2014. (Last accessed: 21/07/2017).

Mileswki, A.E. and Smith, T.M. Providing Presence Cues to Telephone Users. In *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work - CSCW 2000* (Dec. 2-6, Philadelphia, PA, USA). ACM Press, New York, NY, USA, 2000. pp. 89-96.

Mintzberg, H. The Manager's Job: Folklore and Fact. *Harvard Business Review 53*, July–August (1975). pp. 49-61.

Mitchell, T.M. *Machine Learning*. WCB/McGraw-Hill, Boston, MA, USA, 1997.

Miyata, Y. and Norman, D.A. *Psychological Issues in Support of Multiple Activities*. In Norman, D.A. and Draper, S.W., eds. User Centered System Design: New Perspectives on Human-computer Interaction (1st ed.). Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1986. pp. 265-284.

Moskovitch, R. and Shahar, Y. Classification-driven Temporal Discretization of Multivariate Time Series. *Data Mining and Knowledge Discovery 29*, 3 (July 2015). pp. 871-913.

Mu, X., Ming, K. and Zhou, Z.-H. Classification under Streaming Emerging New Classes: A Solution using Completely-random Trees. *IEEE Transactions on Knowledge and Data Engineering 29*, 8 (August 2017). pp. 1605-1618.

Myers, G.J., Badgett, T., Thomas, T.M. and Sandler, C. *The Art of Software Testing*. John Wiley & Sons Inc., Hoboken, NJ, USA, 2004.

Mynatt, E.D. and Tullio, J. Inferring Calendar Event Attendance. In *Proceedings of the 6th International Conference on Intelligent User Interfaces - IUI 2001* (Jan. 14-17, Santa Fe, NM, USA). ACM Press, New York, NY, USA, 2001. pp. 121-128.

Nagel, K.S., Hudson, J.M. and Abowd, G.D. Predictors of Availability in Home Life Context-Mediated Communication. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work - CSCW 2004* (Nov. 6-10, Chicago, IL, USA). ACM Press, New York, NY, USA, 2004. pp. 497-506.

Nardi, B.A., Whittaker, S. and Bradner, E. Interaction and Outeraction: Instant Messaging in Action. In *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work - CSCW 2000* (Dec. 6-10, Philadelphia, PA, USA). ACM Press, New York, NY, USA, 2000. pp. 79 - 88.

National Institute of Standards and Technology. IEEE 1451 Smart Transducer Interface Standards - Definitions. http://www.nist.gov/el/isd/ieee/definitions.cfm, 2011. (Last accessed: 12/08/2017).

Navin Lal, T., Chapelle, O., Weston, J. and Elisseeff, A. *Embedded Methods*. In Guyon, I., Nikravesh, M., Gunn, S. and Zadeh, L.A., eds. Feature Extraction - Foundations and Applications. Springer, Berlin-Heidelberg, Germany, 2006. pp. 137-165.

Newell, P.B. Perspectives on Privacy. *Journal of Environmental Psychology 15*, 2 (June 1995). pp. 87-104.

Nichols, J., Wobbrock, J.O., Gergle, D. and Forlizzi, J. Mediator and Medium: Doors as Interruption Gateways and Aesthetic Displays. In *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques - DIS 2002* (Jun. 25-28, London, England). ACM Press, New York, NY, USA, 2002. pp. 379-386.

Nippert-Eng, C.E. *Home and Work: Negotiating Boundaries Through Everyday Life.* University of Chicago Press, 1996, Chicago, IL, USA, 1996.

Nippert-Eng, C.E. Privacy in the United States: Some Implications for Design. *International Journal of Design 1*, 2 (August 2007). pp. 1-10.

Nissenbaum, H. Privacy as Contextual Integrity. *Washington Law Review 79*, 1 (2004). pp. 119-158.

Norman, D.A. How Might People Interact with Agents. *Communications of the ACM 37*, 7 (July 1997). pp. 68-71.

O'Conaill, B. and Frohlich, D. Timespace in the Workplace: Dealing with Interruptions. In *Companion Proceedings of the Conference on Human Factors in Computing Systems - CHI 1995* (May 7-11, Denver, CO, USA). ACM Press, New York, NY, USA, 1995. pp. 262-263

Oikarinen, J. and Reed, D. Internet Relay Chat Protocol. http://tools.ietf.org/rfc/rfc1459.txt, 1993. (Last accessed: 04/08/2017).

Okoshi, T., Tsubouchi, K., Taji, M., Ichikawa, T. and Tokuda, H. Attention and Engagement-Awareness in the Wild: A Large-Scale Study with Adaptive Notifications. In *IEEE International Conference on Pervasive Computing and Communications - PerCom 2017* (Mar. 13-17, Kailua-Kona, HI, USA). IEEE Computer Society, Los Alamitos, CA, USA, 2017. pp. 100-110.

Olson, J.S., Grudin, J. and Horvitz, E. A Study of Preferences for Sharing and Privacy. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 1985-1988.

Open Geospatial Consortium Inc. OGC KML. http://www.opengeospatial.org/standards/kml/, 2017. (Last accessed: 06/06/2017).

Open Mobile Alliance Ltd. OMA Instant Messaging and Presence Service V1.3. http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/imps-v1-3, 2014. (Last accessed: 03/09/2017).

Oulasvirta, A. *Social Inference Through Technology*. In Markopoulos, P., de Ruyter, B. and Mackay, W., eds. Awareness Systems: Advances in Theory, Methodology and Design. Springer, Berlin/Heidelberg, Germany, 2009. pp. 125-147.

Oulasvirta, A. and Hornbæk, K. HCI Research as Problem-Solving. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2015* (May 7-12, San Jose, CA, USA). ACM Press, New York, NY, USA, 2016. pp. 4956-4967.

Oulasvirta, A., Raento, M. and Titta, S. ContextContacts: Re-designing SmartPhone's Contact Book to Support Mobile Awareness and Collaboration. In *Proceedings of the 7th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2005* (Sep. 19-22, Salzburg, Austria). ACM Press, New York, NY, USA, 2005. pp. 167-174.

Palen, L. and Dourish, P. Unpacking "Privacy" for a Networked World. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2003* (Apr. 5-10, Fort Lauderdale, FL, USA). ACM Press, New York, NY, USA, 2003. pp. 129-136.

Parasuraman, R., Sheridan, T.B. and Wickens, C.D. A Model for Types and Levels of Human Interaction with Automation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 30*, 3 (May 2000). pp. 286-297.

Parnin, C. and Rugaber, S. Resumption Strategies for Interrupted Programming Tasks. *Software Quality Control 19*, 1 (March 2011). pp. 5-34.

Patel, K., Fogarty, J., Landay, J.A. and Harrison, B. Investigating Statistical Machine Learning as a Tool for Software Development. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2008* (Apr. 5-10, Florence, Italy). ACM Press, New York, NY, USA, 2008. pp. 667-676.

Patil, S. and Kobsa, A. Instant Messaging and Privacy. In *Proceedings of the 18th British HCI Group Annual Conference - HCI 2004* (Sep. 6-10, Leeds, UK). Springer, London, UK, 2004. pp. 85-88.

Patil, S. and Kobsa, A. Uncovering Privacy Attitudes and Practices in Instant Messaging. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work* - GROUP 2005 (Nov. 6-9, Sanibel Island, USA). ACM Press, New York, NY, USA, 2005. pp. 109-112.

Patil, S. and Kobsa, A. *Privacy Considerations in Awareness Systems: Designing with Privacy in Mind*. In Markopoulos, P., de Ruyter, B. and Mackay, W., eds. Awareness Systems: Advances in Theory, Methodology and Design. Springer, Berlin/Heidelberg, Germany, 2009. pp. 187-206.

Patil, S. and Lai, J. Who Gets to Know What When: Configuring Privacy Permissions in an Awareness Application. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2005* (Apr. 2-7, Portland, OR, USA). ACM Press, New York, NY, USA, 2005. pp. 101-110.

Patterson, D.J., Baker, C., Ding, X., Kaufman, S.J., Liu, K. and Zaldivar, A. Online Everywhere: Evolving Mobile Instant Messaging Practices. In *Proceedings of 10th International Conference on Ubiquitous Computing - UbiComp 2008* (Sep. 21-24, Seoul, Korea). ACM Press, New York, NY, USA, 2008. pp. 64-73.

Patterson, D.J., Ding, X., Kaufman, S.J., Liu, K. and Zaldivar, A. An Ecosystem for Learning and Using Sensor-Driven IM Status Messages. *Pervasive Computing 8*, 4 (October-December 2009). pp. 42-49.

Pedersen, E.R. Calls.calm: Enabling Caller and Callee to Collaborate In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2001* (Mar. 31 - Apr. 5, Seattle, WA, USA). ACM Press, New York, NY, USA, 2001. pp. 235-236.

Pedersen, E.R. and Sokoler, T. AROMA: Abstract Representation Of Presence Supporting Mutual Awareness. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1997* (Mar. 22-27, Atlanta, GA, USA). ACM Press, New York, NY, USA, 1997. pp. 51-58.

Peek, N., Pitman, D. and The, R. Hangsters: Tangible Peripheral Interactive Avatars for Instant Messaging. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* (Feb. 16-18, Cambridge, UK). ACM Press, New York, NY, USA, 2003. pp. 25-26.

Pejovic, V. and Musolesi, M. InterruptMe: Designing Intelligent Prompting Mechanisms for Pervasive Applications. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 2014* (Sep. 13-17, Seattle, WA, USA). ACM Press, New York, NY, USA, 2014. pp. 897-908.

Perkins, S. and Theiler, J. Online Feature Selection using Grafting. In *Proceedings of the 20th International Conference on Machine Learning - ICML 2003* (Aug. 21-24, Washington, D.C, USA). The AAAI Press, Menlo Park, CA, USA, 2003. pp. 592-599.

Perry, M., O'hara, K., Sellen, A., Brown, B. and Harper, R. Dealing with Mobility: Understanding Access Anytime, Anywhere. *ACM Transactions on Computer-Human Interaction 8*, 4 (December 2001). pp. 323-347.

Petronio, S.S. *Boundaries of Privacy: Dialectics of Disclosure*. State University of New York Press, Albany, NY, USA, 2002.

Philipose, M., Fishkin, K.P., Perkowitz, M., Patterson, D.J., Fox, D., Kautz, H. and Hähnel, D. Inferring Activities from Interactions with Objects. *Pervasive Computing 3*, 4 (October 2004). pp. 50-57.

Pielot, M. Large-Scale Evaluation of Call-Availability Prediction. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 2014* (Sep. 13-17, Seattle, WA, USA). ACM Press, New York, NY, USA, 2014. pp. 933-937.

Pielot, M., Church, K. and de Oliveira, R. An In-situ Study of Mobile Phone Notifications. In *Proceedings of the 16th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2014* (Sep. 23-26, Toronto, ON, Canada). ACM Press, New York, NY, USA, 2014. pp. 233-242.

Platt, J.C. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. In Advances in Kernel Methods - Support Vector Learning. MIT Press, 1999. pp. 185-208.

Poikselkä, M. and Mayer, G. *The IMS: IP Multimedia Concepts and Services*. John Wiley & Sons Ltd, Chichester, UK, 2009.

Poppinga, B., Heuten, W. and Boll, S. Sensor-Based Identification of Opportune Moments for Triggering Notifications. *Pervasive Computing 12*, 1 (January-March. 2014). pp. 22-29.

Post, R.C. Three Concepts of Privacy. *Georgetown Law Journal 89* (2001). pp. 2087-2098.

Pousman, Z. and Stasko, J.T. A Taxonomy of Ambient Information Systems: Four Patterns of Design. In *Proceedings of the Working Conference on Advanced Visual Interfaces - AVI 2006* (May 23-26, Venezia, Italy). ACM Press, New York, NY, USA, 2006. pp. 67-74.

Prinz, W. NESSIE: An Awareness Environment for Cooperative Settings. In *Proceedings of the Sixth European Conference on Computer-Supported Cooperative Work - ECSCW 1999* (Sep. 12-16, Copenhagen, Denmark). Kluwer Academic Publishers, Norwell, MA, USA, 1999. pp. 391-410.

Prinz, W. and Gross, T. Modelling Shared Contexts in Cooperative Environments. *Computer Supported Cooperative Work (CSCW) 13*, 3-4 (August 2004). pp. 283-303.

Prosser, W.L. Privacy. *California Law Review 48*, 3 (August 1960). pp. 383-423.

Quinlan, J.R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, USA, 1993.

Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE 77*, 2 (February 1989). pp. 257-286.

Raento, M. and Oulasvirta, A. Designing for Privacy and Self-Presentation in Social Awareness. *Personal and Ubiquitous Computing 12*, 7 (October 2008). pp. 527-542.

Ramirez-Loaiza, M.E., Sharma, M., Kumar, G. and Bilgic, M. Active Learning: An Empirical Study of Common Baselines. *Data Mining and Knowledge Discovery 31*, 2 (March 2017). pp. 287-313.

Ramorez-Gallego, S., Krawczyk, B., Garcoa, S., Wozniak, M. and Herrera, F. A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions. *Neurocomputing 239* (May 2017). pp. 39–57.

Reiman, J.H. Driving to the Panopticon: A Philosophical Exploration of the Risks to Privacy Posed by the Highway Technology. *Santa Clara Computer & High Technology Law Journal 11*, 1 (1995). pp. 27-44.

Reis, H.T. *Self-Disclosure*. In Kazdin, A.E., ed. Encyclopedia of Psychology - Volume 7. Oxford University Press, London, England, 2000. pp. 210-212.

Reiss, A. and Stricker, D. Personalized Mobile Physical Activity Recognition. In *Proceedings of the 2013 International Symposium on Wearable Computers - ISWC 2013* (Sep. 9-12, Zurich, Switzerland). ACM Press, New York, NY, USA, 2013. pp. 25-28.

Reynolds, L., Smith, M., Birnholtz, J. and Hancock, J. Butler Lies from Both Sides: Actions and Perceptions of Unavailability Management in Texting. In *Proceedings of the 2013 ACM Conference on Computer-Supported Cooperative Work - CSCW 2013* (Feb. 23-27, San Antonio, TX, USA). ACM Press, New York, NY, USA, 2013. pp. 769-778.

Rice, A.C. and Beresford, A.R. Dependability and Accountability for Context-aware Middleware Systems. In *Proceedings of the fourth IEEE International Conference on Pervasive Computing and Communications Workshops - PERCOMW 2006* (Mar. 13-17, Pisa, Italy). IEEE Computer Society, Los Alamitos, TX, USA, 2006. pp. 378-382.

Richards, J. and Christensen, J. People in Our Software. *ACM Queue 1*, 10 (February 2004). pp. 80-86.

Riemer, K. and Frößler, F. Introducing Real-Time Collaboration Systems: Development of a Conceptual Scheme and Research Directions. *Communications of the Association for Information Systems 20* (2007). pp. 204-225.

Riemer, K. and Klein, S. Presence Signalling in Unified Communication Systems – A Framework for Customisation in Context. In *Proceedings of the 9th International Conference Wirtschaftsinformatik - WI 2009* (Feb 25-27, Vienna Austria). Österreichische Computer Gesellschaft (Austrian Computing Society), Vienna Austria, 2009. pp. 265-274.

Rittenbruch, M. and McEwan, G. *An Historical Reflection of Awareness in Collaboration*. In Markopoulos, P., de Ruyter, B. and Mackay, W., eds. Awareness Systems: Advances in Theory, Methodology and Design. Springer, Berlin/Heidelberg, Germany, 2009. pp. 3-48.

Ritter, F.W., Baxter, G.D. and Churchill, E.F. *Foundations for Designing User-Centered Systems-What System Designers Need to Know about People*. Springer-Verlag London Limited, London, UK, 2014.

Roda, C. *Human Attention and its Implications for Human–Computer Interaction*. In Roda, C., ed. Human Attention in Digital Environments. Cambridge University Press, Cambidge, UK, 2011. pp. 11-62.

Roda, C. and Thomas, J. *Attention Aware Systems*. In Ghaoui, C., ed. Encyclopedia of Human Computer Interaction. IGI Global, Hershey, PA, USA, 2006. pp. 38-43.

Rodenstein, R., Abowd, G. and Catrambone, R. OwnTime: A System for Timespace Management. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 1999* (May 15-20, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 1999. pp. 200-201.

Rogers, Y. Moving on from Weiser's Vision of Calm Computing: Engaging Ubicomp Experiences. In *Proceedings of 8th International Conference on Ubiquitous Computing - UbiComp 2006* (Sep. 17-21, Orange County, CA, USA). Springer, Berlin / Heidelberg, 2006. pp. 404-421.

Rokach, L. Ensemble-based Classifiers. *Artificial Intelligence Review 33*, 1-2 (February 2010). pp. 1-39.

Romero, N.A., Markopoulos, P. and Greenberg, S. Grounding Privacy in Mediated Communication. *Computer Supported Cooperative Work (CSCW) 22*, 1 (February 2013). pp. 1-32.

Roobaert, D., Karakoulas, G. and Chawla, N.V. *Information Gain, Correlation and Support Vector Machines* In Guyon, I., Nikravesh, M., Gunn, S. and Zadeh, L.A., eds. Feature Extraction - Foundations and Applications. Springer, Berlin-Heidelberg, Germany, 2006. pp. 463-470.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E. SIP: Session Initiation Protocol (RFC 3261). http://www.ietf.org/rfc/rfc3261.txt, 2002. (Last accessed: 23/08/2017).

Rosenthal, S., Dey, A.K. and Veloso, M. Using Decision-theoretic Experience Sampling to Build Personalized Mobile Phone Interruption Models. In *Proceedings of the 9th International International Conference on Pervasive Computing - Pervasive 2011* (Jun. 12-15, San Francisco, CA, USA). Springer, Berlin/Heidelberg, Germany, 2011. pp. 170-187.

Ruppenthal, L. and Chignell, M. Profiling Users of a Unified Communication Service: Understanding Communication Traits and Styles. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 46*, 12 (September 2002). pp. 993-997.

Sahami Shirazi, A., Henze, N., Dingler, T., Pielot, M., Weber, D. and Schmidt, A. Large-Scale Assessment of Mobile Notifications. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2014* (Apr. 26-May 1, Toronto, Canada). ACM Press, New York, NY, USA, 2014. pp. 3055-3064.

Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core (RFC 6120). http://www.ietf.org/rfc/rfc6120.txt, 2011a. (Last accessed: 02/07/2017).

Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence (RFC 6121). http://www.ietf.org/rfc/rfc6121.txt, 2011b. (Last accessed: 02/11/2017).

Salber, D., Dey, A.K. and Abowd, G.D. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1999* (May 15-20, Pittsburgh, PA, USA). ACM Press, New York, NY, USA, 1999. pp. 434-441.

Sarker, H., Sharmin, M., Ali, A.A., Rahman, M.M., Bari, R., Hossain, S.M. and Kumar, S. Assessing the Availability of Users to Engage in Just-in-time Intervention in the Natural Environment. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp 2014* (Sep. 13-17, Seattle, WA, USA). ACM Press, New York, NY, USA, 2014. pp. 909-920.

Satyanarayanan, M. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications 8*, 4 (August 2001). pp. 10-17.

Schein, A.I., Popescul, A., Ungar, L.H. and Pennock, D.M. Methods and Metrics for Cold-start Recommendations. In *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval - SIGIR 2002* (Aug. 11-15, Tampere, Finland). ACM Press, New York, NY, USA, 2002. pp. 253-260.

Schiele, B. and Kern, N. Context-Aware Notification for Wearable Computing. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers - ISWC 2003* (Oct. 21-23, White Plains, NY, USA). IEEE Computer Society, Los Alamitos, CA, USA, 2003. pp. 223-230

Schilit, B.N., Adams, N. and Want, R. Context-Aware Computing Applications. In *First Workshop on Mobile Computing Systems and Applications - WMCSA 1994* (Dec. 8-9, Santa Cruz, CA, USA). IEEE Computer Society, 1994. pp. 85-90.

Schirmer, M. and Gross, T. Lightweight Editing of Distributed Ubiquitous Environments: The CollaborationBus Aqua Editor. *International Journal of Distributed Systems and Technologies 2*, 4 (October-December 2011). pp. 57-73.

Schlenker, B.R. *Impression Management*. In Kazdin, A.E., ed. Encyclopedia of Psychology - Volume 4. Oxford University Press, London, England, 2000. pp. 236-237.

Schmidt, A. *Context-Aware Computing: Context-Awareness, Context-Aware User Interfaces, and Implicit Interaction*. In Soegaard, M. and Dam, R.F., eds. The Encyclopedia of Human-Computer Interaction. The Interaction Design Foundation, Aarhus, Denmark, 2013.

Schmidt, A., Beigl, M. and Gellersen, H.-W. There is More to Context than Location. *Computers & Graphics 23*, 6 (December 1999). pp. 893-901.

Schmidt, K. The Problem with 'Awareness' - Introductory Remarks on 'Awareness in CSCW'. *Computer Supported Cooperative Work: The Journal of Collaborative Computing and Work Practices 11*, 3 (November 2002). pp. 285-298.

Schmidt, K. *Cooperative Work and Coordinative Practices - Contributions to the Conceptual Foundations of Computer-Supported Cooperative Work (CSCW)*. Springer-Verlag London Limited, Lonodn, UK, 2011.

Settles, B. *Active Learning*. Morgan & Claypool Publishers, San Rafael, CA, USA, 2012.

Sheridan, T.B. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA, USA, 1992.

Sheridan, T.B. Further Musings on the Psychophysics of Presence. In *Proceedings of the 1994 IEEE International Conference on Systems, Man, and Cybernetics* (Oct. 2-5, San Antonio, TX, USA). IEEE Computer Society Press, Los Alamitos, CA, USA, 1994. pp. 1073-1077.

Shneiderman, B. and Maes, P. Direct Manipulation vs. Interface Agents *interactions 4*, 6 (Nov./Dec. 1997). pp. 42-61.

Short, J., Williams, E. and Christie, B. *The Social Psychology of Telecommunications*. John Wiley & Sons Ltd, Chichester, UK, 1976.

Silver, D.L., Yang, Q. and Li, L. Lifelong Machine Learning Systems: Beyond Learning Algorithms. In *Proceedings of 2013 AAAI Spring Symposium: Lifelong Machine Learning* (Mar. 25-27, Stanford, CA, USA). AAAI Press, Palo Alto, CA, USA, 2013.

Simmel, A. *Privacy is not an Isolated Freedom*. In Pennock, J.R. and Chapman, J., eds. Privacy & Personality. (Rpt. in In Pennock, J.R. and Chapman, J., eds. Privacy. Privacy & Personality. Transaction Publishers, 2009. 2nd Paperpack ed.). Atherton Press, New York, NY, USA, 1971. pp. 71-87.

Simmel, G. *Die Großstädte und das Geistesleben*. In Petermann, T., ed. Die Grossstadt - Vorträge und Aufsätze zur Städteausstellung. V. Zahn & Jaensch, Dresden, Germany, 1903. pp. 185-206.

Simmel, G. *The Metropolis and Mental Life*. In Bridge, G. and Watson, S., eds. The Blackwell City Reader. Blackwell Publishing Ltd., Malden, MA, USA, 2002. pp. 11-19.

Skype Limited. Skype. http://www.skype.com/, 2011. (Last accessed: 28/09/2017).

Snyder, M. Self-Monitoring of Expressive Behavior. *Journal of Personality and Social Psychology 30*, 4 (1974). pp. 526-537.

Sohlenkamp, M. and Chwelos, G. Integrating Communication, Cooperation, and Awareness: The DIVA Virtual Office Environment. In *Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work - CSCW 1994* (Oct. 22 - 26, Chapel Hill, NC, USA). ACM Press, New York, NY, USA, 1994. pp. 331-343.

Solove, D.J. Conceptualizing Privacy. *California Law Review 90*, 4 (July 2002). pp. 1087-1155.

Speier, C., Valacich, J.S. and Vessey, I. The Effects of Task Interruption and Information Presentation on Individual Decision Making. In *Proceedings of the 18th International Conference on Information Systems - ICIS 1997* (Dec. 14-17, Atlanta, GA, USA). Association for Information Systems, Atlanta, GA, USA, 1997. pp. 21-36.

Spencer, D. *Card Sorting: Designing Usable Categories*. Rosenfeld Media, New York, NY,USA, 2009.

Spiekermann, S. *Perceived Control: Scales for Privacy in Ubiquitous Computing*. In Acquisti, A., Gritzalis, S., Lambrinoudakis, C. and di Vimercati, S., eds. Digital Privacy: Theory, Technologies, and Practices. Auerbach Publications, Boca Raton, FL, USA, 2007. pp. 267-282.

Steeves, V. *Reclaiming the Social Value of Privacy*. In Kerr, I., Steeves, V. and Lucock, C., eds. Lessons from the Identiy Trail - Anonymity, Privacy and Identity in a Networked Society Oxford University Press, New York, NY, USA, 2009. pp. 191-208.

Stefanowski, J. and Brzezinski, D. *Stream Classification*. In Sammut, C. and Webb, G.I., eds. Encyclopedia of Machine Learning and Data Mining. Springer US, New York, NY, USA, 2017. pp. 1191-1199.

Stolterman, E. and Wiberg, M. Concept-Driven Interaction Design Research. *Human-Computer Interaction 25*, 2 (June 2010). pp. 95-118.

Streitz, N. and Nixon, P. The Disappearing Computer - Introduction. *Communications of the ACM 48*, 3 (March 2005). pp. 32-35.

Stumpf, S., Das, S., Shinsel, A., Bice, F., McIntosh, K., Wong, W.-K., Burnett, M., Zhang, C., Shamasunder, S., Kulesza, T. and Groce, A. You Are the Only Possible Oracle - Effective Test Selection for End Users of Interactive Machine Learning Systems. *IEEE Transactions on Software Engineering 40*, 3 (March 2014). pp. 307-323

SUN Microsystems Inc. JSR 82: Java APIs for Bluetooth. https://jcp.org/en/jsr/detail?id=82, 2010. (Last accessed: 20/09/2017).

Sun, X. and May, A. A Comparison of Field-Based and Lab-Based Experiments to Evaluate User Experience of Personalised Mobile Devices. *Advances in Human-Computer Interaction Volume 2013* (January 2013). pp. 2:1-2:9.

Symonds, D. MarcoPolo—Context-aware computing for Mac OS X. http://www.symonds.id.au/ marcopolo/, 2007. (Last accessed: 25/05/2017).

Takemae, Y., Chaki, S., Ohno, T., Yoda, I. and Ozawa, S. Analysis of Human Interruptibility in the Home Environment. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2007* (Apr. 28 - May 3, San Jose, CA, USA). ACM Press, New York, NY, USA, 2007. pp. 2681-2686.

Talukder, A.K. and Yavagal, R. *Mobile Computing: Technology, Applications, and Service Creation*. McGraw-Hill, New York, NY, USA, 2006.

Tanaka, T. and Fujita, K. Study of User Interruptibility Estimation Based on Focused Application Switching. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2011* (Mar. 19-23, Hangzhou, China). ACM Press, New York, NY, USA, 2011. pp. 721-724.

Tang, J.C. and Begole, J.B. Beyond Instant Messaging. *ACM Queue 1*, 8 (November 2003). pp. 28-37.

Tang, J.C., Isaacs, E.A. and Rua, M. Supporting Distributed Groups with a Montage of Lightweight Interactions. In *Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work - CSCW 1994* (Oct. 22 - 26, Chapel Hill, NC, United States). ACM Press, New York, NY, USA, 1994. pp. 23-34.

Tang, J.C. and Rua, M. Montage: Providing Teleproximity for Distributed Groups. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 1994* (Apr. 24-28, Boston, MA, USA). ACM Press, New York, NY, USA, 1994. pp. 37-43.

Tang, J.C., Yankelovich, N., Begole, J.B., Van Kleek, M., Li, F. and Bhalodia, J. ConNexus to Awarenex - Extending Awareness to Mobile Users. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2001* (Mar. 31 - Apr. 5, Seattle, WA, USA). ACM Press, New York, NY, USA, 2001. pp. 221 - 228.

Tani, T. and Yamada, S. Estimating User Interruptibility by Measuring Table-Top Pressure. In *Extended Abstracts of the Conference on Human Factors in Computing Systems - CHI 2013* (Apr. 27 - May 2, Paris, France). ACM Press, New York, NY, USA, 2013. pp. 1707-1712.

Tapia, E.M., Intille, S.S. and Larson, K. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In *Proceedings of the 2nd International Conference on Pervasive Computing - Pervasive 2004* (Apr. 18-23, Linz/Vienna, Austria). Springer, Berlin/Heidelberg, Germany, 2004. pp. 158-175.

Teevan, J. and Hehmeyer, A. Understanding How the Projection of Availability State Impacts the Reception of Incoming Communication. In *Proceedings of the 2013 ACM Conference on Computer-Supported Cooperative Work - CSCW 2013* (Feb. 23-27, San Antonio, TX, USA). ACM Press, New York, NY, USA, 2013. pp. 753-758.

Tennenhouse, D. Proactive Computing. *Communications of the ACM 43*, 5 (May 2000). pp. 43-50.

Ter Hofte, G.H. Xensible Interruptions from Your Mobile Phone. In *Proceedings of the 9th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2007* (Sep. 9-12, Singapore). ACM Press, New York, NY, USA, 2007. pp. 178-181.

Thomas, V. and Azmitia, M. Tapping Into the App: Updating the Experience Sampling Method for the 21st Century. *Emerging Adulthood 4*, 1 (February 2016). pp. 60-67.

Thrun, S. *Lifelong Learning Algorithms*. In Thrun, S. and Pratt, L., eds. Learning to Learn. Springer Science+Business Media, LLC, New York, NY, USA, 1998. pp. 181-209.

Thurlow, C. and Poff, M. *Text Messaging*. In Herring, S., Stein, D. and Virtanen, T., eds. Pragmatics of Computer-Mediated Communication. De Gruyter, Berlin, Germany, 2013. pp. 163-190.

Tolmie, P., Crabtree, A., Rodden, T. and Benford, S. "Are You Watching This Film or What?": Interruption and the Juggling of Cohorts. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work - CSCW 2008* (Nov. 8–12, San Diego, CA, USA). ACM Press, New York, NY, USA, 2008. pp. 257-266.

Trafton, G., Altmann, E.M., Brock, D.P. and Mintz, F.E. Preparing to Resume an Interrupted Task: Effects of Prospective Goal Encoding and Retrospective Rehearsal. *International Journal of Human-Computer Studies 58,* 5 (May 2003). pp. 583-603.

Tsandilas, T. and Schrafel, M.C. Usable Adaptive Hypermedia Systems. *New Review of Hypermedia and Multimedia 10,* 1 (June 2004). pp. 5-29.

Tullio, J., Begole, J.B., Horvitz, E. and Mynatt, E.D. Forecasting Presence and Availability. In *Extendend Abstracts of the Conference on Human Factors in Computing Systems - CHI 2004* (Apr. 24-29, Vienna, Austria). ACM Press, New York, NY, USA, 2004. pp. 1713-1714

Tullio, J., Goecks, J., Mynatt, E.D. and Nguyen, D.H. Augmenting Shared Personal Calendars. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology - UIST 2002* (Oct. 27-30, Paris, France). ACM Press, New York, NY, USA, 2002. pp. 11-20.

van Bergen, A. *Task Interruption*. North-Holland Publishing Co., Amsterdam, Netherlands, 1968.

Vertegaal, R. Attentive User Interfaces. *Communications of the ACM 46,* 3 (March 2003). pp. 30-33.

Vertegaal, R., Shell , J.S., Chen, D. and Mamuji, A. Designing for Augmented Attention: Towards a Framework for Attentive User Interfaces. *Computers in Human Behavior 22,* 4 (July 2006). pp. 771-789.

Vetere, F., Smith, J. and Gibbs, M. *Phatic Interactions: Being Aware and Feeling Connected*. In Markopoulos, P., de Ruyter, B. and Mackay, W., eds. Awareness Systems: Advances in Theory, Methodology and Design. Springer, Berlin/Heidelberg, Germany, 2009. pp. 173-186.

Vihavainen, S., Lampinen, A., Oulasvirta, A., Silfverberg, S. and Lehmuskallio, A. The Clash between Privacy and Automation in Social Media. *Pervasive Computing, 13,* 1 (January-March 2014). pp. 56-63.

Viola, P. and Jones, M. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR 2001* (Dec. 8-14, Kauai, HI, USA). IEEE Computer Society Press, Los Alamitos, CA, USA, 2001. pp. 511-518.

Voida, A., Newstetter, W.C. and Mynatt, E.D. When Conventions Collide: The Tensions of Instant Messaging Attributed. In *Proceedings of the 2002 Conference on Human Factors in Computing Systems - CHI 2002* (Apr. 20-25, Minneapolis, MN, USA). ACM Press, New York, NY, USA, 2002. pp. 187 - 194.

Voida, S., Patterson, D.J. and Patel, S.N. *Sensor Data Streams*. In Olson, J.S. and Kellogg, W.A., eds. Ways of Knowing in HCI. Springer, Berlin/Heidelberg, Germany, 2014. pp. 291-321.

Waldo, J., Lin, H.S. and Millett, L.I. Engaging Privacy and Information Technology in a Digital Age: Executive Summary. *Journal of Privacy and Confidentiality 2*, 1 (September 2010). pp. 5-18.

Wang, H., Fan, W., Yu, P.S. and Han, J. Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In *Proceedings of the Ninth ACM International Conference on Knowledge Discovery and Data Mining - KDD 2003* (Aug. 24-27, Washington, D.C., USA). ACM Press, New York, NY, USA, 2003. pp. 226-235.

Wang, J., Zhao, P., Hoi, S.C.H. and Jin, R. Online Feature Selection and Its Applications. *IEEE Transactions on Knowledge and Data Engineering* (2013).

Want, R. *An Introduction to Ubiquitous Computing*. In Krumm, J., ed. Ubiquitous Computing Fundamentals. CRC Press, Boca Raton, FL, USA, 2009. pp. 3-35.

Want, R., Hopper, A., Falcão, V. and Gibbons, J. The Active Badge Location System. *ACM Transactions on Information Systems (TOIS) 10*, 1 (January 1992). pp. 91-102.

Ward, J.A., Lukowicz, P. and Troester, G. Gesture Spotting Using Wrist Worn Microphone and 3-Axis Accelerometer In *Proceedings of the 2005 Joint Conference on Smart Objects & Ambient Intelligence - sOc-EUSAI 2005* (Oct. 12-14, Grenoble, France). ACM Press, New York, NY USA, 2005. pp. 99-104.

Ward, J.A., Lukowicz, P. and Tröster, G. Evaluating Performance in Continuous Context Recognition Using Event-Driven Error Characterisation. In *Proceedings of the 2nd International Workshop on Location- and Context-Awareness - LoCA 2006* (May 10-11, Dublin, Ireland). Springer, Berlin/Heidelberg, Germany, 2006. pp. 239-255.

Warren, S.D. and Brandeis, L.D. The Right to Privacy. *Harvard Law Review IV*, 5 (December 1890). pp. 193-220.

Webb, G.I., Pazzani, M.J. and Billsus, D. Machine Learning for User Modeling. *User Modeling and User-Adapted Interaction 11*, 1-2 (2001). pp. 19-29.

Weiser, M. The Computer for the 21st Century. *Scientific American 265*, 3 (Sep. 1991). pp. 94-100.

Weiser, M. The World is not a Desktop. *ACM Transactions on Computer-Human Interaction 1*, 1 (January 1994). pp. 7-8.

Weiser, M. Ubiquitous Computing. http://www.ubiq.com/hypertext/weiser/UbiHome.html, 1996. (Last accessed: 17/09/2017).

Weiser, M. and Brown, J.S. Designing Calm Technology. http://www.ubiq.com/weiser/calmtech/calmtech.htm, 1995. (Last accessed: 01/12/2017).

Werner, C.M., Altman, I. and Brown, B.B. *Privacy*. In Kazdin, A.E., ed. Encyclopedia of Psychology - Volume 6. Oxford University Press, London, England, 2000. pp. 308-310.

Westin, A.F. *Privacy and Freedom*. Atheneum, New York, NY, USA, 1967.

WhatsApp Inc. WhatsApp - Home. http://www.whatsapp.com/, 2014. (Last accessed: 22/02/2017).

Wiberg, M. and Whittaker, S. Managing Availability: Supporting Lightweight Negotiations to Handle Interruptions. *ACM Transactions on Computer-Human Interaction (TOCHI) 12*, 4 (December 2005). pp. 356-387

Widmer, G. and Kubat, M. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning 23*, 1 (April 1996). pp. 69-101

Wiese, J., Biehl, J., Turner, T., van Melle, W. and Girgensohn, A. Beyond 'Yesterday's Tomorrow': Towards the Design of Awareness Technologies for the Contemporary Worker. In *Proceedings of the 13th Conference on Human-Computer Interaction with Mobile Devices and Services - MobileHCI 2011* (Aug 30 - Sep. 2, Stockholm, Sweden). ACM Press, New York, NY, USA, 2011. pp. 455-464.

Willow Garage. OpenCV. http://www.willowgarage.com/pages/software/opencv, 2011. (Last accessed: 02/07/2017).

Winograd, T. Architectures for Context. *Human-Computer Interaction 16*, 2 (December 2001). pp. 401-419.

Witten, I.H., Frank, E. and Hall, M.A. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011a.

Witten, I.H., Frank, E. and Hall, M.A. *Input: Concepts, Instances, and Attributes*. In Data Mining: Practical Machine Learning Tools and Techniques. (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011b.

Wolf, W. Cyber-Physical Systems. *IEEE Computer 42*, 3 (March 2009). pp. 88-89.

Wu, X., Yu, K., Ding, W., Wang, H. and Zhu, X. Online Feature Selection with Streaming Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 5 (May 2013). pp. 1178-1192.

Wu, X., Yu, K., Wang, H. and Ding, W. Online Streaming Feature Selection In *Procedings of the 27th International Conference on Machine Learning - ICML 2010* (June 21-24, Haifa, Israel). Omnipress, Madison, WI, USA, 2010. pp. 1159-1166.

XMPP Standards Foundation. Jabber.org - The Original XMPP Instant Messaging Service. http://www.jabber.org/, 2014. (Last accessed: 24/09/2017).

Xu, A., Biehl, J., Rieffel, E., Turner, T. and van Melle, W. Learning How to Feel Again: Towards Affective Workplace Presence and Communication Technologies In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2012* (May 5-10, Austin, TX, USA). ACM Press, New York, NY, USA, 2012. pp. 839-848.

Yahoo! Inc. Yahoo Messenger - Chat, Instant Messages, SMS, Video Call, PC Calls. http://messenger. yahoo.com/, 2014. (Last accessed: 04/11/2017).

Yang, H., Fong, S., Sun, G. and Wong, R. A Very Fast Decision Tree Algorithm for Real-Time Data Mining of Imperfect Data Streams in a Distributed Wireless Sensor Network. *International Journal of Distributed Sensor Networks 8*, 12 (December 2012). pp. 1-16.

Yea, J., Dobsona, S. and McKeever, S. Situation Identification Techniques in Pervasive Computing: A Review. *Pervasive and Mobile Computing 8*, 1 (Feb. 2012). pp. 36-66.

Yu, K., Ding, W., Simovici, D.A., Wang, H., Pei, J. and Wu, X. Classification with Streaming Features: An Emerging-Pattern Mining Approach. *ACM Transactions on Knowledge Discovery from Data 9*, 4 (June 2015). pp. 30:1-30:31.

Yu, K., Ding, W. and Wu, X. LOFS: A Library of Online Streaming Feature Selection. *Knowledge-Based Systems 113*, C (December 2016). pp. 1-3.

Zhao, Z., Chen, Y., Liu, J., Shen, Z. and Liu, M. Cross-People Mobile-Phone Based Activity Recognition. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - IJCAI 2011* (Jul. 16-22, Barcelona, Catalonia, Spain). AAAI Press, Menlo Park, CA, USA, 2011. pp. 2545-2550.

Zhou, C., Frankowski, D., Ludford, P., Shekhar, S. and Terveen, L. Discovering Personally Meaningful Places: An Interactive Clustering Approach. *ACM Transactions on Information Systems (TOIS) 25*, 3 (July 2007). pp. 12:1-12:31.

Zhou, Z.-H. and Chen, Z.-Q. Hybrid Decision Tree. *Knowledge-Based Systems 15*, 8 (November 2002). pp. 515–528.

Zliobaite, I., Bifet, A., Pfahringer, B. and Holmes, G. Active Learning with Evolving Streaming Data. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - PKDD'11* (Sep. 5-9, Athens, Greece). Springer, 2011. pp. 597-612.

Züger, M., Corley, C., Meyer, A.N., Li, B., Fritz, T., Shepherd, D., Aug.ine, V., Francis, P., Kraft, N. and Snipes, W. Reducing Interruptions at Work: A Large-Scale Field Study of FlowLight. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2017* (May 6-11, Denver, CO, USA). ACM Press, New York, NY, USA, 2017. pp. 61-72.

Züger, M. and Fritz, T. Interruptibility of Software Developers and its Prediction Using Psycho-Physiological Sensors. In *Proceedings of the Conference on Human Factors in Computing Systems - CHI 2015* (Apr. 18-22, Seoul, Republic of Korea). ACM Press, New York, NY, USA, 2015. pp. 2981-2990.

Zukerman, I. and Albrecht, D.W. Predictive Statistical Models for User Modeling. *User Modeling and User-Adapted Interaction (UMUAI) 11*, 1-2 (March 2001). pp. 5-18.

Appendix A — List of Abbreviations

3

3GPP. 3rd Generation Partnership Project

A

AC. Availability Category
AIM. AOL Instant Messenger
AUI. Attentive User Interfaces

B

BSSID. Basic Service Set Identification

C

CMC. Computer-Mediated Communication
CSCW. Computer-Supported Cooperative Work

E

ESM. Experience Sampling Method

G

GUI. Graphical User Interface

H

HCI. Human-Computer Interaction
HTML. HyperText Markup Language

I

ICQ. Homophone for "I seek you" - Instant Messaging Service
ICT. Information and Communication Technology
IE. Inference Engine
IETF. Internet Engineering Task Force
IM. Instant Messaging
IMPS. OMA Instant Messaging and Presence Service
IMS. IP Multimedia Subsystems
IPC. Inter-Process Communication

J

JNI. Java Native Interface

M

MSN. Microsoft Network
MTS. Multivariate Time Series

N

NIST.
National Institute of Standards and Techn ology

O

OMA. Open Mobile Alliance
OS. Operating System
OSA. Open Scripting Architecture
OSCAR. Open System for CommunicAtion in Realtime

P

PARC. Palo Alto Research Center
PASS. PRIMI Advanced Sensor Suite
PRIMI. Platform for Research on Instant Messaging Infrastructures

R

RFC. Request for Comments
RSSI. Received Signal Strength Indication
RTC. Real-Time Collaboration

S

SIMPLE. SIP for Instant Messaging and Presence Leveraging Extensions
SIP. Session Initiation Protocol
SMS. Short Message Service, Short Message Service
SPSS. Statistical Package for the Social Sciences
SSID. Service Set Identifier
SVM. Support Vector Machine

U

UC. Unified Communications

V

VCR. Video Cassette Recorder
VFDT. Very Fast Decision Trees

W

WAP. Wireless Application Protocol
WML. Wireless Markup Language

X

XML. Extensible Markup Language
XML-RPC. Remote Procedure Call, Extensible Markup Language Remote Procedure Call
XMPP. eXtensible Messaging and Presence Protocol

Y

YM. Yahoo Messenger

Appendix B — Examples of Sensor Values

Appendix B.1 — Example of one Sensor Value of a 1×1 Sensor (i.e., Voice Activity)

```
<sensVal sid="VoiceActivity" dimensionality="1x1"
           date="2011-12-09 18:20:40">
    <row>
        <item label="Speech Detected" type="boolean">true</item>
    </row>
</sensVal>
```

Appendix B.2 — Example of one Sensor Value of a 1×m Sensor (i.e., Volume)

```
<sensVal sid="Volume" dimensionality="1xm"
           date="2011-12-09 10:19:08">
    <row>
        <item label="output-vol" type="integer">6</item>
        <item label="input-vol" type="integer">21</item>
        <item label="alert-vol" type="integer">100</item>
        <item label="muted" type="boolean">false</item>
    </row>
</sensVal>
```

Appendix B.3 — Example of one Sensor Value of an n×1 Sensor (i.e., Connected USB)

```
<sensVal sid="Connected USB" dimensionality="nx1"
           date="2011-12-09 18:33:59">
    <row>
        <item label="USB Dev." type="string">FaceTime HD Camera
          (Built-in)</item>
    </row><row>
        <item label="USB Dev." type="string">BRCM2070 Hub</item>
    </row><row>
        <item label="USB Dev." type="string">Apple Optical USB
          Mouse</item>
    </row>
</sensVal>
```

Appendix B.4 — Example of one Sensor Value of a n×m Sensor (i.e., WiFiSensor)

```
<sensVal sid="WiFiSensor" dimensionality="nxm"
           date="2011-12-09 10:21:55">
    <row>
        <item label="BSSID" type="string">0:1f:33:64:c6:80</item>
        <item label="RSSI" type="integer">185</item>
        <item label="SSID" type="string">UniBamberg-802.1X</item>
```

```
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:88</item>
    <item label="RSSI" type="integer">179</item>
    <item label="SSID" type="string">UniBamberg-802.1X</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8f</item>
    <item label="RSSI" type="integer">179</item>
    <item label="SSID" type="string">Dominikanerkirche</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8e</item>
    <item label="RSSI" type="integer">179</item>
    <item label="SSID" type="string">UniBamberg</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8d</item>
    <item label="RSSI" type="integer">178</item>
    <item label="SSID" type="string">VHB-Gastzugang</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8c</item>
    <item label="RSSI" type="integer">179</item>
    <item label="SSID" type="string">VHB</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8b</item>
    <item label="RSSI" type="integer">178</item>
    <item label="SSID" type="string">NEPS</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:8a</item>
    <item label="RSSI" type="integer">179</item>
    <item label="SSID" type="string">Tagung</item>
</row><row>
    <item label="BSSID" type="string">0:1f:33:64:c6:89</item>
    <item label="RSSI" type="integer">178</item>
    <item label="SSID" type="string">DFNRoaming</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e0</item>
    <item label="RSSI" type="integer">176</item>
    <item label="SSID" type="string">UniBamberg-802.1X</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e7</item>
    <item label="RSSI" type="integer">176</item>
    <item label="SSID" type="string">Dominikanerkirche</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e6</item>
    <item label="RSSI" type="integer">176</item>
    <item label="SSID" type="string">UniBamberg</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e5</item>
    <item label="RSSI" type="integer">177</item>
    <item label="SSID" type="string">VHB-Gastzugang</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e4</item>
    <item label="RSSI" type="integer">177</item>
    <item label="SSID" type="string">VHB</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e3</item>
    <item label="RSSI" type="integer">177</item>
    <item label="SSID" type="string">NEPS</item>
</row><row>
    <item label="BSSID" type="string">0:22:3f:61:fa:e2</item>
```

```
            <item label="RSSI" type="integer">176</item>
            <item label="SSID" type="string">Tagung</item>
        </row><row>
            <item label="BSSID" type="string">0:22:3f:61:fa:e1</item>
            <item label="RSSI" type="integer">176</item>
            <item label="SSID" type="string">DFNRoaming</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:87</item>
            <item label="RSSI" type="integer">188</item>
            <item label="SSID" type="string">Dominikanerkirche</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:86</item>
            <item label="RSSI" type="integer">184</item>
            <item label="SSID" type="string">UniBamberg</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:85</item>
            <item label="RSSI" type="integer">185</item>
            <item label="SSID" type="string">VHB-Gastzugang</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:84</item>
            <item label="RSSI" type="integer">185</item>
            <item label="SSID" type="string">VHB</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:83</item>
            <item label="RSSI" type="integer">185</item>
            <item label="SSID" type="string">NEPS</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:81</item>
            <item label="RSSI" type="integer">184</item>
            <item label="SSID" type="string">DFNRoaming</item>
        </row><row>
            <item label="BSSID" type="string">0:1f:33:64:c6:82</item>
            <item label="RSSI" type="integer">184</item>
            <item label="SSID" type="string">Tagung</item>
        </row>
    </sensVal>
```

Appendix C — SensBatch-XML Schema

```
<xs:element name="sensBatch">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="sensSet" nillable="true" minOccurs="0"
         maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="sensVal" nillable="true"
                 minOccurs="0" maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="row" nillable="true"
                       minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="item"
                             maxOccurs="unbounded">
                          <xs:complexType>
                            <xs:simpleContent>
                              <xs:extension base="xs:string">
                                <xs:attribute name="type"
                                     type="xs:string"/>
                                <xs:attribute name="label"
                                     type="xs:string"/>
                              </xs:extension>
                            </xs:simpleContent>
                          </xs:complexType>
                        </xs:element>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="date" type="xs:dateTime"/>
                <xs:attribute name="sid" type="xs:string"/>
                <xs:attribute name="dimensionality"
                    type="xs:string"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Appendix D — ESMconfig-XML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="esmconfig">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="projectname" type="xs:string"
             minOccurs="0"/>
        <xs:element name="sid" type="xs:string"/>
        <xs:element name="requesttimeout" type="xs:integer"/>
        <xs:element name="requestinterval" type="xs:integer"/>
        <xs:element name="question">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="query" type="xs:string"/>
              <xs:element name="description" type="xs:string"/>
              <xs:element name="single" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="value"
                        maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="multiple" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element ref="value"
                       maxOccurs="unbounded"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="quantity" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="range">
                      <xs:complexType>
                        <xs:sequence/>
                        <xs:attribute name="to" type="xs:integer"
                           use="required"/>
                        <xs:attribute name="from"
                           type="xs:integer" use="required"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="character" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="size" type="xs:integer"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
```

```
              <xs:element name="boolean" type="xs:anyType"
            minOccurs="0"/>
             </xs:sequence>
           </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="value">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="code" type="xs:string"
           use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schem
```

University of Bamberg Press

Modern Computer-mediated Communication technologies like Instant Messaging (IM) systems enable spontaneous communication over distance. With the advances in Mobile and Ubiquitous Computing, these technologies move away from the desktop computers of our offices, and become more and more pervasive and interwoven with our daily lives. The introduction of these great possibilities to communicate from everywhere with everyone however comes at a cost: The cost of constantly being available to everybody, everywhere, leading to an increasing number of interruptions in our daily tasks. The challenge is, that current technology does not empower users to manage their availability in an adequate manner. Most IM clients for example, only support one single online status that needs to be managed manually by the user.

In this work I am founding the concepts of Presence and Availability on a deep understanding of human privacy needs, derived from literature. Based on this foundation, I show how the selective and dynamic nature of privacy is not sufficiently reflected in current systems. Based on two user studies I reveal patterns for selective information disclosure and present an analysis of Selective Availability needs. With the collected study data, I further show that Selective Availability for nomadic users can be predicted based on sensors installed on the users' laptop computer with a good accuracy through machine learning. As the personalised nature of the data requires new concepts for building an adaptive system, I introduce the LILOLE Framework. The LILOLE Framework outlines the concept of an adaptive system that relies on stream-based active learning to continuously learn and automatically adapt fine-grained personal availability preferences for individual users. The concept is validated through a proof-of-concept implementation and an evaluation based on real user data.

In comparison to related work, the presented work is one of very few examples that goes beyond the pure analysis of the predictability, but provides a concept and an implementation of a real system as validation. My approach is novel by combining concepts from Data Stream Mining and Active Learning to predict availability, thus making it very flexible for different settings. This way I am able to address the selective and dynamic nature of availability preferences for nomadic users.

eISBN: 978-3-86309-624-3

9 783863 096243

www.uni-bamberg.de/ubp