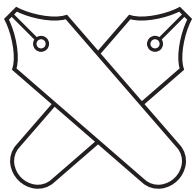
Doctoral Dissertation – Academic Year 2015

Detection of User's Interruptibility for Attention Awareness in Ubiquitous Computing



A dissertation for the degree of
Ph.D. in Media and Governance

Graduate School of Media and Governance
Keio University

Tadashi Okoshi

Thesis Abstract – Academic Year 2015

Detection of User's Interruptibility for Attention Awareness in Ubiquitous Computing

There has been an explosion of information available for people to read and act on in the age of ubiquitous computing. Users computing experience have been getting all-day long, carrying and using an increasing number of mobile and wearable devices with an increasing number of applications, and being connected to more number of remote users. Notification, a side channel for pushing information from a computer to a user has been a taking on greater importance in such computing, with increasing versatility in the notification source, an increasing length of notification experiences, and an increasing number of devices as the notification destination. On the other hand, a human user's attention resource with a limited amount of capacity is, however, remaining constant. This research addresses the problem of **interruption overload**, a situation in which too many ill-timed interruptions by notifications delivered to the user in an as-soon-as-possible manner cause the user's divided attention and negatively affect their performance.

What is fundamentally needed in computer systems is **attention-awareness**, particularly the fundamental functionality of attention sensing. This dissertation shows that the breakpoint of user's activity, as an interruptible timing that lowers the user's perceived workload while preserving their limited attention resource, can be sensed in real-time, in a mobile and wearable multi-device environment without external psycho-physiological sensors, and without modifications to the existing operating systems and applications. The design and the implementation of "Attelia", the first middleware that realizes such detection, are proposed along with an extensive evaluation through user studies on the participants' real mobile and wearable environment. The evaluation validates the effectiveness of Attelia, which results in a significantly lower overhead in the user's workload perception when receiving notifications in the detected breakpoint timing on a smartphone or smart watch, or in a multi-device environment with a combination of such devices.

Keywords: ubiquitous computing, interruption overload, attention-awareness, interruptibility, mobile sensing, mobile multi-device environment

博士論文要旨 – 2015 年度 (平成 27 年度)

ユビキタス・コンピューティングにおけるアテンション・アウェアネスのためのユーザの割り込み可能性検知

ユビキタス・コンピューティングの進展とともに、人々をとりまく情報の量が爆発的に増加している。ユーザは多くのモバイルデバイスやウェアラブルデバイスを携帯・利用し、それらの上で多種多様なアプリケーションを使い、また多くの他のユーザともネットワークを介したコミュニケーションを行う。これらのユーザをとりまくコンピュータ環境によって、コンピューティング体験はユーザの生活をより包括的に支えつつある。コンピュータからのユーザへの情報提供のサイドチャネルである通知は、発信元の多様性の増加、ユーザの通知体験の長時間化、通知先デバイスの増加といった近年の傾向に影響を受け、その重要性を増している。一方で、人間の注意（アテンション）は有限の資源であり、その量は変わらない。本研究は、典型的な既存の通知システムによって「なるべく早く」送信される方式の通知が、過量かつ不適切なタイミングでユーザに割り込みを行う事でユーザの分割的注意能力に悪影響を与え、ユーザタスクの実行効率を低下させる問題である「Interruption Overload」問題に取り組む。

同問題の解決に向けて、計算機システムには「ユーザのアテンションに対する適応性（attention-awareness）」が求められる。同適応性を実現する諸機能の中でも特に、アテンション状態の検知機能は、適応機能、管理機能、予測機能といった他機能実現のために必要であり、アテンション適応性における中核機能として重要である。本研究は、ユーザ注意への負担を抑制し同資源を守ることができる情報通知のタイミングとしての、ユーザ活動の “breakpoint” に着目する。本研究は、breakpoint が実時間で、複数のモバイル・ウェアラブル端末上で外部の生体センサを必要とせず、また既存のオペレーティングシステムや多様なアプリケーションに改変を加えること無く検知できることを示す。本研究では以上の様な検知を行う新しいミドルウェア Attelia を提案し、その設計と実装、および被験者の実環境上で行う広範なユーザ評価実験を行う。実験の結果、モバイルデバイス、ウェアラブルデバイス、およびそれらの組み合わせからなるマルチデバイス環境において、Attelia が検知する breakpoint タイミングでの通知が、通知受信時におけるユーザの負担を有意に抑制できることが判明した。

キーワード: ubiquitous computing, interruption overload, attention-awareness, interruptibility, mobile sensing, mobile multi-device environment

Acknowledgements

At the brink of completing this dissertation, I have found that completing the present research, including this dissertation, has been absolutely one of the most enjoyable periods in my life. Looking back not only on the last two and half years but also on my journey since the year 2000 (when I first entered the Ph.D. program at Keio), the faces of many tremendous people I want to acknowledge come to my mind.

I wish to acknowledge:

Professor Hideyuki Tokuda, for all of his advice, guidance and encouragement for the many years from year 1995 when I firstly met him. It is absolutely my honor to have taken on this Ph.D. research under his supervision.

Professor Yasushi Kiyoki, for his thoughtful advice and encouragement from the viewpoints of database, machine learning, and the global environmental system problems.

Professor Jin Nakazawa, for his consistently useful advice regarding this research. At the same time, I greatly thank Jin Nakazawa for our lengthy and continuous friendship since my high school days.

Professor Anind K. Dey, for his advice regarding my research from an HCI viewpoint. In addition, as a host faculty member of my six-month international GESL training, I thank him greatly for his hospitality at Carnegie Mellon University.

Professor Mahadev Satyanarayanan, for his lengthy support, supervision, and essential advice regarding my research and Ph.D. student life in Carnegie Mellon University since year 2001.

Professors Archan Misra, Rajesh Krishna Balan, and Youngki Lee in the LiveLabs Urban Lifestyle Innovation Platform, School of Information Systems, Singapore Management University, for their kindness in offering me a great research opportunity in Singapore (to rehabilitate my research capabilities), and for their advice and encouragement regarding our joint research, from which I obtained a great deal of knowledge and skills in machine learning, mobile sensing, and mobile programming.

Professor Kazunori Takashio, Professor Takuro Yonezawa and Professor Chen Yin as faculty members in Tokuda Laboratory for their great encouragement, discussion, advice, and other daily enjoyable research experiences.

Professors Jun Murai, Hiroyuki Kusumoto, Osamu Nakamura, Rod Van Meter, Keisuke Uehara, Jin Mitsugi, and Keiji Takeda, as faculty members of RG, for their lengthy and tremendous support and advice since I first entered RG in 1995, nearly 20 years ago.

Professor Yoshito Tobe, for his lengthy advice, supervision, enjoyable discussions, brain storming, and support of my research life since the late 1990s, including his mentorship when I first applied to the graduate program at Carnegie Mellon University in year 2000.

Professor Nobuhiko Nishio, for his lengthy advice, supervision, productive and critical discussions, and support to my research life since the late 1990s.

Professor David A. Eckhardt, for his lengthy and tremendous support, mentoring, and advice regarding my student life as a Ph.D. student in Pittsburgh. During the difficult times of my student life, his thoughtful care was absolutely supportive and encouraging.

Professors Yasushi Kiyoki, Shuichi Kurabayashi, Shiori Sasaki, Asako Uraki, Kanako Morita, Kohei Matsunaga, and Jeremy Hall, as well as all other faculty members at the Global Environmental System Leaders (GESL) Program; Professor Toshihisa Ueda, Professor Osamu Kurita, Professor Takuma Akimoto, Professor Kenji Yasuoka, and all other GESL faculty members in the Graduate School of Science and Technology; and Kayoko Imachi, Ryoko Kuroda, and all other officers in the GESL office, for their tremendous advice, mentorship, and various support during my time in the GESL program. In particular, my six-month intensive research experience in the GESL international training program at Carnegie Mellon University was one of the significant keystones to this research and could not be realized without the support of the GESL program.

Julian Ramos, Hiroki Nozaki, Yu (Victor) Lu, Chetna Vig, Rahul Majethia, and Takuya Takimoto for their great and devoted joint collaborative work on our research, including productive discussions, implementations, data collection and experiments, management and direction of user studies, and data analysis.

Kiyonobu Kojima, Tomotaka Ito, Yutaro Kyono, Yuuki Nishiyama, Masaki Ogawa, Mina Sakamura, and all members of Tokuda laboratory, for their enormous amount of support and daily productive and enjoyable discussions on my (and our) researches. I am honored to be a member of our laboratory.

SeungJun Kim, Sang Won Bae, Sunyoung Cho, Nikola Banovic, Christian Koehler, Adrian de Freitas, Brandon Taylor, Denzil Ferreira, Alaaeddine Yousfi, Shiwei Cheng, and the members of Ubicomp Lab, Human Computer Interaction Institute at Carnegie Mellon University for their kind and productive advice on our joint research, as well as for their friendship and hospitality during my internship at CMU.

Jan Harkes, Yoshihisa Abe, Zhuo Chen, Wenlu Hu, and the Elijah group members for their productive discussions and advice on my research, as well as their hospitality in allowing me to attend their regular research meetings.

Tan Kiat Wee William, Jeena Sebastian, Swetha Gottipati, Kartik Muralidharan, Sougata

Sen, Joseph Chan, Huynh Nguyen, Kasthuri Jayarajah, Azeem Javed Khan, Rijurekha Sen, and all other members at LiveLabs Urban Lifestyle Innovation Platform, School of Information Systems, Singapore Management University, for their very welcoming hospitality, great friendship, and kind contributions to our joint research.

Yuka Matsuo, for her continuous and encouraging support to my life as a Ph.D. student since 2013. Daily enjoyable chats with her in the office, and during lunch-time with folks in the lab, were indispensable relaxation that gave me further energy for my research.

Michiko Nitta, for her lengthy and kind support to my research life. During my younger years in my undergraduate and master's program, her support and guidance was just like having a "mother in the lab."

Jessica Stanley, for her kind administrative support during my six-month stay at Carnegie Mellon.

Tracy Farbacher, for her kind and thoughtful administrative support as well as enjoyable chats during my CSD Ph.D. student life in Carnegie Mellon.

Jonathan Wang Wah Kiat, Huang SiPei, Kazae Quek, Yvonne Mok, and Kenneth Fu Tsing Jin, for their kind hospitality and friendship during my stay in Singapore, as well as for our joint research workshop from which I was able to brush up on my research idea.

Jin Nakazawa, Takeshi Iwamoto, and Tomohiro Nagata, as members of "the four" of rg94 in Tokuda Laboratory, for their lengthy and fantastic friendship and rivalry. Finally, "the final Ph.D." has arrived.

Rajesh Krishna Balan, Yamuna Balan, Dushyanth Narayanan, So Young Park, Jan Harkes, and Jason Flinn, for their great friendship since my first stay in Carnegie Mellon. I can now state "Battery no juden ga kanryo shimashita" in my Ph.D. research.

Shigeya Suzuki, Atsuhi Onoe, Shoko Miyagawa, Ysuke Doi, Masafumi Nakane, Akiko Orita, Akimichi Ogawa, Miyoko Kumaki and all members of Futoru-kai, for their continuous friendship, encouragement, and occasionally nutritious meetings.

Shin'ichi Koizumi, Takefumi Yamashita, Kengo Ando, Shuichi Shibukawa, and Yuki Mototani, for their great, long-term, good and bad friendships since our Keio SFC student ages.

Finally, my parents, for their *limitless* support to my life.

August 17, 2015
Tadashi Okoshi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There has been a huge increase in the amount of information available for people to read and act upon. However, the amount of user attention that can be applied to this growing amount of information has remained constant with a limited capacity [53]. Approaches for dealing with this include multitasking or dividing one's attention among a number of sources, and relying on push notifications to bring information from the background of their attention to the forefront.

However, notifications are responsible for an even greater number of interruptions. This is exacerbated by the fact that users are carrying, wearing, and using a growing number of computing devices including notebook computers, tablets, smartphones, smart watches, or wearable sensors [31, 71], all of which can deliver interruptive notifications. Making the problem even worse is the growing number of applications installed on each device (along with a back-end service running on the cloud), each of which can also interrupt a mobile device owner. In particular, communication-based applications that support phone calls, text chats, and social networking suffer from such interruptions. However, games, news, and other applications also have similar issues. With these diverse types of devices and applications supporting the daily lives of users "ubiquitously," the overall computing of users has become a 24-hour experience, rather than just an 8-hour a day experience when old style computers only supported the user's computing while at work. Thus, the users' experience with interruptive notifications is also becoming an all-day affair.

Each of these trends has contributed to a setting in which the every-day lives of users are significantly impacted [19, 65, 82] by the feeling of being constantly interrupted by such computing systems. This form of distraction caused by the excessive number and inappropriate delivery of notifications from computing systems is defined as "interruption overload."

1.1 The Problem

To address the interruption overload that takes up a user's limited amount of attention, computer systems need to have a capability of "attention-awareness," in which a computer

system uses the status of the user's attention to provide information and/or services to the user in a way that contributes to preserving their precious attention. In particular, among the possible concrete functionalities of attention-awareness such as sensing, adaptation, prediction, and management, **attention sensing** is the first and most challenging research problem because (1) attention sensing literally requires the sensing of a human's internal attention state, and (2) all other functionalities depend on information regarding the status of their sensed attention.

In this thesis, I concentrate particularly on the following key challenge: **how to sense a user's current attention status, which enables adaptive information delivery in a notification system in real-time, in users' mobile and wearable computing situations, and easily.**

Real-time sensing of a solution is important because a system's adaptive behavior needs to be executed in real-time, and not in a post-hoc analysis-based manner. Affinity with a user's *mobile and wearable computing situations* is also crucial because the ubiquity of diverse mobile and wearable devices has become the daily computing experience of users. *Easiness* of the solution, in terms of minimizing the burden of end users and developers in the deployment of a solution, is another significant characteristic. The requirement of additional external devices, such as psycho-physiological sensors, including ECG monitors, can be a big obstacle for a user's day-long use. In addition, requiring modifications to the existing computer systems, such as operating systems and each of their numerous applications, has brought about significant burden to developers, lowering the deployability of a solution.

1.2 Previous Approaches to Attention Sensing

Several different approaches have been taken to sense a user's level of attention, particularly in terms of the current availability or load of the resources.

The first approach is to use various types of psycho-physiological sensors, such as an eye tracker, an ECG-monitor, an EEG headset, and/or a heart rate monitor. Haapalainen *et al.* [32] found that a combinational use of an electrocardiogram and heat flux is the most accurate at classifying low and high levels of cognitive load. Although this approach can detect the load of a resource in real-time, the burden to users in wearing two different sensors is not trivial.

The second approach is to estimate the user's interruptibility based on various types of context information of the user. After an early work by Hudson *et al.* [38] in the field of desktop computing, which constructed statistical models for predicting the interruptibility of office workers in a *posteriori* manner, there have been several studies in the field of mobile computing.

Works by Hofte *et al.* [83] and Pejovic *et al.* [68] addressed interruptibility estimations based on the user's context information collected mainly on the smartphones. However, such context information, such as the emotional status of users and the number of compa-

nies, needs to be input manually and continuously by users, with rooms for improvement remaining in terms of *easiness* of use.

Following the work on desktop computing by Iqbal *et al.* [45], another class of work in the field of mobile computing has been conducted to find the **breakpoint** [63], which according to several studies [2, 42, 43] is the boundary between actions within the user's activity and the timing at which a notification delivery results in a lower cost. Ho *et al.* [34] focused on the breakpoint in a user's physical activity, but their system requires the use of an external on-body sensor. Fischer *et al.* [23] targeted the breakpoint timing immediately after phone-call and texting activities. Although their system showed positive results, the applicability of their system is limited only to a specific class of phone applications.

1.3 Solution: Attelia

As a solution to the present research problem, this study proposes *Attelia*, a middleware that detects the opportune interruptive notification timing of users (concretely, the timing of the breakpoint) on mobile and wearable devices in real-time without the need for external dedicated psycho-physiological sensors or any modifications of the applications running on the devices. Attelia uses the breakpoint as a temporal adaptation target of notification delivery, and uses a mobile machine learning technique with various types of mobile sensing for the breakpoint detection.

My first prototype, Attelia I, is a novel middleware used on a smartphone that identifies the breakpoint timing during the user's manipulation of their smartphone device. Using time-series UI event data during the user's device interaction as the sensor data, and machine-learning based real-time breakpoint classification, this system detects the user's breakpoint while the user is actively manipulating their device. My evaluation proved the effectiveness of Attelia I. The first controlled user study conducted showed that notifications at the detected breakpoint timing resulted in a 46% lower workload perception compared to randomly timed notifications. A second in-the-wild user study with 30 participants that took place over 16 days further validated the value of Attelia I, showing a 33% decrease in workload perception compared to randomly timed notifications.

My second prototype, Attelia II, extended Attelia I by additionally supporting breakpoint detection in multiple mobile user devices and under wearable computing situations, and by supporting breakpoint detection both while the user actively manipulates their device and when they do not. My in-the-wild evaluation in a multiple mobile user device environment (smartphones and smart watches) with 41 participants for a one-month long period proved the effectiveness of the proposed system. The new physical-activity based breakpoint detection, in addition to the UI-event based breakpoint detection, resulted in a 71.8% greater reduction of the user's workload perception compared with my previous system in which only UI events are used. Adding this functionality to a smart watch reduced the workload perception by 19.4% compared to a random timing of notification deliveries. Finally, I demonstrated that my multi-device breakpoint detection method across smartphones and

smart watches reduced the user's workload perception by 31.7%.

1.4 The Thesis Statement

The thesis statement of this research is as follows.

> **As an interruptive notification timing that lowers the user's workload perception overhead in preserving their limited attention resource, the breakpoint of the users in terms of both physical activity and user-device interaction can be sensed in real-time on mobile and wearable devices without the need for external psycho-physiological sensors or modifications to existing operating systems and applications.**

1.5 Contributions

The contributions of this dissertation are three-fold: (1) conceptual contributions, (2) artifacts, and (3) the evaluation results.

Conceptual contribution

The first contribution is the concept of real-time breakpoint detection using sensor data on mobile and wearable devices. To the best of my knowledge, Attelia is the first system that detects such breakpoints in real-time, solely on a smartphone, and not needing external psycho-physiological sensors or modifications to versatile applications on the system. This concept is supported by several software technologies and the concepts proposed in earlier studies, such as machine learning and mobile activity recognition using the sensors on a mobile device. Utilizing a sensor among a rich set of sensors on a powerful device, and using a real-time classification based on the machine learning approach, a periodic execution of classification of real-time sensor data enables such detection of breakpoints, which actually lowered the user's workload perception in my evaluation.

The second contribution is the concept of combinational breakpoint detection using two different types of detection, namely, user interaction-based and physical activity-based breakpoint detections. To the best of my knowledge, Attelia II is the first system on mobile and wearable devices that in-combination and opportunistically uses such multiple types of breakpoints to detect a final conclusive breakpoint, covering the comprehensive daily ubiquitous computing lives of users. Covering the user's real-world life in ubiquitous computing, a combinational use of various types of sensor data is important for several reasons. First, this is because applications in ubiquitous computing have versatility in their form of usage, such as conventional interactive applications where users manipulate the application on the screen in real-time by interactively manipulating the user interface, or various types

of new applications, such as those that utilize embedded sensors and track the user's activities, by running and sensing in the background continuously and rather silently. The user's computing experience (hence, their notification experience as well) has become an all-day long affair, as users carry their mobile and wearable devices, and continuously run diverse types of applications. The combinational use of the two types of breakpoint detection is significant in comprehensively covering such "24/7" computing experience of users.

Artifacts

In this research, two versions of my research prototype, namely Attelia I and II, were developed on the Android platform (both the generic Android 4.3 and Android Wear 5 platforms). As proposed, the Attelia prototypes detect the user's actual breakpoints, and works effectively without modifying the original Android operating systems, or versatile Android applications contributed to by developers, installed on the device, and utilized by the user. Toward further research opportunities on attention-awareness, this artifact will be the fundamental platform on mobile and wearable devices.

Evaluation

An actual evaluation of the system through a series of in-the-wild user studies on a user's real daily computing environment, as well as their results, are yet another contribution of this research. To the best of my knowledge, this research is the first to evaluate the real-time breakpoint detection in single and multi-device ubiquitous computing environments with such extensive "real-world" user studies.

Using Attelia I with user interaction-based breakpoint detection on a smartphone, my 16-day long in-the-wild user study with 30 participants validated the value of my proposal, showing a 33% decrease in workload perception compared to randomly timed notifications.

Another in-the-wild user study on Attelia II, which used 41 participants for a one-month long period, validated the further effectiveness of Attelia in a multi-device environment (smartphones and smart watches). My new physical activity-based breakpoint detection, in addition to user interaction-based breakpoint detection, resulted in a 71.8% greater reduction of user workload perception as compared with my previous system that used UI events only. Adding this functionality to a smart watch reduced the workload perception by 19.4% compared to the random timing of notification deliveries. Finally, I demonstrated that my multi-device breakpoint detection across smartphones and watches reduces the user's workload perception by 31.7%, which is a 295% greater reduction than my own previous system.

1.6 Dissertation Road-map

This dissertation establishes the above thesis through the following steps:

1. First, in Chapter 2, several distinctive key phenomena in ubiquitous computing are specified, namely, the (1) ubiquity of computers around users, (2) ubiquity of applications and services around users, (3) communication connection among users, and (4) user's day-long computing experience.

2. Next, in Chapter 3, notifications used in computing and several of their distinct trends are presented. After interaction models between the user and computer are categorized, the basic concept of notification in computing with the given background is introduced. Finally, the key trends in the notification, namely (1) increasing notifications from versatile sources, (2) using multiple mobile devices as targets, and (3) increasing the length of the notification experience, are specified.

3. Next, in Chapter 4, **attention-awareness** in computing is discussed. After introducing several past studies in the area of attention including its limited capacity and the concept of **divided attention** in the field of cognitive psychology, the **interruption overload** problem is defined. Toward resolving the interruption overload problem, this dissertation discusses attention-awareness as a fundamentally needed capability in computer systems. Finally, attention sensing, the most fundamental part of attention-awareness, is defined along with its requirement in the given background, namely, (1) compatibility with a user's multiple mobile and wearable devices, (2) applicability to diverse types of notification sources, (3) day-long use, and (4) real-time sensing.

4. Next, in Chapter 5, several previous approaches and works on attention sensing are presented, including a psycho-physiological sensor-based approach, interruptibility sensing based on various types of context information, and an approach for finding users' breakpoints at which notification delivery is known to lower the user's workload perception overhead.

5. Then, in Chapter 6, Attelia, my proposal for attention status sensing, is overviewed. In addition, some key features, technical approaches, and a hybrid multi-device breakpoint detection model are presented.

6. Using the first prototype, Attelia I, Chapter 7 indicates that, on a single mobile device, breakpoints during a user's device interaction period can be detected in real-time using the proposed middleware on the mobile device solely, and without any external psycho-physiological sensors or modifications to existing systems and applications. An extensive user study validates the effectiveness of Attelia I, illustrating that notification delivery during breakpoint timing (detected by Attelia) reduces the user's workload perception overhead compared with the overload when delivering notifications through random timing.

7. Using the second prototype, Attelia II, Chapter 8 shows that breakpoints during both the user's device interaction period and non-active manipulation period can be de-

tected in real-time solely with the Attelia middleware on a combination of mobile and wearable devices without the need for any external psycho-physiological sensors or modifications to existing systems and applications. Another user study showed the effectiveness of Attelia II. i.e., (1) Attelia effectively reduces the user's workload perception on smart watches, (2) the additional breakpoint detection of user's physical activity improves Attelia's performance, and (3) the combinational use of multiple breakpoint detection across multiple devices further improves Attelia's level of performance.

8. Finally, Chapter 9 provides concluding remarks regarding the present research and clarifies some areas of future work.

Chapter 2

Background: Ubiquitous Computing

This section describes the background of this research, ubiquitous computing. After briefly introducing the concept of ubiquitous computing, which in a broader sense is the target computing area of the present research, I will introduce several different recent key phenomena.

2.1 Ubiquitous Computing

Ubiquitous computing [88] is a concept in the computer science field in which computing occurs "everywhere," supported by various types of computing devices and their networks, which exist "ubiquitously."

> *"The most profound technologies are those that disappear."*
>
> *"Consider writing, perhaps the first information technology."*
>
> *"Today this technology is ubiquitous in industrialized countries."*
>
> *"The constant background presence of these products of "literacy technology" does not require active attention."*
>
> Weiser [89]

Weiser used the electric motor as another example of a ubiquitous technology in current society [89] to explain the concept of ubiquity. Typical industrial facilities, such as factories and workshops, used to apply single engines to provide motive power to numerous machines through physical systems, such as shafts and pulleys. The innovation of the small electric motor, with efficiency and a cheap price, enabled such machines to contain their own inner motive power source. After a while, such machines eventually started to be equipped with multiple electric motors inside them. Nowadays, vehicles are typically equipped with 40 to 100 electric motors of their own [13].

The Third Wave

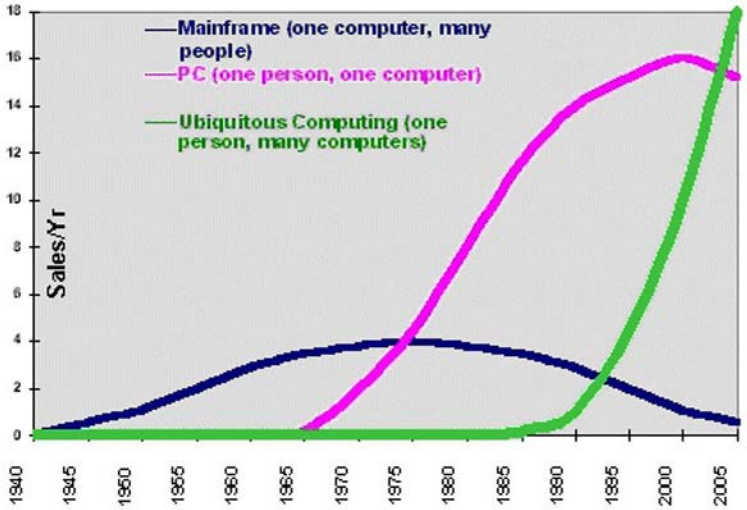
> *"Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop."*
>
> Weiser [89]

Ubiquitous computing is said to be the "third wave in computing" [90], coming after the first wave of mainframe computing and the second wave of personal computing, as shown in Figure 2.1.

Mainframe computing is a situation in which large numbers of people share a single computer. In personal computing, in contrast, "you have your own computer" [90]. Although it has often been said that mobile phones, including smartphones, are representative examples of the third ubiquitous computing wave, it is natural for them to be classified as a part of personal computing because, according to Weiser, "any computer that fully engages or occupies you when you use it is a personal computer" [90]. Weiser pointed out that the differences between these waves are not in the types of devices themselves, but in the relationships between people and devices. In the era of ubiquitous computing, enormous

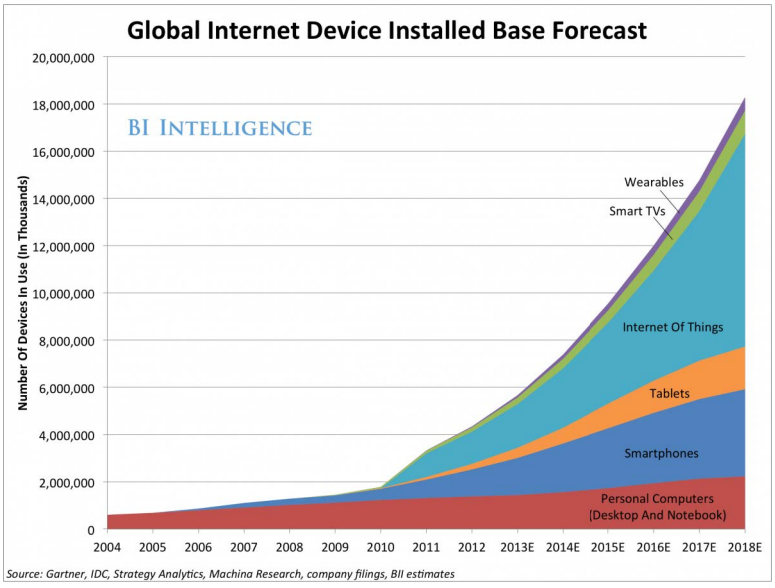
numbers of different types of computers will be included inside "everything," sharing each of us, and a fundamental change in the relationship between such computers and users will occur.

A world with numerous types of Internet of Things (IoT) devices, along with recent mobile and wearable devices, can be regarded as the real third wave, at least in terms of the device configuration, as shown in Figure 2.2.



Source: Ubiquitous Computing by Weiser [88]

Figure 2.1: "The Major Trends in Computing" in [88]



Source: Here Comes The Internet Of Things by BI Intelligence [10]

Figure 2.2: "Global Internet Device Installed Base Forecast" in [10]

Calm Technology Enabling "Disappearance"

However, the fundamental technology for ubiquitous computing is not powerful processors or Internet connectivity, although these are definitely necessary components. The key functionality in computer systems is what Weiser called "calm technology" in ubiquitous computing. When computers are all around us in our daily lives, "they better stay out of the way" [90], allowing us to remain calm and serene.

Although it may seem that information technology is often regarded as being incompatible with calmness, Weiser pointed out that several existing ubiquitous technologies, such as a fine writing pen and a comfortable pair of shoes, have already been bringing us a sense of calmness. He mentioned the following as the key.

> *"We believe the difference is in how they engage our attention. Calm technology engages both the center and the periphery of our attention, and in fact moves back and forth between the two."*
>
> Weiser [90]

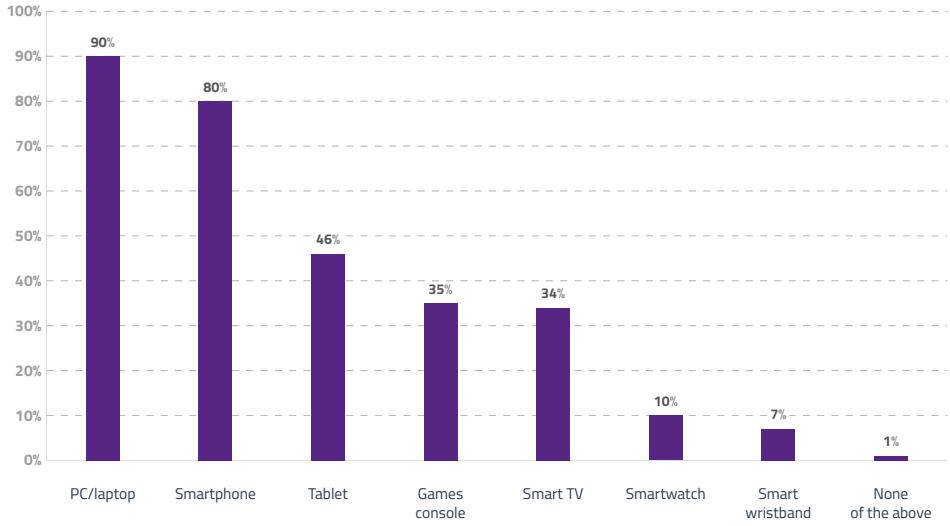
In the following sections of this chapter, I focus particularly on several concrete phenomena and trends happening in the recent age of ubiquitous computing. All of what I introduce here is important background of this research area.

2.2 Ubiquity of Computers around Users

The number of networked computing devices in a user's surrounding environment has been increasing. According to a 2014 survey conducted by GlobalWebIndex on Internet users aged 16 to 64, people tend to own an increasing number of computing devices. Figure 2.3 shows the rate of ownership of seven representative personal IT devices: a PC/laptop, smartphone, tablet, game console, smart TV, smart watch, and smart wristband. The average number of devices owned per person is 3.35 (worldwide), and Figure 2.4 shows these numbers for various countries. In addition, when looking at the average number of devices that people are using to access the Internet, the number is clearly increasing over time, as illustrated in Figure 2.5. In 2011, people went online using 2.12 devices on average. However, as of the end of 2014, this number has risen to 2.78.

Moreover, another recent report reveals that users tend to carry multiple devices and even use them simultaneously [31]. For example, 75% of the time when users are using a tablet, they are using another device (35%, a smartphone; and 44%, a television). In particular, the trend of watching TV with simultaneous Web access from a tablet or smartphone is called "second-displaying."

Among these various types of devices, the current center is the smartphone, a mobile phone with a modern multi-tasking operating system, Internet connectivity, a rich computing performance, and various types of sensors. Table 2.1 shows the technical specifications



Source: Multi-Device Owners by globalwebindex [87]

Figure 2.3: "Personal Device Ownership" in [87]

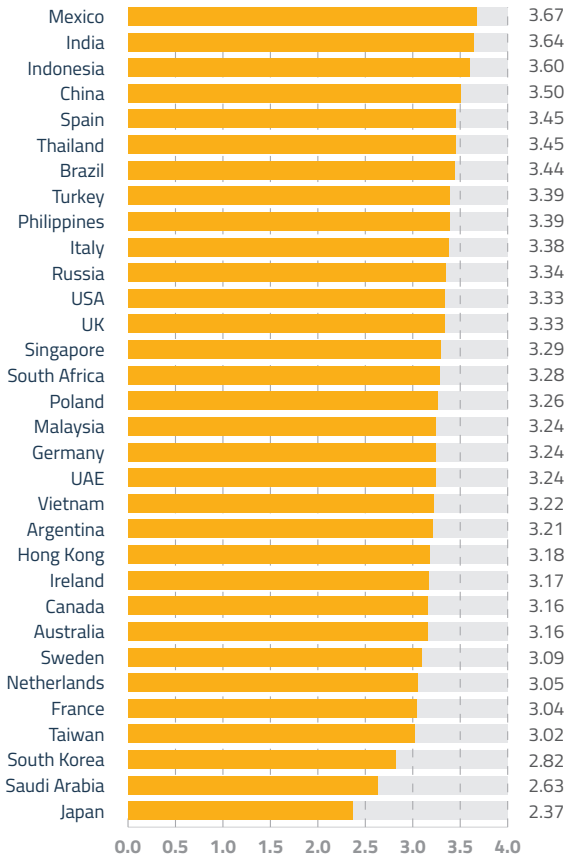
of an Apple iPhone 6 Plus, which was released in 2014. Its A8 dual-core CPU with a 1.4-GHz speed is considered to have a performance of 115.2 Giga FLoating-point Operations Per Second (GFLOPS). This performance roughly equals the performance of super computers in the early 1990s, such as the Fujitsu Numerical Wind Tunnel [40, 85], which recorded 124.50 GFLOPS in 1993. Another notable feature of the iPhone is its rich configuration of sensors. An accelerometer, a gyroscope, a proximity sensor, and a compass, as well as a GPS, cell phone data network interface, Wi-Fi, and Bluetooth enable many opportunities to sense the environment surrounding the device (in other words, context information around the user carrying the device).

2.3 Applications and Services in Ubiquitous Computing

Interacting with one's own carrying (and using) devices, as well as other devices embedded in the surrounding environment, users can utilize an increasing number of applications and associated services on the cloud side.

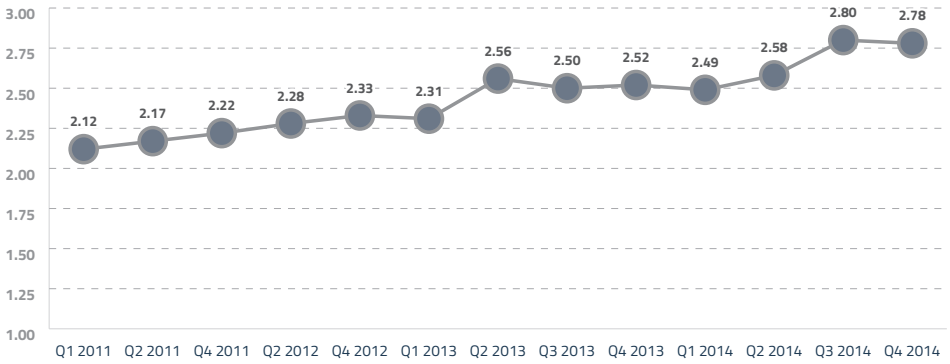
Because computer operating systems have advanced and acquired multi-tasking capabilities, multiple applications have started to be operated on a single computer simultaneously. In other words, users have started to be able to use multiple applications on one computer at the same time. Examples of such operating systems are various time-sharing systems, UNIX and Microsoft Windows.

On recent mobile devices, users are utilizing an increasing number of applications as the mobile application market, from which users can easily find and download new applications, grows drastically. Launched in 2008, Apple's AppStore is reported to have 1.7-million active applications as of June 2015. The competing Google Play store has 1.5-



Source: Multi-Device Owners by globalwebindex [87]

Figure 2.4: "Average Number of Devices Owned Per Person" in [87]



Source: Multi-Device Owners by globalwebindex [87]

Figure 2.5: "The Number of Devices Used to Access the Internet" in [87]

million applications. From these numerous applications on the market, Yahoo Aviate's research has shown that smartphone users on average install 95 applications on their phone

Table 2.1: Specification of iPhone 6 Plus

Type	Specification
Weight	129g
CPU	Apple A8 (ARM v8-based) (Dual-core 1.4GHz)
RAM	1GB
Display	1080 x 1920 dots
Storage	128GB
Network Interface	LTE (Maximum 100Mbps downlink) Wi-Fi 802.11b/g/n/ac (433Mbps) Bluetooth (3Mbps)
Sensors	Camera with 8 Mega Pixels Mic GPS Accelerometer Gyroscope Proximity sensor Compass Barometer

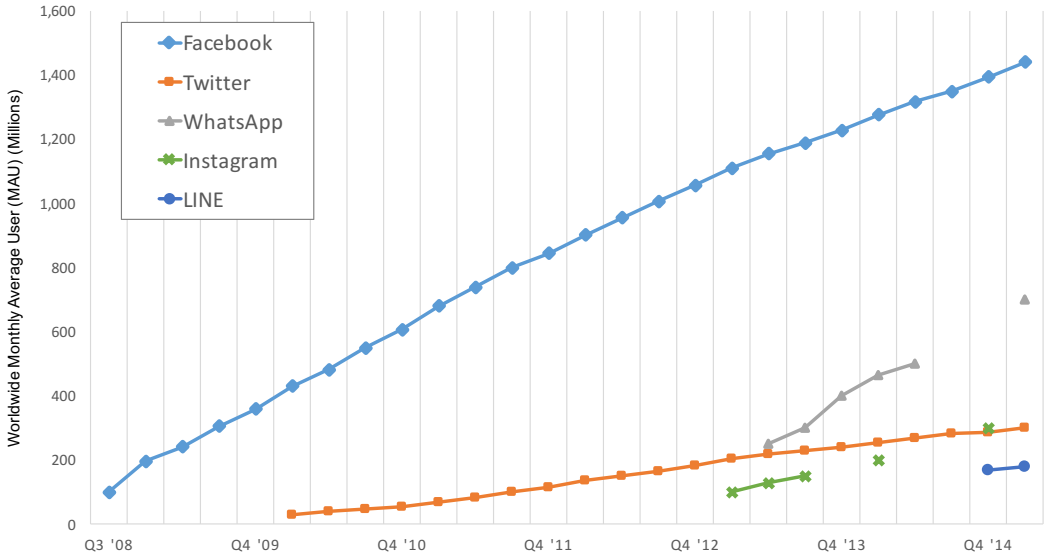
and use 35 of them throughout the day [92]. Other research [25] has shown that users continuously download new applications. Even in one of the most mature app markets in the U.S.A., consumers have been continually downloading applications at the same rate since 2011 (8.9 apps per month in 2011 versus 8.8 apps per month in 2014).

Finally, behind these numerous applications on user devices are also numerous cloud services. Driven by several technological advancements, such as various types of Web Application Framework (WAF) middleware that has enabled rapid service development, and an elastic cloud infrastructure that has enabled a rapid and scalable service deployment, we now have an uncountable number of Web services on the global Internet.

2.4 Connection and Communication among Users

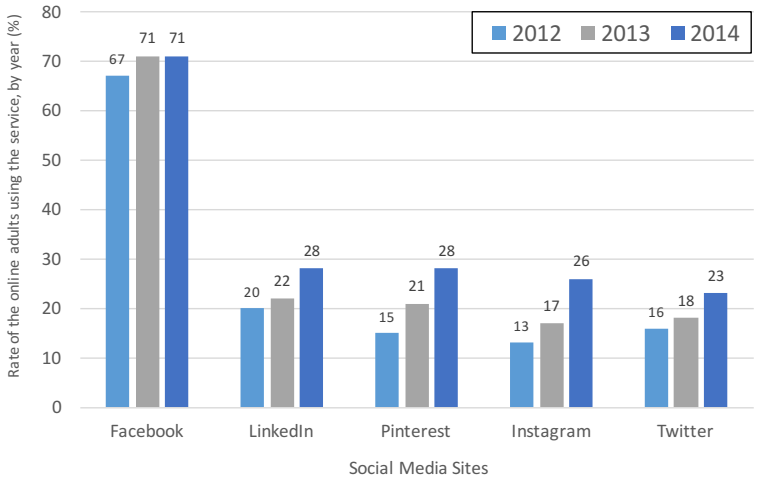
The advent of social networking services, in addition to conventional communication channels (such as email and SMS), has increased the number of people that users communicate with on a daily basis.

Figure 2.6 shows the number of monthly active users (MAU) of the major social media services worldwide from 2008 to 2014. The biggest social network, Facebook, has an MAU of 1.4 billion as of Q1 2015. This number is approximately 20% of the world's population. According to the results of a survey conducted on online users worldwide, all of the major social media sites have been receiving greater interest over the past recent years, as illustrated in Figure 2.7. In addition, the same survey showed that users have been using an increasing number of social media sites over time, as shown in Figure 2.8.



Source: Facebook, Inc., Twitter, Inc., LINE Corporation and Instagram, Inc.,

Figure 2.6: The Number of Worldwide Monthly Active Users (MAU) of the Major Social Media Services



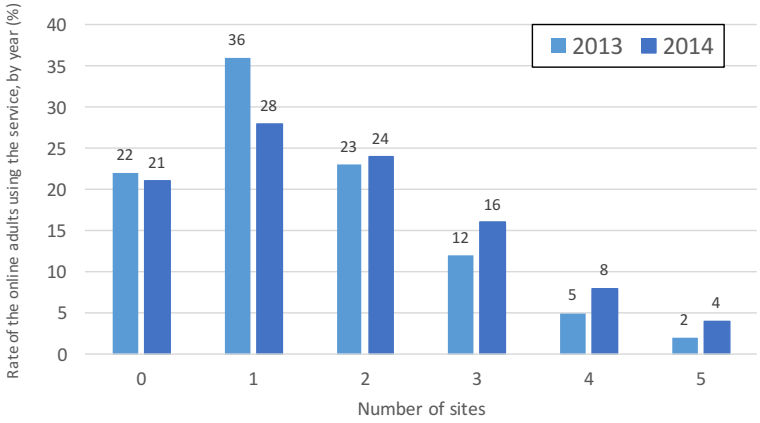
Source: Social Media Update 2014 [69]

September, 2014. $N = 1,597$, internet users ages 18+

Figure 2.7: Online Adults using the Major Social Media Sites, 2012-2014

Figure 2.9 shows that more than 50% of Facebook users (in the U.S. market, 18 years or older, $n = 1,074$, as of September 2014) have more than 100 "Facebook friends", with the median number being 155.

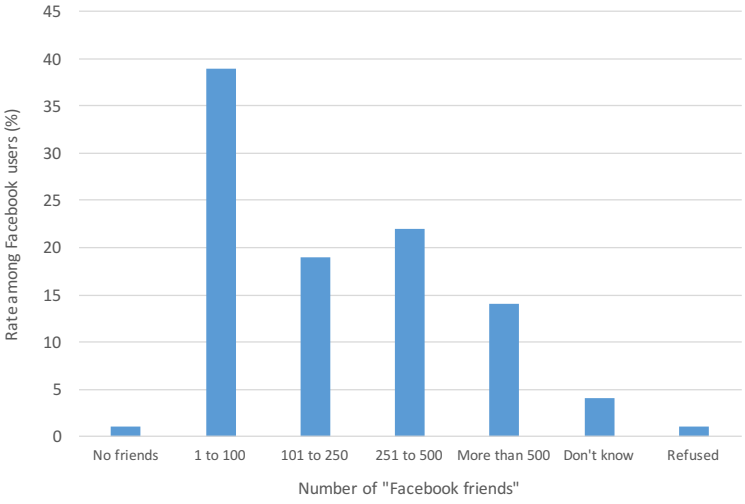
Furthermore, the frequency of access to each of the major social media sites by registered users is increasing, as shown in Figure 2.10. In particular, as the survey results show, some sites such as Facebook and Instagram have nearly 50% (Instagram) or far more



Source: Social Media Update 2014 [69]

September, 2014. $N = 1,597$, internet users ages 18+

Figure 2.8: People's Use of Multiple Social Media Sites, 2013-2014



Source: Social Media Update 2014 [69]

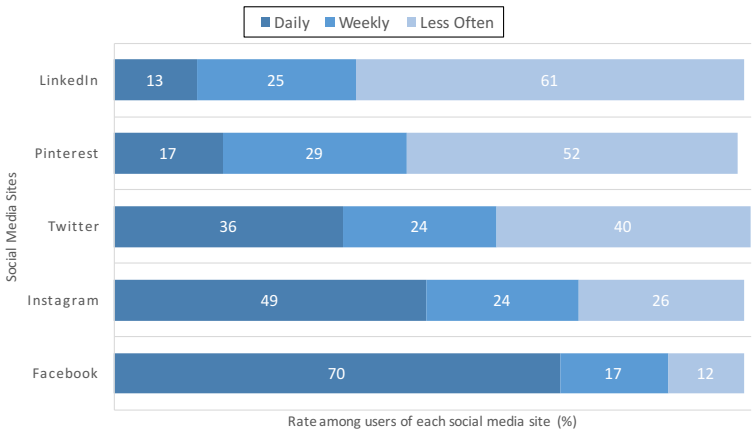
September, 2014. $N = 1,074$, Facebook users ages 18+

Figure 2.9: The Number of "Facebook Friends" for Each U.S. Facebook User

than 50% (Facebook) of users who access the site every day. A combined estimation from the numbers described here tells us that approximately 1 billion people around the world accesses Facebook daily.

2.5 User's All-Day Long Computing Experience

Finally, user's computing is becoming an all-day long experience. The duration of typical user's computing used to be 8 hours a day in the era of office computing. However, in the

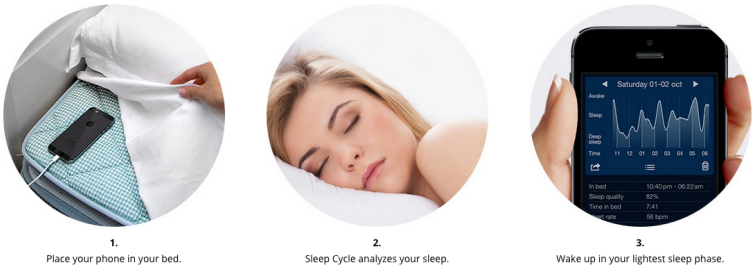


Source: Social Media Update 2014 [69]
September, 2014. $N(LinkedIn) = 463$, $N(Pinterest) = 398$, $N(Twitter) = 323$, $N(Instagram) = 317$, $N(Facebook) = 1,074$, Users ages 18+

Figure 2.10: Frequency of Major Social Media Site Use

age of ubiquitous computing, the duration is reaching close to 24 hours a day, meaning that a user's computing experience sometimes takes place even while in bed.

For example, with smartphones or smart wristbands such as Fitbit [24] and Jawbone UP [51], users can now track their own sleep activity. Figure 2.11 shows the use case of the Sleep Cycle iPhone application, with which users can wake up at a "comfortable" time based on their sleep activity as detected by the application and accelerometer on their iPhone.



Source: Sleep Cycle [64]

Figure 2.11: Sleep Cycle iPhone Application

Clearly, this trend is strongly related to and is accelerated by other trends presented in this chapter, such as emerging mobile and wearable devices with a small size, rich computational capability, and long battery life, and the versatile ubiquitous computing applications running on them.

2.6 Summary

In this section, I introduced the concept of ubiquitous computing, which is the background of the present research, along with several key phenomena:

1. In ubiquitous computing, users interact with an increasing number of diverse types of networking computer devices, either of their own or of other users in their surrounding environment.

2. In ubiquitous computing, users are utilizing an increasing number of applications on their devices and cloud services through a network.

3. In ubiquitous computing, users are communicating with an increasing number of other connected people through various types of communication services, and more in real-time.

4. In ubiquitous computing, users' computing experiences are becoming all-day long affairs, benefiting from more compact, powerful, and energy-efficient mobile devices.

Chapter 3

Notification in Computing

This section describes the use of notification in computing. The first section classifies interactions between users and computers into four basic models. The next section describes a notification system used in computing, with an introduction to its historical background. The final section specifies some distinctive trends in recent computing notification.

3.1 Interaction Models between Users and Computers

Interactions between a user and a computer can be categorized into four types in terms of the number and timing of the inputs from the user and outputs from the computer: Job Dispatching (JD), Real-Time Interaction (RTI), Continuous Updates (CI), and Full Proactivity (FP), which are shown in Figure 3.1.

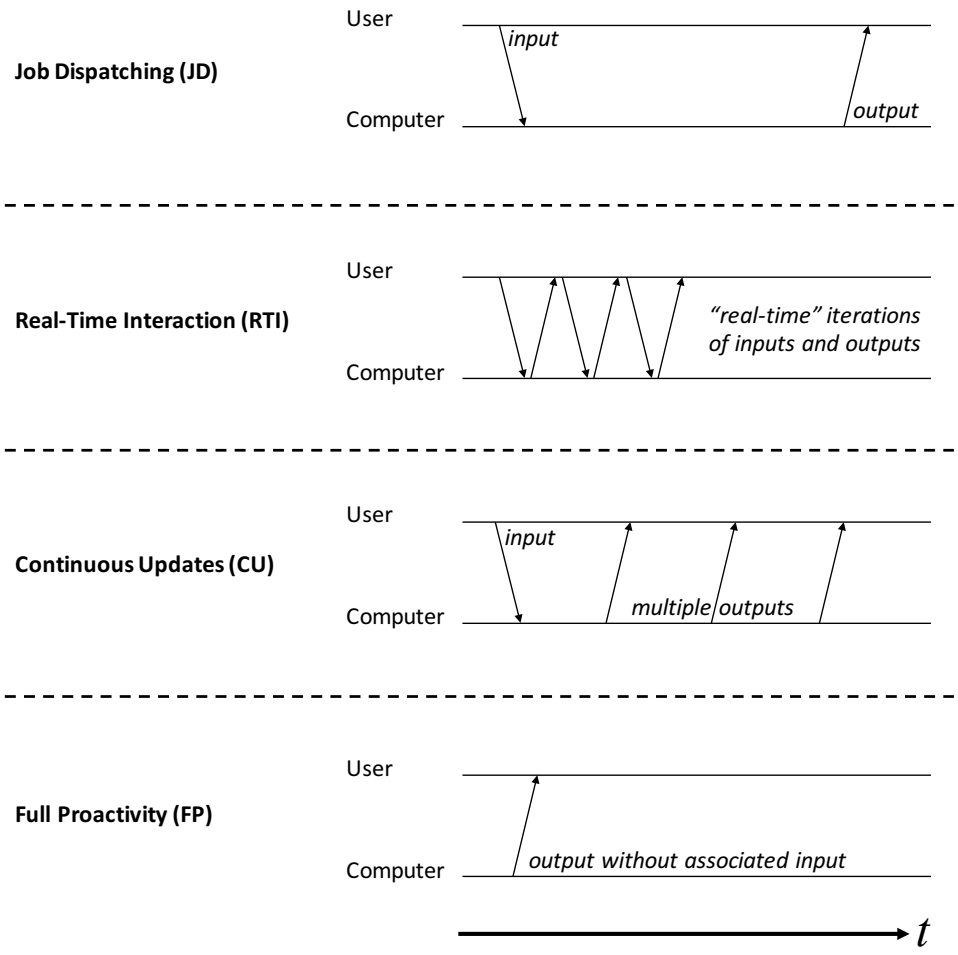


Figure 3.1: Models of Interaction between User and Computer

- Job Dispatching (JD)

  In JD, a user operates a computer by dispatching a command (or a "job" as a sequence of commands) and receives a result from the computer after a certain period of time. Because the time lag from the command dispatch to receiving the result is relatively long (usually on the order of several minutes to even hours), interactions between users and computers are not defined as "interactive" or in "real-time." Historically,

batch systems in the 1960s used this type of operation. Even now, CPU-intensive computing, such as simulations, video rendering, and large-scale data manipulation are examples of this type.

- Real-Time Interaction (RTI)

  With this model, a user interacts with a computer through multiple iterations of input from the user and output from the computer in real-time. Examples of this include, particularly as an application in the foreground of the computer, a command-line interface based operation with a shell program and a GUI-based interactive manipulation of an application in a modern widow-based operating system.

  Users issue an input to an application by issuing a new character-based command on a shell or by clicking a button on the GUI interface. The application processes the issued input accordingly and returns the computational result to the user **promptly** by printing a character-based result of the command or displaying a GUI-based result, usually with a response time on the order of seconds.

  Having such a short waiting time, the user basically waits for a response from the computer and progresses with their task synchronously and in real-time, without conducting any other tasks during the waiting period. The user's computing task, such as word processing or step-by-step data processing, will progress through iterations of command issues by the user and the provisioning of results from the computer. Such iterations will be repeated during the user's computing session.

  This type of interactive computing was developed in Time Sharing Systems in the 1960s to shorten the waiting time for users. CTSS [16] is a representative first-generation example of such a system. Using either a Character User Interface (CUI) or a Graphical User Interface (GUI), this interaction model was introduced to Personal Computers (PC) in the 1980s, as well as to mobile PCs and even mobile devices and smartphones.

- Continuous Updates (CU)

  The Continuous Update (CU) interaction model is composed of a single input from a user and one or multiple outputs from a computer repeatedly. A simple example of this interaction is a continuous query in a database system.

  This model was developed to obtain the computational results even more speedily than using the RTI interaction model. With an RTI, a user needs to issue a new input command to obtain a new computing result (polling). In contrast, with a CU, the user can issue a single input and wait for multiple continuous outputs from the computer.

  Concrete examples of this model are often seen in Web-based services, such as various kinds of alert or recommendation services. Google Alert [29] is an alert service based on a Web search result with a specified keyword. A user configures a specific search keyword as the key of an alert. Whenever Google finds a new Web page that

includes the specified keyword, the service notifies the user. Other examples are various types of recommendation services that find and notify "shops to recommend" repeatedly according to the user's current context (such as location), based on the user's singly pre-configured preferences.

- Full Proactivity (FP)

  This interaction models involves zero input by the user but one or more responses from the computer.

  The first example of this model is a generic notification from the operating system. When the computer is almost out of local resources such as free space in the main memory or hard drive, a warning message or pop-up window notifies the user of the situation.

  Another example of this model is communication application and services, such as email, chat, or phone calls on the network. When a user has an incoming email, message, or phone call, the system notifies the user and provides the associated information, such as the content of the email or message.

3.2 Notification System

This section describes the concept and the system of notification as a distinctive means of providing information from a computer to a user in the computer system, along with the background motivation for the invention of a notification system.

Motivation for Notification

Figure 3.2 shows the evolution of computing with different sets of interaction models (introduced in the previous section) over time.

In the early age of computers, including a batch system, interactions between the user and a computer were mainly based on JD, as shown in "Phase 1" of the figure.

In the age of the Time Sharing System, computers became more interactive in real-time, as illustrated in "Phase 2." The interactions between a user and a computer were based on the RTI model.

In the age of multi-task operating systems ("Phase 3"), such as UNIX, where each user can execute multiple applications concurrently, interactions between a user and a computer changed to a mixture of JD and RTI. At a certain moment, the user basically interacts with the "foreground" application using an RTI, leaving other applications in the background. However, because task switching between foreground and background applications can be done easily and virtually in real-time, the user pursues one or more tasks simultaneously by instantly switching between multiple applications on the system. This is the moment when the motivation for using a "notification" arose. Without any notification, a background

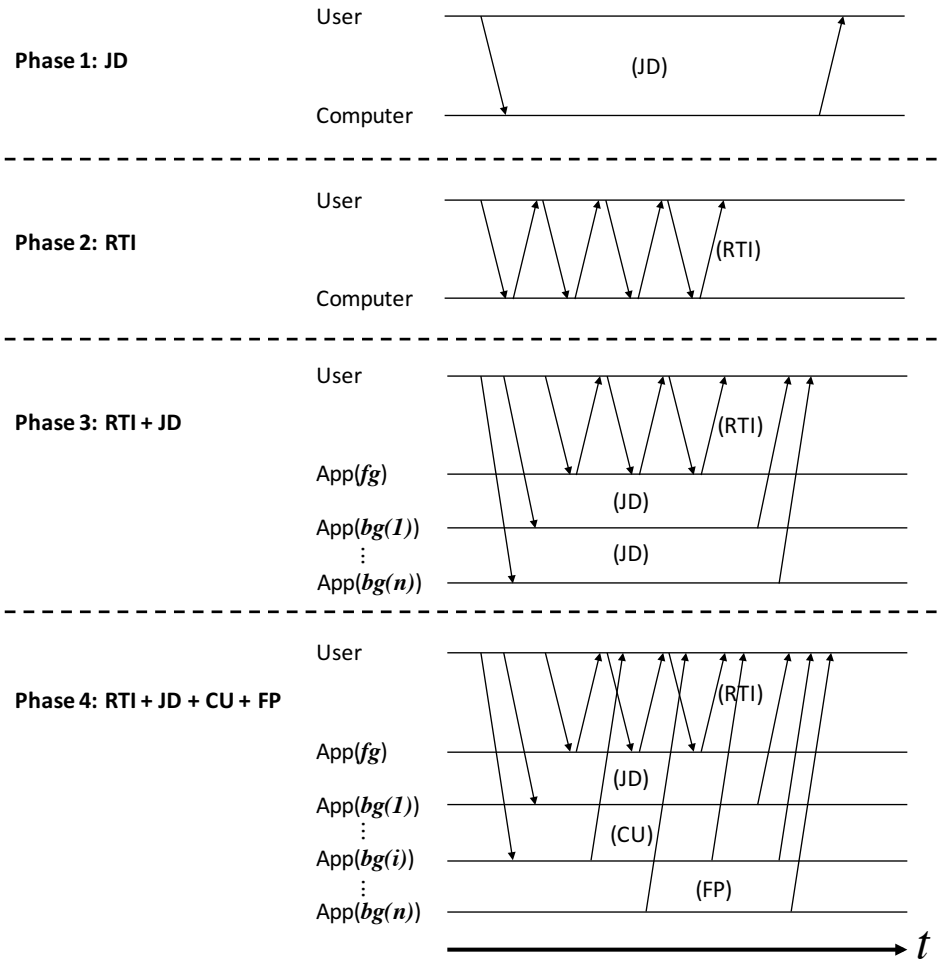


Figure 3.2: Transition of Computing with Different Set of Interactions

application with a new computing result to be output to the user needs to wait to be switched to the user's foreground before presenting its output. In other words, the user needs to switch the background application to the foreground to check new outputs from the application. With a notification capability, such background applications can inform any new events to the user even before being switched to the user's foreground.

More recently, as shown in "Phase 4" in the figure, applications running on a computer tend to have more CU and FP interactions. There are multiple reasons for such a trend. First, applications are becoming context-aware, having richer capability of adaptation against various types of user contexts. Various types of events from the operating system, changes in sensor data read from a local mobile device, or any application layer event sent from a remote over the network are examples of context changes. The second driver of this trend is the emergence of numerous types of Web services that are continuously being run on cloud servers. Many applications on the user's local device are backed up by such a cloud

service. From a different point of view, many applications are now "stalls" of the Web service. Under such a configuration, a variety of application-specific (i.e., service-specific) computations executed continuously at the server are continuously output, which eventually leads to a CU output of the application.

Notification System

Since the age of "Phase 3," where multiple applications could output their computations in random timing to the user, a notification system was developed as a side channel software and/or hardware-based component of the operating system, providing information more speedily and proactively to the user.

The key feature of a notification system is its capability to push information to users in a more speedy and timely manner. Notifying newly available information (or, at least, the fact that such information is available) enables the immediate recognition of such information by the user. The user can recognize the information (or the fact of its availability) immediately, rather than by polling the computer to check whether there any such new information is available, which takes an unnecessarily longer period of time before receiving the information.

Another important characteristic of a notification system is that, especially in multitasking operating systems, the notification system can provide new information to users from the background of their current task. While a user is using a specific application in the foreground, other applications and operating systems running in the background can interrupt the user with a notification. Referring to the fundamental motivation of the notification systems mentioned above, the delivery of notifications from the user's background of activity matches the original design principle of speediness. However, depending on the current activity status of the user with the foreground application, this delivery scheme may not be ideal in terms of timeliness from the user's viewpoint, possibly hindering the user's current activity through an interruption.

A notification system in a narrower sense usually means software components in the operating system. As an early operating system, UNIX has `wall` (abbreviation of "Write All") command that provided specified messages to all users currently logged onto the computer. Although `wall` is basically a general-purpose messaging program between users of the same computer, this command has often been used to notify all users of a rebooting operation. Most modern operating systems, including Microsoft Windows, MacOS, iOS, and Android OS, have their own built-in notification systems. Figure 3.3 shows examples of the notifications used in Microsoft Windows and Apple iOS. In addition, several operating systems have a centralized view of currently posted notifications, such as the Notification Center in iOS, shown in Figure 3.4.

Almost all current computer users of a major operating system are faced with a certain amount of notifications. Notifications are used for a number of purposes and situations in modern computers, such as notifications of a change in status of the computer, incoming calls, messages from other users, and even emergency disaster alerts [37].

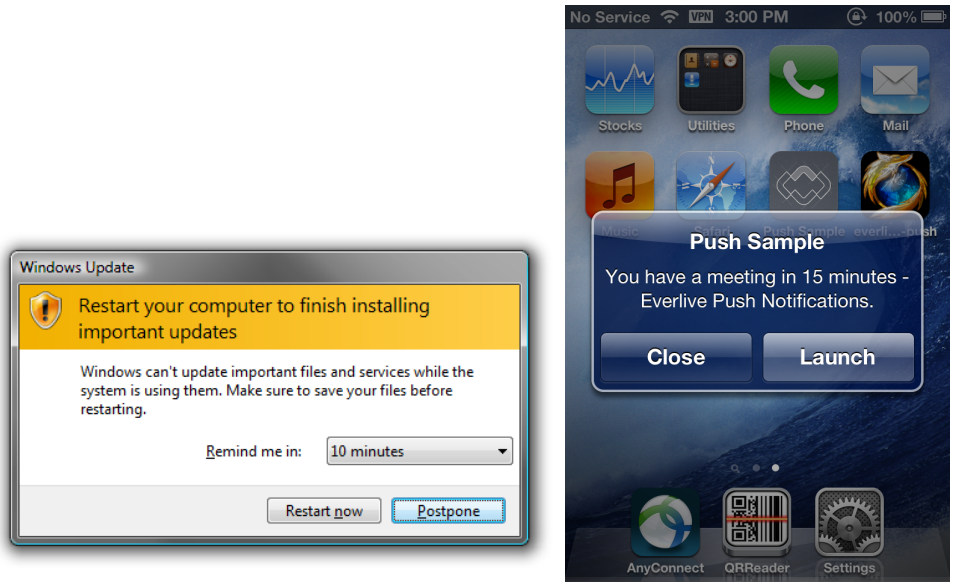


Figure 3.3: Examples of Notifications in the Modern Operating Systems (Left: Microsoft Windows, Right: iOS)

Figure 3.4: Notification Center in iOS

3.3 Recent Trends in Notifications

Following the transition of computing with different sets of interactions over time, as shown in Figure 3.2, herein I specify three recent and distinctive trends in notifications. Figure 3.5 illustrates such trends. Each trend is related to the background phenomena in the ubiquitous computing presented in Chapter 2.

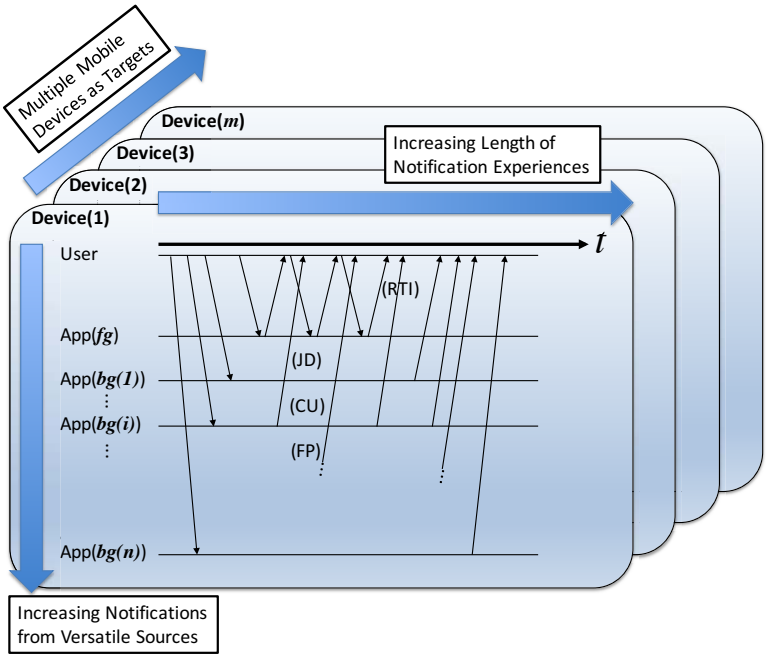


Figure 3.5: Recent Trends in Notification

- **Increasing Notifications from Versatile Sources**
  The first trend is an increasing number of notifications from an increasing number of applications installed and running on a computer. Furthermore, behind such applications are versatile services on the net, and an increasing number of other users connected to the network. Due to the CU and FP interaction capabilities of such applications, more notifications are being delivered to users.

- **Multiple Mobile Devices as Targets**
  As mentioned in Section 2.2, users have been carrying and using an increasing number of devices. Users receive notifications on each device individually. Furthermore, a user may often install the same application, which can be viewed as a front-end of a Web service, into their multiple devices. This may lead to a situation with multiple duplicated notifications with the same content delivered to multiple devices.

- **Increasing Length of Notification Experiences**
  As described in Section 2.5, user's computing has been changing to an all-day long

experience, with users surrounded by multiple mobile and wearable devices with a long battery life and various types of ubiquitous computing applications that support the users' lives comprehensively. Under this situation, the notification experience is also becoming an all-day long affair.

3.4 Summary

This section introduced the use of computing notifications. First, I classified the interactions between users and computers into four different models, namely, Job Dispatching (JD), Real-Time Interaction (RTI), Continuous Updates (CU), and Full Proactivity (FP).

As the interactions between users and computers progresses and becomes multiplexed with the emergence of multi-tasking operating systems, applications in the background have started to provide users with speedy information through the use of notifications.

Following the transition of computing with different sets of interactions over time, three distinctive notification trends were specified: (1) an increasing number of notifications from more versatile sources, (2) multiple mobile devices as targets of notifications, and (3) an increasing length of notification experience of users.

Chapter 4

Attention-Awareness in Computing

This section addresses attention-awareness in computing. First, I clarify the concept of "attention" along with its nature of capacity limitation, and the situation of "divided attention" by referring to past articles on cognitive psychology. Next, I define the problem of "interruption overload," a negative impact on a human user's attention by too many ill-timed interruptions from notifications. The final section defines attention-awareness in computing and the necessary functionalities including attention sensing, which is the scope of this research. Finally, the requirements of the attention sensing solution will be specified.

4.1 Attention in Cognitive Psychology

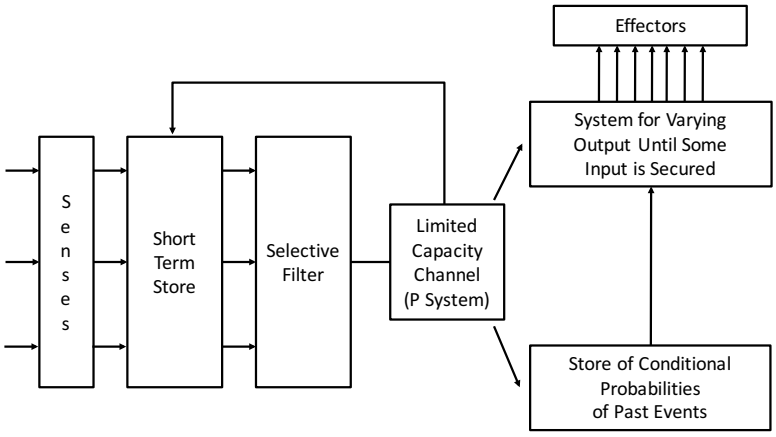
Attention is a concept studied in the field of cognitive psychology and refers to how humans actively process specific information available in their environment. William James, an American philosopher and psychologist, defines attention as follows.

> *"Everyone knows what attention is. It is the taking possession by the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration, of consciousness are of its essence. It implies withdrawal from some things in order to deal effectively with others, and is a condition which has a real opposite in the confused, dazed, scatter-brained state which in French is called distraction, and Zerstreutheit in German."*
>
> James [50]

Limited Capacity of Attention

As one of the representative characteristics of attention, it is commonly understood that attention has a limited amount of capacity.

The concept that humans can process only a limited amount of information at any given time goes back to "filter models" of attention, such as those by Broadbent [12], Treisman [86], and Deutsch [20]. These models treated attention as a **structural mechanism** that works as a "bottleneck" and prevents an excessive amount of information (more than the limit) from being processed at any given time. Figure 4.1 illustrates the processing structure proposed in the "Filter Theory" by Broadbent [12].

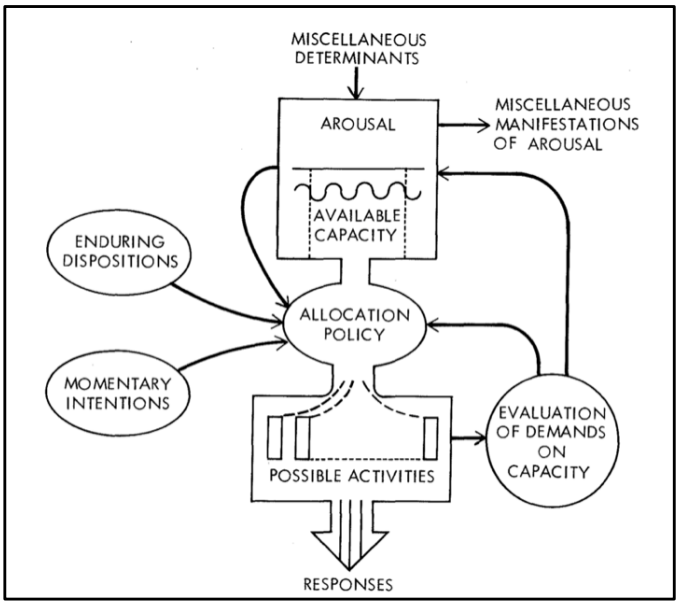


Source: [12]

Figure 4.1: Structure of Broadbent's Filter Theory in [12]

Later, Morary and Kahneman explored a new idea that information processing is regulated by a more general limit of capacity. At the center of Kahneman's capacity model,

there is the idea of **a limited capacity of mental effort** (used as a synonym of attention) that humans can devote to mental work. Figure 4.2 shows the capacity model of attention proposed by Kahneman [53]. This idea of limited capacity of attention was widely spread and has been largely influencing other studies. Also in this work, attention was treated as a "resource" that will be allocated to single or multiple target tasks. Since this release of this research, attention has been widely understood as a resource, although there have been several different models of attention resource and allocation, such as the central resource theory [53] and multiple resource theory [75].



Source: [53]

Figure 4.2: A Capacity Model of Attention by Kahneman [53]

Divided Attention

In theories regarding the concept of a limited capacity of attention resources, one basic idea is that a limited amount of attention will be allocated to a wide range of current tasks of the user. Thus, the level of performance of each of these tasks is dependent on the amount of attention demanded. A decrease in task performance will be observed when the total attention demanded of the task is more than the available amount of attention resource. This prudent allocation of available attentional resources to coordinate the performance of more than one task at a time has been defined as "divided attention" [81].

4.2 Interruption Overload Problem

As introduced in the previous section, human attention is a resource with a limited capacity. Meanwhile, in the area of ubiquitous computing, the amount of information available for each user is increasing, as described in Chapter 2.

Information Overload

Given such a background, a gap between the (unchanged) amount of available attention resources of a user and the amount of resources demanded by the increasing amount of information will be significant. In other words, attention will be recognized as a significantly precious resource. Herbert A. Simon (political scientist, economist, sociologist, psychologist, and computer scientist) was one of the first people to emphasize the preciousness of the limited amount of attention expected in ubiquitous computing (using the words "information-rich world").

> *"in an information-rich world, the wealth of information means a dearth of something else: a scarcity of whatever it is that information consumes. What information consumes is rather obvious: it consumes the attention of its recipients. Hence a wealth of information creates a poverty of attention and a need to allocate that attention efficiently among the overabundance of information sources that might consume it."*
>
> Simon [76]

Toffler used the term "information overload" in his book "Future Shock" [84], meaning a situation in which a person has a difficulty in understanding a problem and decision making owing to the existence of too much information. When the amount of input into a system surpasses the processing capacity, an information overload occurs [58].

Several past studies in various fields have revealed that an information overload causes a negative impact in terms of increasing the time required to make a decision, increasing confusion regarding a decision [15,48,49,57], and the quality of the decision [1,14,74,78].

Interruption Overload

An interruption overload is a sub-component of an information overload. An interruption overload is a situation in which too many ill-timed interruptions cause a negative impact on the user's attention resource and task performance.

As mentioned in Chapter 3, users have been faced with an increasing number of notifications from versatile sources to multiple mobile and wearable devices in the recent era of ubiquitous computing. Events causing notifications occur individually with random timing. Meanwhile, a typical notification system delivers notifications immediately to the user once they are available based on the original concept of speedy information provisioning. As a

result, users end up facing numerous notifications with random timing, regardless of their timing preference. When a notification is perceived and recognized by a user, some amount of the user's attention will be allocated to the information carried by the notification. At this moment, an interruption occurs.

This type of interruptive notification, despite its obvious benefits, has been shown to negatively affect a user's work. Several researchers have found that it leads to a reduction in work productivity, including the resumption time from the interruption back to the primary task, along with the quality and amount of time available for decision making [2, 4, 18, 55, 80, 93]. Other researchers have found increasing negative affects or emotional states, social attribution [2], and psycho-physiological states [93] as a result of these interruptive notifications.

In addition, it is known that users tend to keep using interruptive notifications rather than simply disabling them. Although notifications can be configured by users, and can even be disabled, simply disabling user notifications negates their benefit and cannot satisfy the users' need for a timely provisioning of information. Previous research has shown that users prefer to keep using notification systems for information delivery, even given the interruption costs, rather than turning them off and checking for new information manually [46].

Given the backgrounds and trends presented in Chapter 2 and Chapter 3, the interruption overload problem is becoming of greater importance in the area of ubiquitous computing.

Considering the concept of ubiquitous computing, ubiquitously existing computers that cause interruption overloads are not a calm technology, instead providing interruptive notifications at the center of the user's attention with random timing, hence having several negative impacts on the user's performance. Toward the realization of ubiquitous computing, the interruption overload problem is a significant one to be addressed and solved.

4.3 Attention-Awareness in Computing

To resolve the interruption overload problem, what is fundamentally needed is "attention-awareness" in computing. Choosing words as generally as possible by partially following the definition of "context-awareness" by Dey [21], attention-awareness is defined as follows:

> Attention-awareness: *A system is attention-aware if it uses the status of user's attention resource to provide information and/or services to the user in a way that contributes to preserving the user's precious attention resource.*

Applications

With the capability of attention-awareness, a series of new applications may be realized, including the followings:

- **Attention-aware information provisioning under versatile situations**: Information provisioning that is aware of the user's attention status is the most immediate application on top of attention-sensing. In participatory sensing, asking for a sensing task to a user with appropriate timing is expected to have a higher responsiveness to the query by the user. In a car-driving situation in which the driver basically needs to pay attention into the physical world around the car rather than to the cyber world, showing any non-emergency information in "attention-aware timings" (such as after the driver stops at the traffic signal) is expected to result in a lower cognitive load of the driver.

- **Attentive Call**: Attentive Call is used for a network chat and voice talk service using an application on a smartphone and the back-end server in the cloud. With conventional phones and several voice/video-talk services, a user calls another user, often with some concern that they may end up interrupting the other user by calling in the middle of their important business. With Attentive Call, the user can initiate a call "at a future time when a remote user has a low attention load."

- **Proactive Attention Management**: Possibly combining the system with the user's external calendar data and/or other systems to sense "what the user needs to do now," a proactive management of the user's attention will be possible. For example, suppose a user needs to concentrate on a document-editing task. If the system detects the user is in a state of "divided attention" by detecting that the user's attention is moving around his target devices (e.g., frequent and periodic checks of social media updates on a carried smartphone), proactive attention management logic can possibly (1) lock the smartphone until the user's primary work is completed or (2) continuously display the content of the user's primary task window even on the smartphone screen to keep the user's attention on their primary task.

Functionalities of Attention-Awareness

Toward the realization of attention-awareness, the following specific functionalities are needed.

- Attention sensing
- Attention-aware adaptation
- Attention prediction
- Attention management

Among the functionalities, attention sensing is the first and most challenging research problem because (1) attention sensing literally needs the sensing of a human's internal attention state, and (2) all other functionalities depend on information on the sensed attention status.

Requirements for Solution

Referring to the research backgrounds, the solution for attention sensing has the following design requirements.

- R1: **Compatibility with a user's multiple mobile and wearable devices:** Users carry and use multiple mobile and wearable devices, such as smartphones, tablets, or smart watches for their everyday computing and communication. Thus, a solution needs to be compatible with such computing situations, such as its feasibility on mobile and wearable multi-device platforms.
- R2: **Applicability to diverse types of notification applications:** Users are experiencing notifications from diverse types of notification source applications. Thus, the solution needs to have compatibility and applicability against such versatile notification sources.
- R3: **All-day long use:** The solution needs to be suitable for a user's day-long computing experiences.
- R4: **Real-time sensing:** To realize other functionalities of attention-awareness, such as attention-aware adaptation on the fly, the solution needs to be performed in real-time.

4.4 Summary

This section clarified the concept of attention, how humans actively process specific information available in their environment. Although multiple models have been proposed, it is generally said that human attention is a resource with limited capacity. In addition, it is well known that, when an attention resource is divided into multiple user tasks with a greater amount of attention demanded than currently available, the user's task performance will decrease. Given this existing knowledge on attention, along with the research background specified in Chapters 2 and 3, interruption overload was defined as a situation in which too many ill-timed interruptions causes a negative impact on the user's attention resource and task performance. What is fundamentally needed in the computing area is attention-awareness. In particular, attention-sensing functionality is a fundamental research issue because all other functionalities, such as attention-aware adaptation or attention prediction, depend on the sensing. Finally, I specified four requirements for the solution of attention sensing: (1) compatibility with the user's multiple mobile and wearable devices, (2) applicability to diverse types of notification applications, (3) all-day long use, and (4) real-time sensing.

Chapter 5

Related Work

This section introduces related studies with several different approaches for sensing the user's attention status in the context of interruption overload. After introducing them, I clarify how each of the works succeeds and fails in satisfying the requirements for the attention sensing solution specified in the previous chapter.

5.1 Categorization of Approaches

Before introducing each item of the related studies, I will clarify some of the categorizations of the approaches taken in the research in terms of the target of sensing and methodology of adaptation.

Approaches in the Target of Sensing

When considering the sensing status of a human user's attention, there are several multiple possibilities for a concrete target of the sensing.

- Sensing the user's cognitive load
  The concept of **cognitive load** is used as the total amount of mental effort allocated to working memory in the field of cognitive psychology. Several different approaches for measuring such load have been proposed, such as (a) subjective rating-based methods, (b) task performance-based methods, and (c) physiological response-based methods.

- Sensing the user's interruptibility
  A user's self-reporting value of **interruptibility**, such as an answer to the 5-point Likert scale question "Is this an interruptible time for you now?" is considered the ground truth value in this method. Several studies took an approach of estimating the user's current interruptibility based on various types of user context information, such as the user's calendar schedule, sensor data from the user's device, and the user's location. As a means of collecting the interruptibility ground truth, Experience Sampling Methodology (ESM) [17] is often used.

- Sensing the user's breakpoint
  For the time in which the user's cognitive load is low, a number of researchers have used the concept of **breakpoint** [63]. Breakpoint is a concept in psychology in which a human's perceptual system segments activities into a hierarchical structure of discrete sub-actions. The boundary between two adjacent action units is called a breakpoint. According to several past studies [2, 42, 43], deferring notifications until a detected breakpoint has been shown to reduce the interruption cost in terms of task resumption lag and subjective frustration.

Approaches to Adaptation

Several related studies have also involved how to adapt interruptive notifications so as to prevent the user's high cognitive load and/or mental burden, in addition to sensing the user's attentional status. As opposed to disabling notifications completely, there are two common approaches for addressing an interruptive overload described in the literature: (a) deferring

notifications until a more appropriate time, and (b) mitigating the interruptive nature of the notifications.

- Adaptation in Timing
  When deferring the notifications, an appropriate deferral time must be identified. A number of researchers have identified breakpoints in user activities as this deferral time. As mentioned, deferring notifications until a detected breakpoint occurs has been shown to reduce the interruption cost in terms of task resumption lag and subjective frustration [2, 42, 43].

- Adaptation in Modality
  The other approach, mitigation, tries to reduce the impact of a notification on the user's cognitive load by changing the modality used to deliver notifications. This can include the use of "silent" mode, "vibration" mode, or simply flashing an LED (e.g., [61]). This approach serves to change the saliency of the interruption, while leaving the timing of the notifications unchanged.

While these two approaches are complementary, in this dissertation, my research focuses on notification deferral. Given the growing number of notifications that users must deal with, changing the timing of the notifications rather than their saliency would seem to have a greater potential impact on the interruption overload of users. With this focus, I turn to the concept of identifying interruptive moments or breakpoints.

5.2 Measuring Cognitive Load

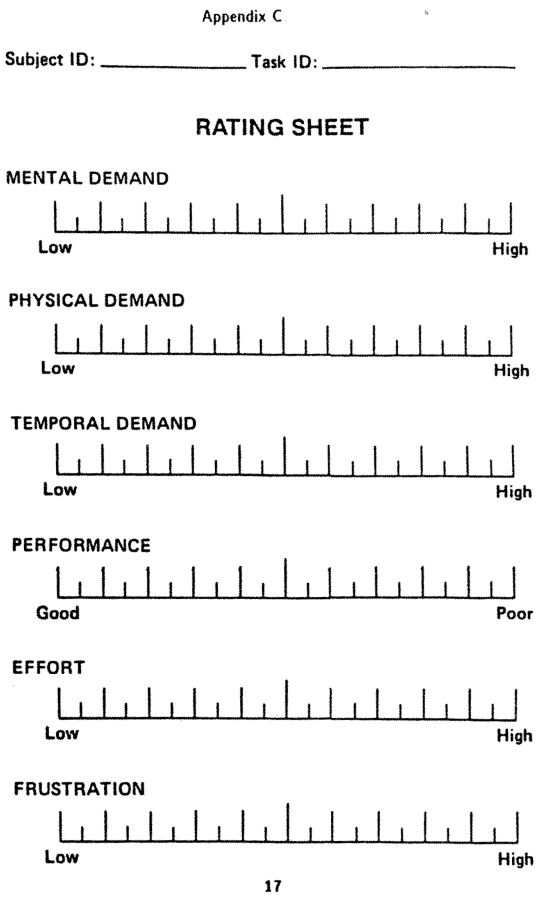
As mentioned previously, multiple approaches have been proposed for measuring a user's cognitive load.

Subjective Rating-based Approach

The first is a subjective rating-based approach. Several past studies have identified that the measurement of cognitive load through post-hoc self-reporting is a relatively reliable methodology for mental effort assessment [67].

The most widely used tool for assessing a user's cognitive load is the NASA Task Load Index (NASA-TLX) [33]. In this method, each user answers a survey consisting of two parts. In the first part, the user gives their subjective ratings on six different scales: (1) mental demand, (2) physical demand, (3) temporal demand, (4) performance, (5) effort, and (6) frustration using a 100-point scale. Figure 5.1 shows the rating sheet for the first part. In the second part, the user proceeds to a series of pair-wise comparisons among all of these six scales (i.e., ${}_6C_2 = 15$ comparisons) based on their perceived importance. The resulting "weight" of each scale is the number of times the scale was chosen out of 15 pair-wise comparisons. Finally, the Weighted Workload (WWL) score, which indicates the

user's workload, will be calculated as shown in Formula 5.1. The calculation is conducted based the average of all "weighted scores" (the score of each scale multiplied by its "weight" and divided by 15).



Source: NASA Task Load Index (TLX) v. 1.0 Manual [62]

Figure 5.1: Rating Sheet of NASA-TLX in [62]

$$WWL = \sum_{scale} \frac{Score_{scale} Weight_{scale}}{15} \tag{5.1}$$

Although this methodology is widespread, the post-hoc nature of this approach makes it difficult to be applied to versatile ubiquitous computing systems where an assessment needs to be completed in real-time. Referring to my requirements for the solution in Section 4.3, this type of approach is not compatible with the **real-time sensing** requirement.

Meanwhile, in this research, user studies for evaluating my proposed system used NASA-TLX on a nightly basis as a means of measuring the user's daily "workload perception" (or "perceived workload") rather than the user's (relatively in real-time) "cognitive load." The

reason of this is the time gap between users' actual notification experiences and the survey. During the experiment (16 days and 31 months, respectively, for my two different user studies), each user was given a form of NASA-TLX each night. Each user was asked to look back on the day and review their own "notification experience" by answering the given form. Because the actual notification by my system was designed to be experienced by the users from 8 a.m. to 9 p.m. daily, and because the nightly survey was sent using email at 9 p.m. every night, the survey actually had a temporal interval from the actual notification experience until the survey time.

Task Performance-based Approach

The measurement of a user's task performance is used to objectively assess the user's cognitive load during the task execution. The user's performance regarding their primary and focal task is used in the "primary task measurements," whereas "secondary task measurements" exploit the performance of a secondary task that was (often asked to be) executed simultaneously with the primary task [67]. In this methodology, the variation in reaction performance indicates the variation in cognitive load.

However, this methodology may not be feasible in situations of ubiquitous computing, since a user conducts multitasking with frequent task switching between multiple tasks, and since it is often difficult to measure response performances of the user's versatile types of tasks using uniform measurement criteria.

Physiological Response-based Approach

The third approach is to measure the user's physiological response using psycho-physiological sensors. Such sensing includes several different techniques, such as tracking of the eye movement and pupil size [8, 39, 41, 91], and readings from an electrocardiograms (ECG), galvanic skin response (GSR) [39, 70, 73], electroencephalogram (EEG) [70, 91], heart rate (HR), and its variability (HRV) [26, 60, 91].

Haapalainen *et al.* [32] found that, in desktop computing, a combinational use of an electrocardiogram and the heat flux is the most accurate at classifying low and high levels of cognitive load. Although this approach looks promising in detecting a user's cognitive load in real-time, the burden placed on users who have to wear such sensors is not trivial. In ubiquitous computing where users are mobile, such a burden is expected to be even more bothersome for users. Thus, a solution using this approach may not be compatible with the **all-day long use** requirement presented in Section 4.3. In addition, these physiological response data will be noisier in such mobile situations, and the sensing of cognitive load is expected to be more difficult.

5.3 Work in Desktop Computing Domain

In the latter two approaches, in terms of target of sensing, namely, sensing the user's interruptibility and breakpoint, several past works have been conducted in the area of desktop computing, and later, in the mobile computing domains.

Early work on such detection has naturally focused on desktop environments.

Horvitz *et al.* inferred interruptibility accurately in desktop computing environments, by using context information, such as interaction with computing devices, visual and acoustical analyses, and online calendars [35].

Hudson *et al.* constructed statistical models for predicting the interruptibility of office workers by using long-term audio/video recordings with manually-emulated sensors of the user's activity status, along with the experience sampling technique [38]. For these two systems, recognition was performed in a posteriori manner, and is thus not compatible the **real-time sensing** requirement.

Later works by Begole *et al.* [9] and by Horvitz et al. [36] focused on systems that supported real-time detection of interruptibility; however, these systems required the use of dedicated custom hardware.

In contrast, OASIS also identified breakpoints in real-time, but did not require custom hardware, instead using information regarding user interactions with an application and user-provided annotations [45]. OASIS deferred the delivery of desktop-based notifications until a breakpoint was detected. While both my system and OASIS use breakpoints, there are some significant differences. OASIS only focuses on users interacting with devices in desktop computing, whereas my solution for ubiquitous computing focuses on users interacting with multiple devices while mobile, including both user-device interaction and physical activities. Although OASIS employs a post-hoc breakpoint annotation, my system uses a real-time annotation scheme. Finally, OASIS was evaluated in the lab with a specific set of applications, whereas my valuation was performed in the wild on the user's own devices with their own applications.

5.4 Work in Mobile Computing Domain

Naturally, following the trend of emerging mobile computing, more recent works have been conducted in the domain of mobile computing.

Finding Breakpoints

Often referring the work by Iqbal *et al.* [45], breakpoint detection research has also been conducted in the context of mobile devices.

Ho *et al.* used wireless on-body accelerometers to trigger interruptions when users transition between activities [34]. Interruptions delivered at these transition times reduce user annoyance. This approach is promising but requires the use of an external on-body sensor,

being incompatible with the first requirement of **compatibility with the user's multiple mobile and wearable devices**.

Fischer *et al.* also identified breakpoints based on transitions between activities, but focused on moments immediately after phone-based activities including the completion of phone calls and text messages [23]. Users tended to be more responsive to notifications after these activities than at other random times. Again, this approach is promising, but is limited to a small set of communication activities. Referring to the solution requirements, this approach does not satisfy **applicability to diverse types of notification applications**.

Finding Interruptibility based on Contexts

Other researchers have focused on using a wider variety of user context to determine the moments of interruptibility.

Hofte *et al.* used an experience sampling methodology to collect information on the location, transit status, company, and activities in order to build a model of interruptibility [83], particularly for phone calls. Thus, this system has a limitation in terms of the requirement of **applicability to diverse types of notification applications**.

Pejovic *et al.* expanded the use of context for detecting moments of interruptibility on smartphones including user activity, location, time of day, emotions, and engagement. Their system, InterruptMe, uses this information to decide when to interrupt the user [68]. Compared with the requirements specified in this research, their system needs manually provided information regarding the user's interruptibility, such as their company or emotion. This approach leads to a limitation of the system in terms of **all-day long use** requirement. In contrast, my solution simply relies on sensor data from the user's devices and does not need any manual input.

5.5 Mitigating Notifications by Modality Adaptation

Other works in mobile computing have focused on mitigating the impact of notifications. This is a complementary approach to my focus on deferring notifications. Smith *et al.* attempted to mitigate the impact of disruptive phone calls by automatically setting the phone call ring tones to different modes, such as silent answering, declining, and ignoring [77]. Their user study showed that this approach to identifying which ring tone to use was useful even when underlying the change in user behavior.

Böhmer *et al.* also focused on incoming phone calls, and explored the design space of incoming call alerts to users on smartphones, instead of conventional full-screen notifications [11]. Their proposed strategies include "postponing" a call acceptance and "multiplexing" the alert screen. The multiplexing approach obtained the best evaluation in their large-scale user study, and can also be combined with my deferral approach. The postponing approach looks similar to my deferred scheduling approach, but is specifically focused on phone call notifications.

5.6 Attention with Multiple Devices

Research has also been conducted on attention-awareness in multi-device environments.

Dostal proposed DiffDisplays [22], a system for tracking the display the user is currently looking at by cameras and computer vision. Inspired by several techniques for visualizing changes in unattended displays, Garrido proposed AwToolkit [27] for developers to support maintaining user's awareness in multi-display systems. The toolkit detects which display is currently being looked at by the user and provides interruptive notifications with multiple different levels of "subtlety" to draw the user's attention to unattended changes in the displays.

Although this Gaze-tracking technique can be adopted in mobile environments, it is not considered to be compatible with diverse low-computation mobile and wearable devices, such as watches or bands, especially for the purpose of attention target classification. Attelia currently uses display on/off event for such classification. On the other hand, for activity recognition, gaze-tracking or blink-tracking has been used [47], thus it has potential as a source of information to use for breakpoint detection.

5.7 Summary

This section overviewed the literature related to the present research. Table 5.1 summarizes the major works related to this research and their fulfillment of requirement I specified in Section 4.3. As the table shows, none of the works fully satisfy all four requirements.

Table 5.1: Related Work and Their Fulfillment of Requirements

Work	Approach	Requirements			
		R1:**Compatibility with user's multiple mobile and wearable devices**	R2:**Applicability to diverse types of notification applications**	R3:**All-day-long use**	R4:**Real-time sensing**
Subjective Rating [33]	cognitive load	✓	✓		
Task Performance Measurements [67]	cognitive load				✓
Physiological Responses [32]	cognitive load		✓		✓
Horvitz [35]	interruptibility				
Hudson [38]	interruptibility				
Begole [9], Horvitz [36]	interruptibility				✓
Iqbal [45]	breakpoint		✓		✓
Ho [34]	breakpoint	(needs sensors)			✓
Fischer [23]	breakpoint	✓		✓	✓
Hofte [83]	interruptibility	✓		✓	✓
Pejovic [68]	interruptibility	✓	✓		✓

Starting from the next Chapter, this dissertation introduces my proposal, Attelia, which detects a user's breakpoint in mobile and ubiquitous computing situations, satisfying all of the four requirements.

Chapter 6

Attelia: Approach and Model

This section overviews Attelia, my proposal for attention status sensing. I introduce an overview of Attelia with some key features, followed by several technical approaches it employs. Finally, I will explain the multi-device hybrid breakpoint detection model, the core of attention sensing model in Attelia.

6.1 Overview of Attelia

As a solution for the interruption overload problem, this research proposes Attelia. Attelia is a middleware software system that detects user's interruptible timing, with the following four features to fulfill the requirements described in Section 4.3.

- Attelia **works on a user's mobile and wearable devices**, such as a smartphone, smart watch, and tablet, without the use of an external server or any psycho-physiological sensors, such as an ECG sensor.
- Attelia detects such timings **in real-time** (not post-hoc). Thus other functionalities of attention-awareness, such as adaptation and prediction, can be executed on the fly.
- Attelia detects such timing **opportunistically during the user's comprehensive computing life**, including both during a user's active interaction (manipulation) with their devices, and during other non-active periods, such as when carrying mobile wearable devices but not actively manipulating them.
- Attelia has compatibility with versatile sources of notifications. It detects such timing **without any modification to existing applications and services**.

Imagine the scenario illustrated in Figure 6.1, where Melissa carries, wears, or uses multiple devices, including her smart watch on her wrist, a smartphone, and a tablet. In the beginning, she is sitting down and doing office work on her tablet. After a while, she decides to take a coffee break. Melissa stands up, walks to the kitchen, pours some coffee, walks back to the lab, sits down on the couch, and enjoys her beverage. In the current computing environment, Melissa experiences notifications at "random" timings, that is, as they arrive on her devices. In other words, notifications from a variety of applications and services reach Melissa without any consideration of whether she is actually interruptible, causing her attention to be divided and possibly having a negative impact on her work productivity.

In contrast, Figure 6.2 revisits the same scenario of Melissa and shows how Attelia helps her situation. Using multiple types of sensing techniques, Attelia detects her interruptible times, both during her active device manipulation (interacting with applications on her tablet) and during her physical activities. Notifications from a variety of applications and services, originally delivered to Melissa at random times without Attelia, are now delivered to her with the detected interruptible timing. This notification delivery is less interruptive and lowers Melissa's cognitive load.

6.2 Real-Time Detection with Mobile Sensing and Machine Learning

As illustrated in Figure 6.3, Attelia uses mobile sensing and machine learning techniques on mobile and wearable devices for its real-time detection of the user's interruptible timing.

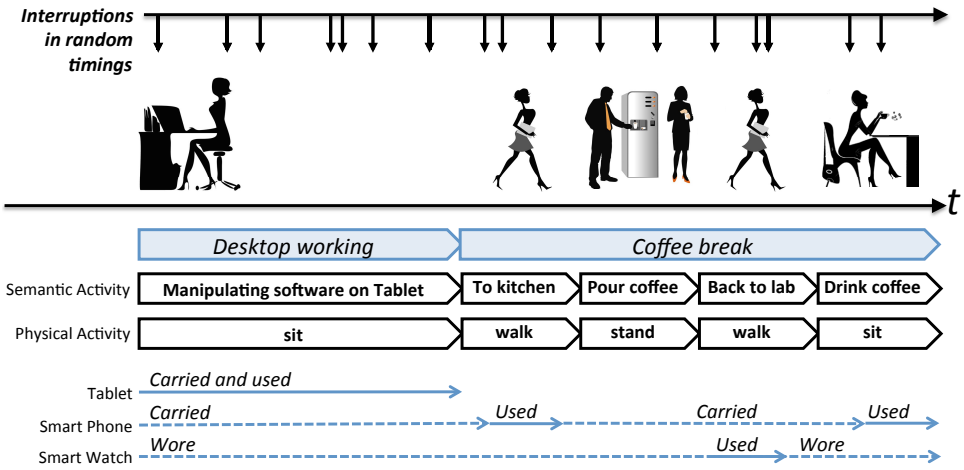


Figure 6.1: User's Notification Experience Scenario without Attelia

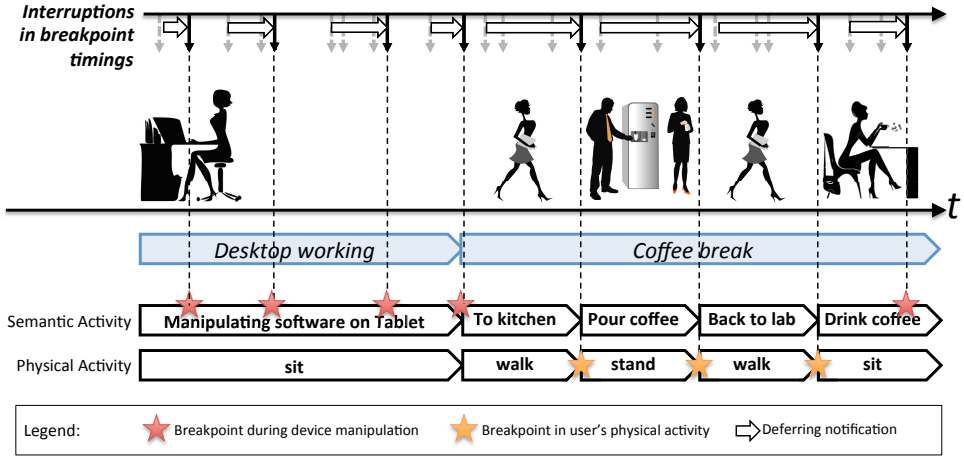


Figure 6.2: User's Notification Experience Scenario with Attelia

This basic idea of the data processing flow refers to several existing research systems on activity recognition [7, 54]. With this approach, Attelia's detection process is essentially to read sensor data from various types of sensors on the user's devices and to detect whether the current moment is within a user's interruptible timing or not.

Data from a sensor will be continuously and periodically read by the system. The data will be split into a **time frame** with a specific length T_f. With this specific periodicity, further processing of the sensor data is executed for each frame. First, the feature extractor calculates the vector of "features" (V) from the time-series sensor data. Next, the classifier inputs the calculated feature vector, classifies the given data into one of the pre-configured labels, and outputs the resulting label (C). Specifically in the Attelia system, this classification processing is either a binary classification of an "interruptible timing" or not. Thus,

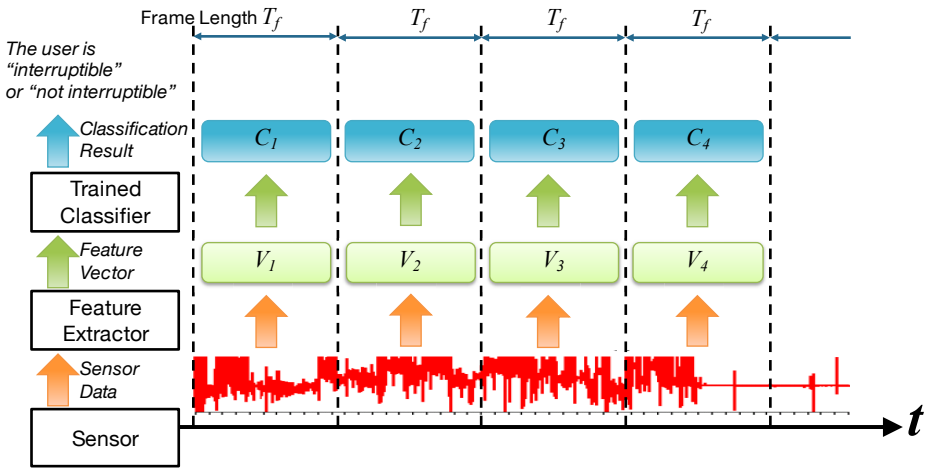


Figure 6.3: Mobile Sensing and Real-Time Detection in Attelia

for each frame, the classifier outputs a binary classification result.

Attelia also uses machine learning techniques to build a classifier that can actually classify the user's data, as shown in Figure 6.4. There are three different phases, namely (1) ground truth collection, (2) model training, and (3) real-time detection. In the ground truth collection phase, sensor data (training data) along with the ground truth on the occurrences of interruptible timing will be collected from multiple users. In the model training phase, a model (classifier) will be trained from the collected training data and the ground truth by using a machine learning engine. In the real-time detection phase, the trained classifier model will be installed into each user's mobile device and actual real-time detection, illustrated in Figure 6.3, will be executed on the devices.

6.3 Breakpoint as a Temporal Target

As the concrete timing of an interruptible timing (the target of the sensing described above), Attelia uses the concept of a **breakpoint** [63]. As introduced in Chapter 5, an approach with psycho-physiological sensors needs at least two sensors during non-mobile situations [32]. Other context-information based approaches need continuous manual information input by the user.

Given the burden of constantly wearing a psycho-physiological device and using manual inputs, the approach in Attelia employs a breakpoint, attempting to sense more coarse-grained but easier-to-sense signals, from which the appropriate timings for notifications can be inferred. Using this approach, Attelia works solely on the user's mobile and wearable devices, and does not need any external psycho-physiological sensors, such as EEG or ECG sensors.

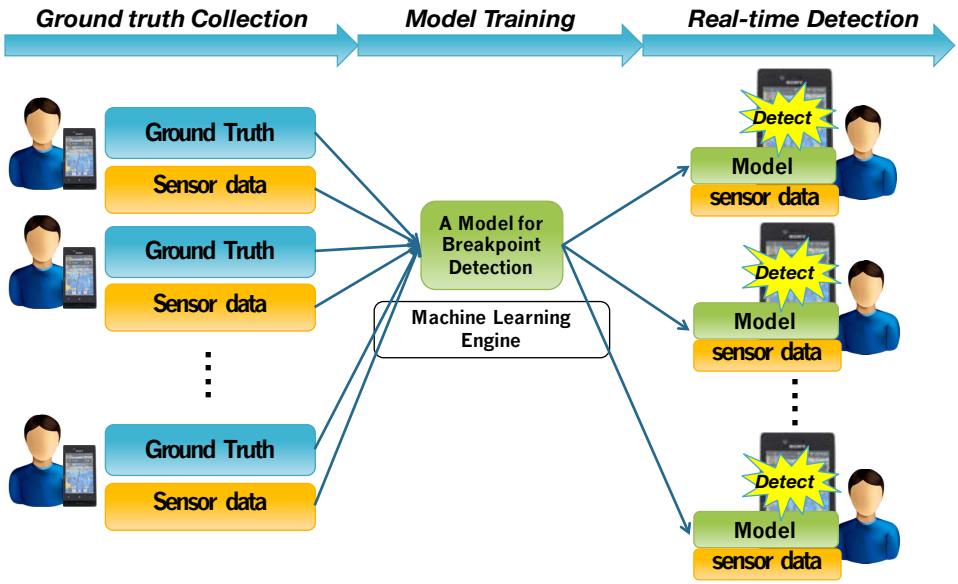


Figure 6.4: Overview of Phases of Machine Learning Technique Attelia Uses

Hybrid Breakpoint Detection

To support breakpoints in the user's everyday life (both during the user's active device interaction and during other "non-active" periods) of ubiquitous computing, Attelia detects the following two different types of breakpoints, namely, "user interaction-based breakpoints" and "physical activity-based breakpoints."

- **User Interaction-based Breakpoint:** This is a breakpoint in a user's device manipulation activity, such as using an application on a smartphone and manipulating a setting screen on a smart watch. In Melissa's example, this type of breakpoint is detected while Melissa is working on her tablet and using her smartphone during her coffee break.

  While the user is manipulating a device they are carrying or wearing, an application as the target of their manipulation exists on the device. Thus, during this type of period, Attelia focuses on the interaction between the user and application, and uses information on such interaction for detecting the user's breakpoints.

  Although the application itself is one possible source of knowledge about the breakpoints, using knowledge from the internals of any specific application is not feasible or scalable given the huge number of applications available and the fact that application developers would need to expose internal information at the development time. Instead, we collect the run-time status events from the operating system and executing applications, and use them to identify relationships to the ground-truth values of the interruptive overload provided by users during the training phase.

  More details on user interaction-based breakpoint will be described in Chapter 7.

- **Physical Activity-based Breakpoint:** This is a breakpoint in the user's physical activity, such as sitting, standing, walking, and running. More specifically, **changes** in these activities of users, such as "when a user sits down" or "when a user stops walking" are detected as a physical-activity based breakpoint in Attelia.

  In our daily lives using smartphones and smart watches, there is a significant amount of time when we simply carry or wear them but do not actively use (manipulate) them. For example, in Melissa's scenario, she wears her smart watch and carries her smartphone in her pocket but does not actively manipulate them while moving from the lab to the kitchen, getting coffee, and returning to the lab. Even during this type of period, various types of user applications and services may be processing information and trying to provide new information at a random times through interruptive notifications. To comprehensively address the information overload in a user's daily life, Attelia needs to handle this type of situation by finding an opportune moment to deliver notifications during this type of period.

  To this end, I focus on transitions in a user's physical activity, such as "when a user stands up" or "when a user stops running." I specifically hypothesize that when a person changes their activity from a high-energy state to a lower-energy state, such timing can be strongly considered as their breakpoint (later I validate this hypothesis with input gathered from users). Concretely, on mobile and wearable devices, Attelia declares a physical activity-based breakpoint when such a change in the user's activity is detected, using activity recognition mechanisms built on top of the hardware sensors already available on mobile platforms, such as an accelerometer or GPS.

  More details on physical activity-based breakpoints are given in Chapter 8.

6.4 Multi-Device Hybrid Breakpoint Detection Architecture

Combining (1) the described approach of the hybrid breakpoint detection and (2) a user's carrying of multiple mobile and wearable devices as an opportunity, Attelia introduces its original **multi-device hybrid breakpoint detection architecture** to detect the users' interruptible timing in their comprehensive everyday life of ubiquitous computing. Figure 6.5 illustrates such an architecture.

1. On each device, both the "User Interaction-based Breakpoint Detection" (while the device is being actively manipulated) and "Physical Activity-based Breakpoint Detection" (while the device is not being manipulated) will be running, according to the current device usage.

2. Each detection component executes its own local binary classifier to detect breakpoints at a configured periodicity and outputs the binary value. Such local classifi-

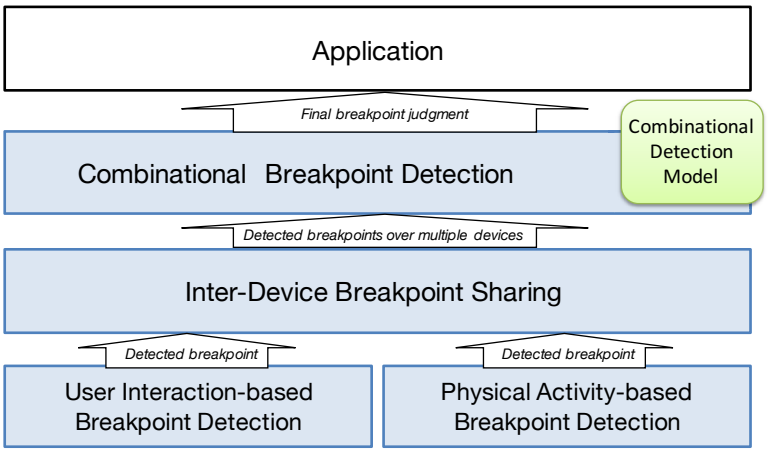


Figure 6.5: Attelia's Multi-Device Hybrid Breakpoint Detection Architecture

cation outputs along with the device usage status information are exchanged across a user's multiple devices through an "Inter-Device Breakpoint Sharing" layer.

3. An installed "Combinational Breakpoint Detection" algorithm reads the current values of all underlying local breakpoint detectors and device usage statuses, and generates a **final decision on the user's breakpoint status** across devices, based on the selected "Combinational Detection Model".

6.5 Attelia Prototypes: I and II

Figure 6.6 summarizes the detection models and device configuration covered in different prototype implementations of Attelia, namely Attelia I and II.

In Attelia I, the first prototype, research on the User Interaction-based Breakpoint Detection model on a single mobile device was explored. Following the first version, the second prototype, Attelia II addressed both User Interaction-based and Physical Activity-based Breakpoint Detection models on single and multiple device configurations. Starting from the next chapter,

Chapter 7 describes the research using Attelia I, followed by the research on Attelia II, described in Chapter 8.

6.6 Summary

This section described an overview of Attelia, with its features, technical approaches, and the main model of attention status sensing. Attelia detects a user's interruptible timing in real-time on their mobile and wearable devices without any external psycho-physiological sensors. Attelia detects such timing opportunistically during the user's comprehensive computing life. Attelia has compatibility with versatile sources of notification, without needing

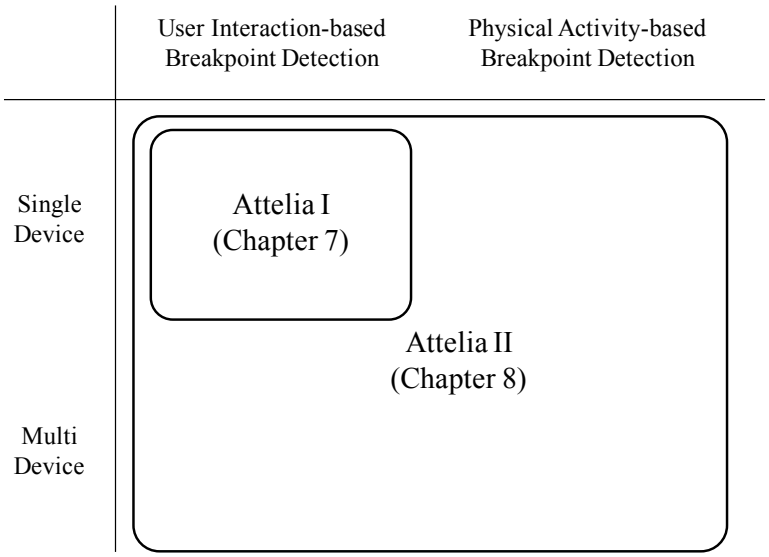


Figure 6.6: Attelia Prototypes and Covered Detection Models

any modification of the existing applications. The key technical approaches employed by Attelia are (1) the use of a breakpoint as a temporal target of detection, (2) real-time detection with mobile sensing and machine learning, (3) hybrid breakpoint detection, and (4) multi-device breakpoint detection. The multi-device hybrid breakpoint detection architecture consists of two different breakpoint detections, user interaction-based breakpoint detection and physical activity-based breakpoint detection among multiple devices to finally conclude the user's current breakpoint status.

The next chapter describes the research using Attelia I, focusing particularly on User-Interaction based Breakpoint Detection on a single device. After that, Chapter 8 describes Attelia II, the second prototype, which covers both breakpoint detection types on multiple devices.

Chapter 7

Breakpoint Detection on A Single Device

This chapter describes my first Attelia prototype “Attelia I”. Focusing on user’s “mobile experience” on a single mobile device, Attelia I detects user’s “user-interaction-based breakpoints” in real-time, solely on the smartphone device, without modification to versatile installed applications. This chapter details the design, implementation, and evaluation of Attelia I. Note that Attelia I is abbreviated as “Attelia” in some cases in this chapter.

7.1 Design of Attelia I

This section presents the design of Attelia I. As presented in the previous chapter, Attelia I particularly focuses on user's "mobile experience" during his/her active manipulation of devices, and finds appropriate timings for delivering interruptive notifications to users. Attelia I scopes such detection during their active engagement with mobile devices, and does not consider moments when users are not interacting with them.

Attelia I has the following three distinctive features:

- Attelia detects those timings on smartphones, without the use of an external server or any psycho-physiological sensors.
- Attelia detects such timings in real-time (not post-hoc) so that it can be used to adapt notification timings at run-time.
- Attelia's detection can be applied to a wide range of applications installed on users' smartphones, not requiring any modifications in to the applications.

As the target of the detection of "appropriate timing", Attelia I finds **user interaction-based breakpoints**, which is breakpoint during user's device manipulation. To detect such timings in real-time on smartphones, Attelia I deploys the concept of mobile sensing and real-time classification (of breakpoints) based on the machine learning approach.

7.1.1 Real-Time Detection with Mobile Sensing

Figure 7.1 illustrates the mobile sensing and real-time detection of breakpoints of Attelia on user's mobile devices. Sensor data from a sensor will be continuously and periodically read by the system. The data will be split into a time frame with length T_f. With that specific periodicity, further processing of sensor data executes for each frame basis. Firstly, feature extractor calculates a vector of features (V) from a time series sensor data. Classifier inputs the calculated feature vector, classifies the given data into one of pre-configured labels, and outputs the resulted label (C). Specifically in the Attelia system, this classification processing is a binary classification of breakpoint or not. For each frame, the classifier outputs a label of either "breakpoint" (meaning that the input data of the current time frame are classified as a breakpoint) or "non-breakpoint."

7.1.2 Use of Machine Learning Technique

Also, Figure 7.2 shows the overview of how machine learning technique is used in Attelia. To build a classifier that actually can classify user's data into either "breakpoint" or "non-breakpoint", Attelia utilizes the approach of machine learning.

There are three different phases, namely (1) Ground Truth Collection, (2) Model Training, and (3) Real-Time Detection. In the Ground Truth Collection phase, sensor data ("training data") along with a ground truth on the occurrences of breakpoint will be collected from

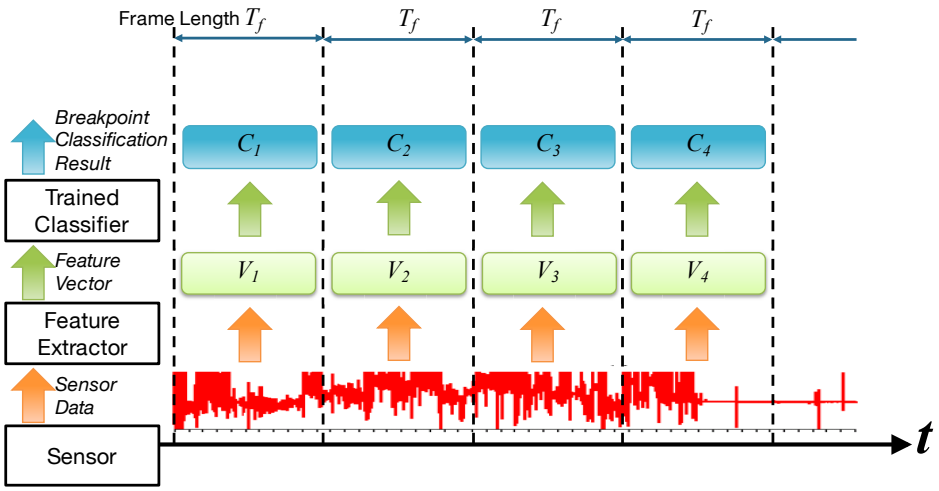


Figure 7.1: Mobile Sensing and Real-Time Detection in Attelia I

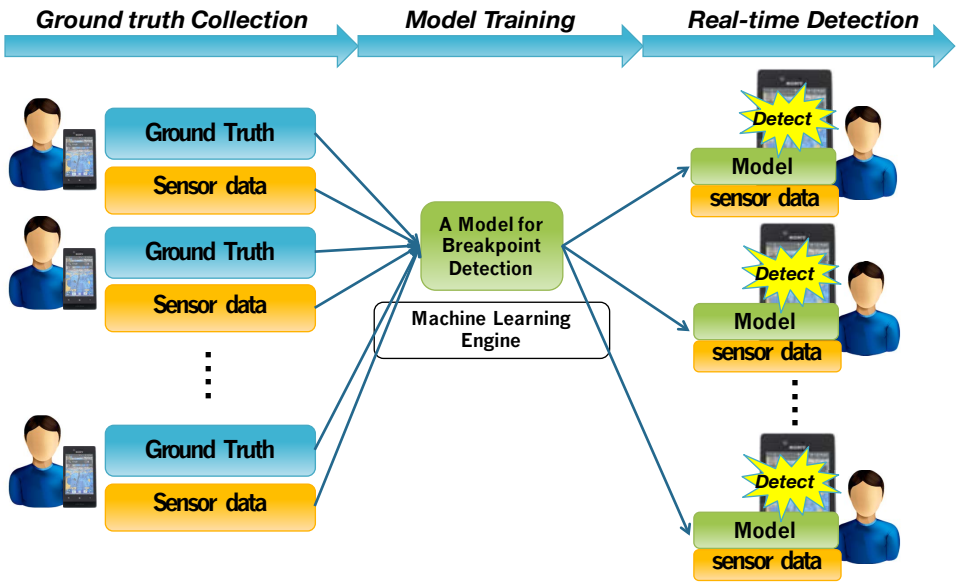


Figure 7.2: Machine Learning Approach that Attelia I Utilizes

multiple users. In the Model Training phase, a model (classifier) will be trained from the collected training data and the ground truth by using a machine learning engine. In the Real-Time Detection phase, the trained classifier model will be installed into each user's mobile device and actual real-time breakpoint detection, illustrated in Figure 7.1 will be executed on the devices.

7.1.3 User Interaction as a Sensor

With the scoping to active use of mobile devices, Attelia I focuses on how users interact with mobile applications running on the device and use that information as a sensor data for detecting user's breakpoints. When a user interacts with a mobile device, such as smart phones, the interaction always involves in manipulation of an application running on the foreground of the device, as the "target" of interaction. Thus I focused on the interaction between a user and such application.

Particularly for the breakpoint detection during the mobile experience periods, Attelia focuses on device (and running application) usage and not physical sensors, despite their wide proliferation on mobile devices for two reasons: simplicity of implementation and reducing the reliance on a sensor that may not exist on all target mobile devices (or may be mounted in different locations).

Table 7.1 shows some possible knowledge sources for identifying breakpoint and Table 7.2 shows, for each source type, how it can be acquired. The application-related knowledge and information can include both relatively static knowledge that is specific to each application, such as when users transition between multiple "stages" in game applications, and that are designed and implemented by the application developers in the development phase; and relatively dynamic information, such as run-time status and events that result from the running applications. Using knowledge from the internals of any specific application is not feasible given the huge number of applications available and the fact that application developers would need to expose internal information at development time. Instead, I collect run-time status events from the operating system and executing applications, and use them to identify relationships to ground truth values of interruptive overload provided by users, in the Ground Truth Collection phase.

Table 7.1: Approaches of Knowledge Collection for Breakpoint Detection

Approaches on Knowledge Source of Breakpoint	Examples of Data Types
Application-specific breakpoint knowledge	explicit breakpoint declaration inside application, explicit future breakpoint forecast inside application
Runtime status/event of systems and applications	stack trace, number of threads, thread names, memory consumption Android API invocation, system call invocation, rendered screen image, Low-level GUI events, switches between applications

7.2 Attelia I System

Figure 7.3 shows the system structure of Attelia implemented on the Android 4 platform. Attelia consists of an Android service that includes several internal components for UI event logging, breakpoint ground truth annotation logging, as well as a machine learning engine that performs feature extraction and classification (using an embedded Weka [56] engine).

Table 7.2: Timings of Knowledge Input and Data Collection

Approaches on Knowledge Source of Breakpoint	Knowledge on Breakpoints: When? By Who? and How?		Data Collection at Application Run-Time
	Application Development Phase	System Training Phase	
Application-specific breakpoint knowledge	Embedding additional API calls to provide explicit breakpoint knowledge (by application developer)	None	From API calls embedded inside running applications
Runtime status/event of systems and applications	None	Ground truth annotation of collected status/event information (by application users)	From the middleware and operating system

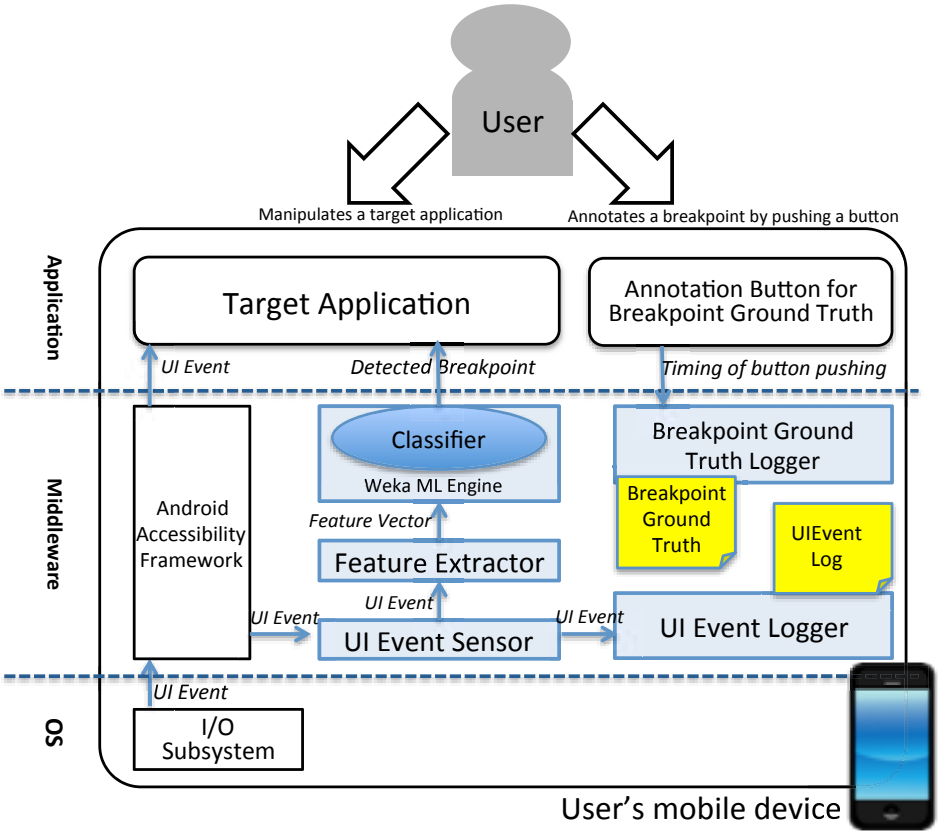


Figure 7.3: System Architecture of Attelia I on Android Platform

7.2.1 Execution Modes

Attelia can execute in ground truth annotation mode, off-line training mode or real-time breakpoint detection mode. In the annotation and detection modes, the UIEventLogger component listens to the stream of incoming UI events and records relevant events to the log file.

- **Ground truth collection:** In this mode, users manually provide ground truth about breakpoints during application usage. Figure 7.4 shows a screen-shot of Attelia, with

my Annotation widget floating on the screen. While manipulating ordinary Android applications, users push the floating button when they are switching activities. The Attelia service records the stream of UI events (excluding those from the annotation button) and breakpoint timestamps (moments when the annotation button was pushed).

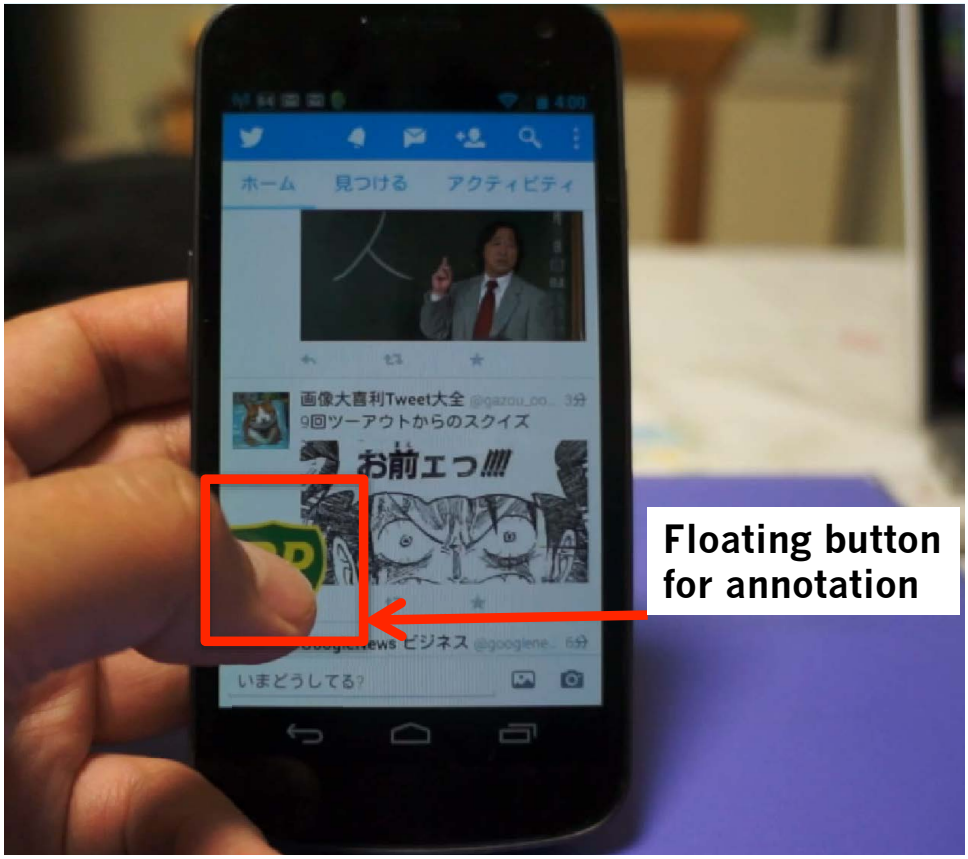


Figure 7.4: Ground Truth Annotation with Attelia I

- **Off-line model training:** In this mode, feature extraction and classifier training is executed off-line, using the previously-stored sensor and ground truth data.
- **Real-time mobile breakpoint detection:** Sensing, feature extraction, and classification with a previously-trained model is performed in real-time on a smartphone.

7.2.2 Sensing Data

To obtain the stream of UI events from the middleware, I use the Android Accessibility Framework [28] provided by Android OS. This framework was originally for supporting those who have visual, physical or age-related limitations. Since those users may not be able to seeing or using touch screen or to hear the audible information, several different

applications and services, including text-to-speech, haptic feedback, or gesture navigation are already implemented by Google and other software developers on top this framework.

Using this framework, with one-time explicit permission setting operation by the user, Attelia can collect UI events and data about the UI components the user is interacting with. A list of the UI Events I collect is shown in Table 7.3.

Table 7.3: UI Events Collected in Attelia I

Event Types	Events
View	`View clicked`, `View long clicked`, `View selected`, `View focused`, `View text changed`, `View selection changed`, `View text traversed at movement granularity`, `View scrolled`
Transition	`Window state changed`, `Window content changed`
Notification	`Notification state changed`

7.2.3 Feature Vector

From these events, Attelia I extracts 45 features outlined in Table 7.4. These features are extracted for the data within each **frame**. In preparing the features, I attempted to be exhaustive in providing possible features to capture as many characteristics of the real execution environment as possible.

During the ground truth collection execution mode, the calculated feature vector values will be stored into a local storage in the device. In the off-line training phase, those values are read and fed in to the Weka [56] machine learning system to train a model.

In contrast, in the real-time mobile breakpoint detection mode, the result vector values will be used in breakpoint classification on the fly. After a feature vector is calculated periodically, with the periodicity of T_f, the resulted values will be directly fed in to a configured classifier on the mobile device.

7.2.4 Ground Truth Collection and Model Training

Ground Truth Collection

To collect ground truth for the model training, I conducted a small user study. In the experiment, eight participants were recruited. All of them are university undergraduate and graduate students and staff with ages between 18 and 27 years, who use smartphones daily. During the experiment, participants were handed a Samsung Galaxy Nexus [72] smartphone running Android version 4 as well as the Attelia I itself. Each participant was asked to manipulated five common Android applications (Twitter, Yahoo News, YouTube, Kindle, Browser) for 5 minutes each (per application) performing everyday tasks. Also, during the application manipulation, they were asked to annotate their subjective breakpoint timings

Table 7.4: Features Used in Attelia I

Feature Types	Features
Rate of occurrence of each UI Event type inside the frame	snipped (one for each event type presented in Table 7.3)
Statistics on the status of the event source UI component	*rate(*`isEnabled`*)*, *rate(*`isChecked`*)*, *rate(*`isPassword`*)*
Statistics on the events' timings in the frame	*min_timegap*, *mean_timegap*, *max_timegap*, *stdev_timegap*
Statistics on the location of the event source UI components	*min.*, *mean.*, *max.*, *stdev.*, *the value of the smallest rectangle*, *the value of the biggest rectangle* of `X-left`, `X-right`, `X-width`, `Y-top`, `Y-bottom`, `Y-height`

by pushing the floating annotator button appearing on the smartphone screen. Attelia was running in the ground truth collection mode, thus their UI event sensor data, along with the timestamps of their breakpoint annotation, were stored into the local storage of each phone.

Model Training

Model training was done with Weka 3.7.9 off-line, after downloading the sensor data and feature vector files from all phones. The training was done by a original Perl script that controls several different data processings one by one. The overview of the data processing flow is as follows.

1. For each sensor data file, a time series sensor data, along with the breakpoint ground truth data, will be split into a pre-configured time frame length T_f.
2. For each time frame, a feature vector gets calculated from the sensor data. (Thus the number of resulting vectors will equal to the number of time frames.)
3. The order of resulted vectors will be randomized, using "randomize" plug-in of Weka.
4. The balance between vectors with "breakpoint" ground truth annotation and ones with "non-breakpoint" annotation gets roughly evened, using "resample" plug-in of Weka.
5. The feature vectors will be fed into Weka engine to model a train.

Frame Length and Accuracy

With an expectation that my choice of time frame length T_f will affect my ability to perform breakpoint detection, I build several different models with different T_f settings and compared its detection accuracy.

Figure 7.5 shows the classification accuracy results with different frame lengths (0.25 to 5 seconds), using 10-fold cross validation on Weka 3.7.9 and J48 classifier. The data for each application is aggregated from all eight participants, and is represented as a separate line in the graph. An additional line in the graph (bolded) represents all application data aggregated together from all the participants. Accuracy is low when the frame length is very short (e.g., 0.25 seconds), because there are not enough sensed UI events within that time span to achieve a high classification accuracy. However, around 2 to 2.5 seconds, the accuracy begins to stabilize. At the 2.5-second setting, accuracy was 82.6%, precision was 82.7% and recall was 82.3%.

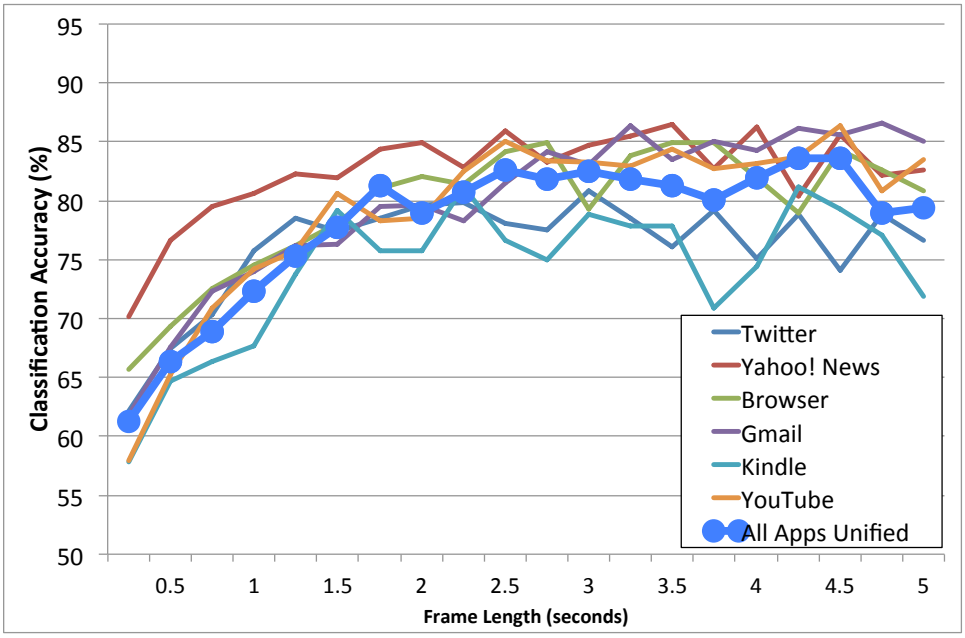


Figure 7.5: Classification Accuracy and Frame Length

Trained Model

The trained J48 classifier model includes 281 leaves and the total number of 561 nodes. The top 10 features with the biggest information gain are shown in Table 7.5. Features such as `TimeGap_min`, `TimeGap_max`, `TimeGap_mean` and `TimeGap_stdev` are the minimum, maximum, mean, and standard deviations of time gaps between two consecutive UI event in the frame. Features whose names end with `_rate` are the rate of occurrence of a specific event type (defined in the Android Accessibility Framework) among all event times in a time frame.

Table 7.5: The Top 10 Features with the Biggest Information Gain in the Model Training Data

Information Gain	Feature Name
0.07589	TimeGap_min
0.05022	TYPE_WINDOW_CONTENT_CHANGED_rate
0.04335	TimeGap_max
0.03951	TimeGap_mean
0.03933	TYPE_VIEW_SCROLLED_rate
0.03657	isEnabled_rate
0.02823	TYPE_VIEW_TEXT_CHANGED_rate
0.02138	TimeGap_stdev
0.01503	TYPE_VIEW_ACCESSIBILITY_FOCUS_CLEARED_rate
0.01294	TYPE_VIEW_TEXT_SELECTION_CHANGED_rate

7.2.5 Power Saving

To save power, Attelia I disables real-time feature extraction and classification when the device screen is off, as the system is concerned with detecting breakpoints when the user is engaged with the device. In addition, if no UI event occurs within a given time frame, no classification is performed.

Table 7.6 shows a power comparison between using my UI events and using common sensors. I used a Samsung Galaxy Nexus with Android 4.4.4 and measured the data with a Monsoon Power Monitor [59]. Each table value is the average of five 5-minute measurements.

Table 7.6: Comparisons of Power Consumption Overhead

Sensor Type	Setting	Frequency (Hz)	Overhead (mW)
UI Events	N/A	10	51.70
Accelerometer	“Fastest”	120	102.90
	“Game”	60	48.76
	“UI”	15	12.08
Gyroscope	“Fastest”	100	158.88
	“Game”	50	129.24
	“UI”	15	74.04

In Attelia I, since the number of incoming UI events depends on user interaction, I looked to my user study data to determine an appropriate number. Based on the data collected from 30 users for 16 days, the average number of UI events was 10.6 per second on average ($min = 1$, $max = 549$, $stdev. = 15.1$) during users' active manipulation of their device. I then logged the power consumption using Android instrumentation that fired approximately 10 UI events every second.

To compare to the other sensors, I implemented a basic application which reads and stores the sensor data with the specified speed settings among various different types of preset settings available in the Android API, such as “Fastest”, “Game”, and “UI”.

The result shows that the overhead of my UIEvent data collection software is quite low compared with other sensors and considering that multiple types of sensors, such as the accelerometer, gyroscope and GPS, are used in combination for related systems [68, 83] that detect user’s interruptibility by using data from these sensors.

7.2.6 Portable Implementation

Attelia I is implemented as a “Service” inside the Android platform. By appropriately setting the permissions for the service, it can log the stream of UI events, such as tapping, clicking, and scrolling or modifications of UI components inside the currently-active Android application without requiring root privileges. This implementation allows the service to be distributed through the Google Play store and contributes to the deployability of the system to end users.

7.3 Evaluation: Controlled User Study

To further understand how Attelia I works, I conducted a controlled user study based on my implementation. The overall purpose of study was to investigate if providing notifications to users at the timing of breakpoints detected in real-time lowers user’s workload perception.

7.3.1 Participants

For the study, 37 participants were recruited. Among them there were university students, staff members, and research engineers, with ages between 19 and 54. All the participants were smartphone users in their daily lives. Subjects were not told the specific objectives of the study at the beginning, and not paid for the participation.

7.3.2 Experimental Setup

For the study, I prepared Samsung Galaxy Nexus smartphones with Android 4.3 for each participant. The original notification feature of each phone was disabled. For my experiment, I installed my Attelia I prototype software and six representative Android applications (Twitter, Gmail, Yahoo News, YouTube, Kindle, and Browser) in to each phone. The Attelia I service was configured to “real-time mobile breakpoint detection” mode, with a J48 decision tree classifier trained through my previous experiment, with 2.5-second time frame T_f setting.

I prepared four different notification strategies in this study, namely (1) disabled (no notification at all), (2) random timing (emulating a conventional notification situation), (3)

breakpoint timing (my approach), and (4) non-breakpoint timing (interrupting at times that my system determines as inopportune). The approaches (2), (3), and (4) were configured to have intervals of at least 30 seconds between two consecutive interruptions. During the study, each participant was exposed to one of the four different notification strategies for being interrupted by notifications. Strategies were changed for each participant, and for each session. The order of the selection of the strategies was randomized, and the information on the selection was not revealed to the participants.

On the interruptive tasks, a full screen pop-up window appeared on the screen to ensure that the interruption would not go unnoticed, when participants were interrupted. The pop-up contained the first paragraph from a random news article. During interruption, the participants were given a interruptive task: To read the paragraph and select an appropriate title for the article given three options. I chose this interruptive task from the similar previous interruption studies [5,6]. Subjects were asked to finished the task as fast and accurately as possible. After the participant finished the task, the pop-up window disappeared so that the user could return to the original task that she was performing.

7.3.3 Experiment Procedure

My experimental procedure contained two parts. In the first part, each user was given a printed email and was told to compose and send an email with the specified text using the Gmail app. Each user repeated this task five times, with different text and different notification strategy. In the second part, each user was asked to use each of the other selected applications (Twitter, Yahoo News, YouTube, Kindle, Browser) as they "normally would" for 5 minutes each, and experienced a different notification strategy with each application.

The order of the email texts (part 1), applications (part 2), and notification strategies were counterbalanced using a balanced Latin Square to remove ordering effects. Since there were 4 strategies, and the email and app use tasks were performed 5 times, each user saw one strategy twice, which was randomly selected. A repeated measures within-subject design was used with the notification strategy as factors.

7.3.4 Measurements

To measure participant's subjective perception of workload, I used the web page version of the NASA-TLX [33] questionnaire. Each participant was asked to answer the questionnaire after each task (i.e., a total of 10 times per participant).

7.3.5 Result Analysis: Subjective Workload

As shown in Figure 7.6, I observed differences in the range of subjective workload (NASA-TLX Weighted Workload (WWL) score) in terms of their individual means and variances across different notification strategies. More specifically, I noticed that some of my participants were more sensitive (higher variance in their WWL) to the different notification

strategies. Also some of my participants seem to not react (e.g., insensitive) to the notification strategies (low variance in their WWL). This fact motivated us to try to identify clusters within my user population.

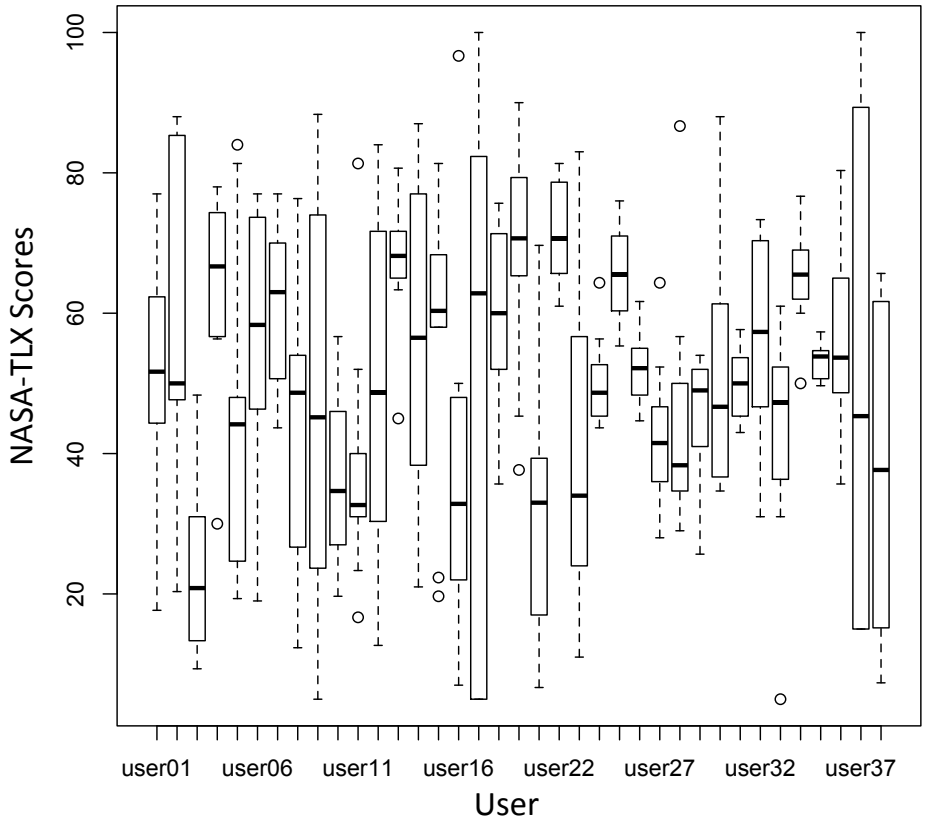


Figure 7.6: Variance of NASA-TLX WWL Scores (Controlled User Study)

I conducted hierarchical clustering (using the Ward method and Euclidean distance) on the variance of each participant's NASA-TLX WWL scores, in order to observe the dissimilarity between users. Figure 7.7 shows the resulted dendrogram.

When looking at the dendrogram, it is quite obvious that these users can be split into two quite distinctive clusters since the height from the top of the figure to where two vertical lines (i.e., two biggest clusters) further splits to more groups in this figure with using the Ward method and the Euclidean distance. Thus I decided to do further analysis on each of these 2 clusters.

The number of participants and the mean of personal WWL score standard deviation in each cluster are shown in Table 7.7. I named these cluster "WWL-sensitive users" (those with higher score variance among the different strategies) and "WWL-insensitive users", since this clustering was on the variance of each participant's **personal score variance**.

(For further confirmation on clustering, I also tried non-hierarchical clustering with K-means (K=2) with the Hartigan-Wong method. The results both from the hierarchical clus-

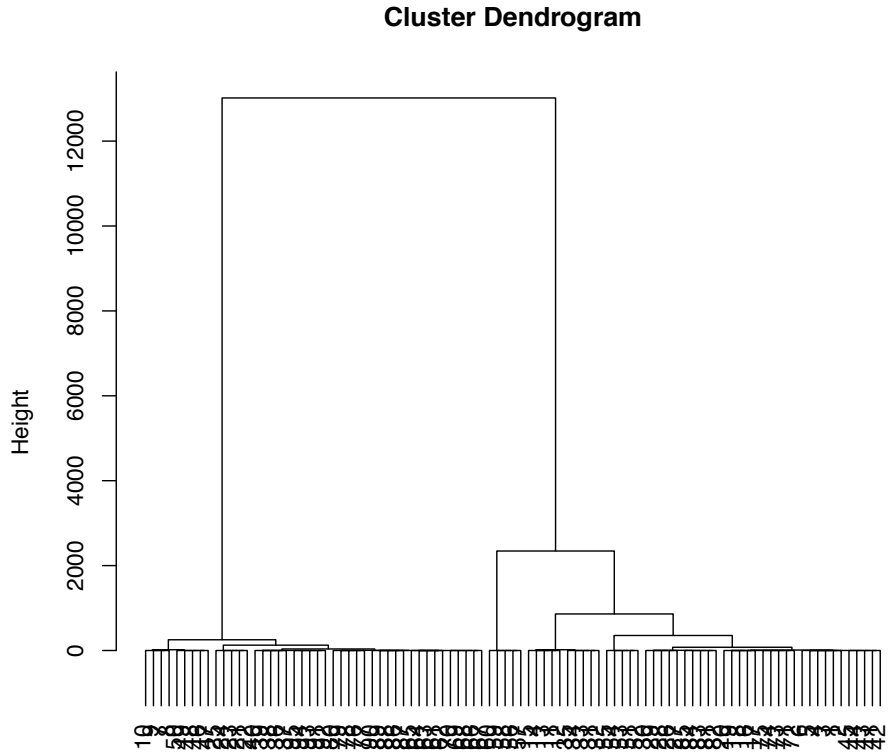


Figure 7.7: Dendrogram from Structured Clustering (Personal WWL Score Variances) (Controlled User Study)

tering and the non-hierarchical clustering were identical, having the same users in each cluster.)

Table 7.7: Two WWL-based Clusters in the Controlled User Study

Cluster name	Users	Mean WWL Stdev.
"WWL-sensitive users"	19	23.11
"WWL-insensitive users"	18	9.92

Figure 7.8 shows the average NASA-TLX WWL scores for the different notification strategies, for the two clusters respectively.

The most significant finding in this analysis is that, for the "WWL-sensitive users", a 46% decrease in perceived workload was observed in my breakpoint strategy ("BP") results, compared to the workload in the random strategy ("Random"), that emulates how people are currently experiencing interruptions on the standard Android notification system. "BP" strategy (workload score of 44.56) resulted in only an increase of 35% in workload when compared to the baseline "Disabled" strategy (workload score of 32.95) with no notifica-

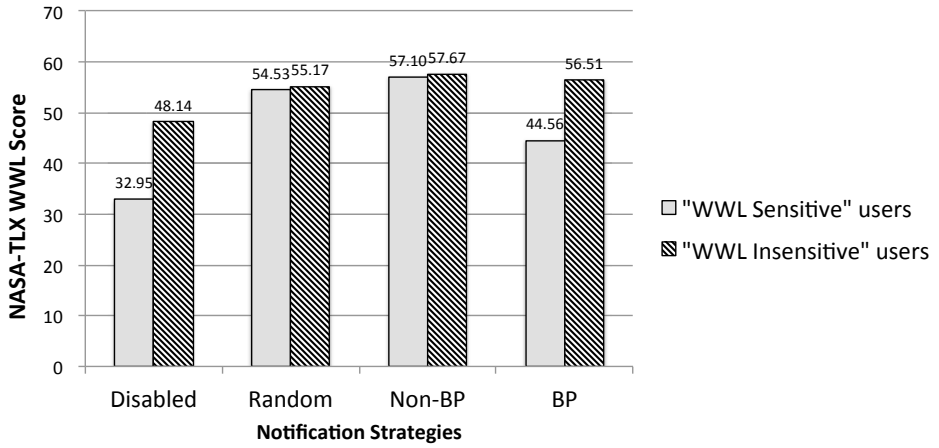


Figure 7.8: NASA-TLX WWL Scores for Each Cluster (Controlled User Study)

tions, while "random" strategy (workload score of 54.53) resulted in an increase of 66%. Also, as I expected, the non-breakpoint strategy ("Non-BP"), where notifications were be intentionally displayed only at the timings that were not detected as breakpoints, resulted in the highest 73% increase in workload, with a score of 57.10.

A Friedman test revealed a significant effect of notification strategy on the WWL score ($\chi^2(3) = 16.5, p < 0.05$). A post-hoc pair-wise comparison using Wilcoxon rank sum tests with Holm correction showed the significant differences between "Disabled" and "Random" ($p < 0.01, \gamma = 0.34$), between "Disabled" and "non-BP" ($p < 0.01, \gamma = 0.39$), between "Disabled" and "BP" ($p < 0.05, \gamma = 0.29$), between "Random" and "BP" ($p < 0.05$, $\gamma = 0.24$), and between "non-BP" and "BP" ($p < 0.05, \gamma = 0.26$). Between "Random" and "non-BP", a statistical difference was not observed.

On the other hand, For the "WWL-insensitive users", the result shows the insensitivity of the participants. As expected, from my Friedman test and pair-wise test with Wilcoxon rank sum tests, no significant differences were observed during "Random", "BP", and "non-BP", while significant differences between "Disabled" and the other strategies were found (Friedman test with the effect of notification strategy on the WWL score ($\chi^2(3) = 9.4, p < 0.05$)). The significant differences from the post-hoc test using Wilcoxon rank sum tests with Holm correction are observed between "Disabled" and "Random" ($p < 0.01, \gamma = 0.30$), between "Disabled" and "non-BP" ($p < 0.01, \gamma = 0.35$), and between "Disabled" and "BP" ($p < 0.01, \gamma = 0.34$).

7.4 Evaluation: In-the-Wild User Study

Based on the promising results from my controlled user study, I proceeded to in-the-wild user study to better understand how Attelia I could reduce user's perceived workload in the user's real computing lives. In this study, I installed my Attelia I service on each partic-

ipant's smartphone. I compared multiple different notification strategies and investigated if notifications displayed at the timings of detected breakpoints could reduce participants' workload.

7.4.1 Participants

For this study, 30 (20 male and 10 female) people, who are using an Android 4.3 (or above) smartphone in their daily lives, were recruited as the participants. Among the participants there were university staff members and students, with ages ranging from 18 to 29 years old. 20 participants belonged to computer science and information technology related departments, while the remaining participants belonged to other schools, such as social sciences, economics, and psychology. All of the participants were using Android OS version 4.3 (or above) smartphones in their daily lives. Subjects were paid $60 for their participation.

7.4.2 Experimental Setup

I packaged the Attelia I service and some additional experiment-related data collection services and their parameters into a single Android service. With each participant's permission, I installed the service to each participant's own smartphone. The Attelia I service was configured to the real-time mobile breakpoint detection mode, with a J48 decision tree classifier trained using my previous experiment, with a 2.5-second time frame T_f setting. In this study, I prepared three different notification strategies, namely (1)"Disabled" (no notification), (2)"Random", and (3)"Breakpoint" (my approach). Everyday, for each user, the data collection logic randomly chose one of these strategies to be used for notification throughout the day.

I set the following study-specific parameters for each user: (1)the daily maximum number of interruptive tasks to 12, (2)the minimum interval between two consecutive notifications was set to 15 minutes, (3)the maximum interval was set to 30 minutes, (4)the service was configured to show notifications only from 8AM to 9PM daily. These parameter values were carefully chosen to get enough data points without requiring too much effort from the participants. The last was estimated from interviews to the participants about their daily life patterns.

Regarding on the interruptive task, a full screen pop-up window appeared on the screen to ensure that the interruption would not go unnoticed, when participants were interrupted. Figure 7.9 shows the screen-shots of the pop-up interruptive notifications.

The first screen asked if the timing was during a natural breakpoint. The second pop-up was shown regardless of the user's answer to the question. On the pop-up, the participants were given a interruptive task: To read the paragraph and select an appropriate title for the article given three options. I chose this interruptive task from the similar previous interruption studies [5, 6]. Subjects were asked to finished the task as fast and accurately as possible. After the participant finished the task, the pop-up window disappeared so that the user could return to the original task that she was performing.

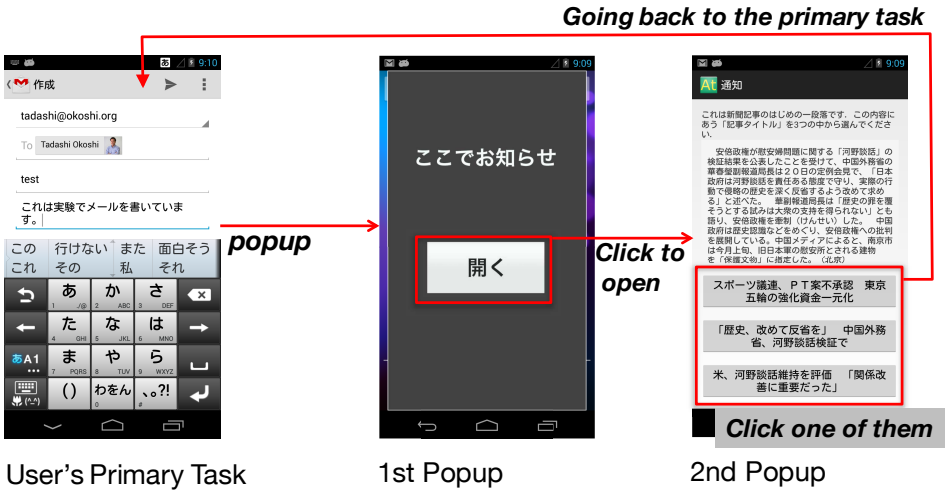


Figure 7.9: Notification Screens

7.4.3 Experiment Procedure

My experimental procedure consists of the following three parts.

1. Each participant had a meeting with a study researcher at the beginning of the user study. The participant received basic information and instructions on the study, followed by signing a consent form. Afterwards, the researcher installed and started the Attelia I software on the participant's smartphone. The existence of multiple different notification strategies was explained to the participants, but the detailed behavior was not explained.

2. The 16-day long experiment started after the meeting. As mentioned above, every day the notification strategy for each user was randomly changed. Information about the notification strategy working every day was not revealed to the participant. During the experiment, at the end of each day, a NASA-TLX survey was sent to all participants. Each participant was required to individually answer NASA-TLX survey every night, for 16 days.

3. After the 16-day period finished, participants filled out the post-experiment survey, uninstalled the Attelia I service, and were paid.

7.4.4 Measurements

The Attelia I service recorded the time taken to respond to the first and second notifications, time to answer the quiz, and the answer to the quiz. The data was uploaded to my server every night. The NASA-TLX questionnaires (implemented as a web page on my web server) were sent to each user via email every night, thus the survey results were stored inside my database on the server.

7.4.5 Result Analysis: Subjective Workload

From the experiment, I collected the answers to NASA-TLX surveys from each of 30 participants over 16 days. The data from 3 users was discarded due to several issues: data not properly recorded and uploaded to the server or the user forgot to fill out the daily survey. My final data set consisted of 27 users' data and I used it for the following data analysis.

As shown in Figure 7.10, again, I observed differences in the range of subjective NASA-TLX WWL score in terms of individual personal means and variances across different notification strategies. More specifically, I observed once again sensitive and insensitive (to the notification strategy) users.

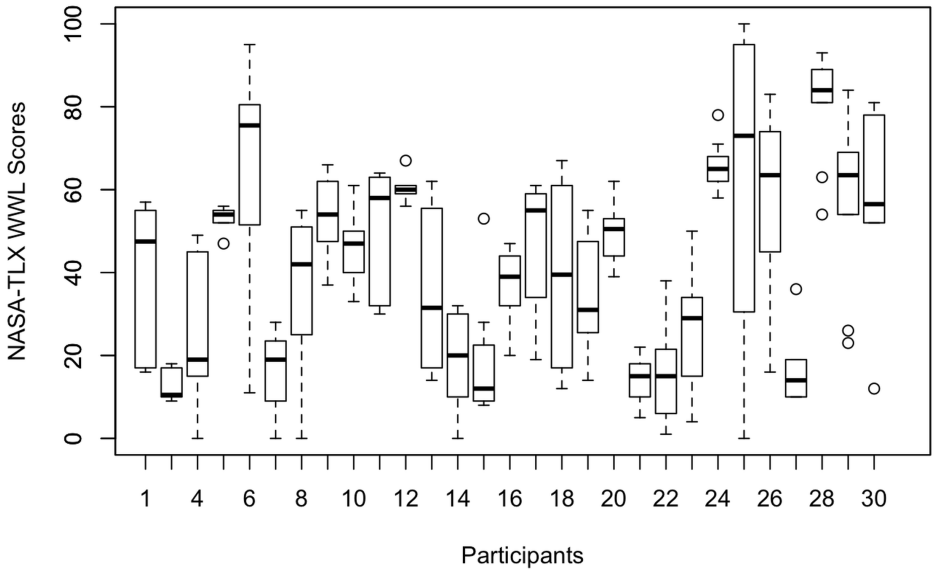


Figure 7.10: Variance of NASA-TLX WWL Scores (In-the-Wild User Study)

Thus, I first conducted a hierarchical clustering with the Ward method and Euclidean distance on the variance of each user's NASA-TLX WWL scores. Figure 7.11 shows the resulting dendrogram for this clustering.

When looking at the dendrogram, I again identified 2 distinct clusters. It is quite obvious that these users can be split into two quite distinctive clusters since the height from the top of the figure to where two vertical lines (i.e., two biggest clusters) further splits to more groups in this figure with using the Ward method and the Euclidean distance. Table 7.8 shows the the number of users and the mean of personal WWL score standard deviation in each cluster. Following my naming convention in the controlled study, I named the clusters "WWL-sensitive users" and "WWL-insensitive users" respectively.

Also, for further confirmation on clustering, I conducted another non-hierarchical clustering with K-means algorithm (K=2) with the Hartigan-Wong method, and confirmed that the both clustering methods output the same clustering of the users.

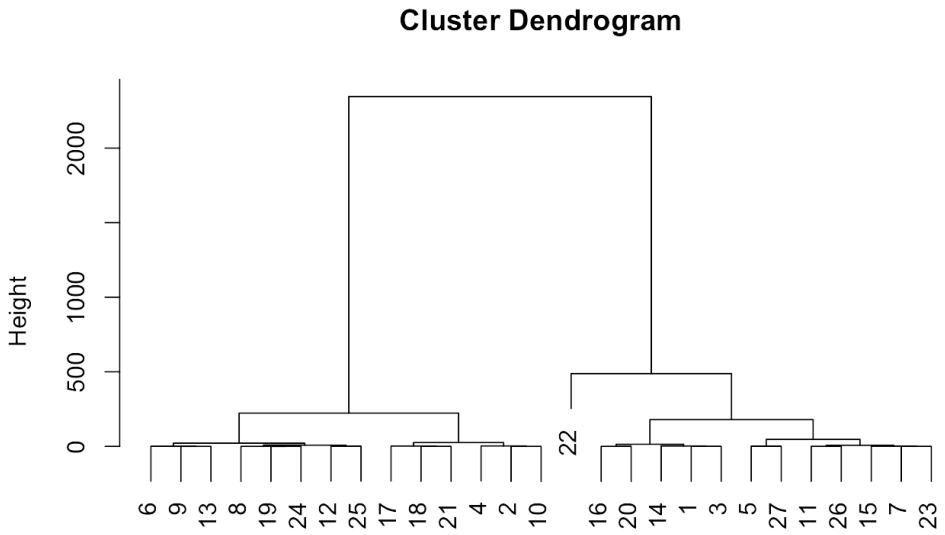


Figure 7.11: Dendrogram from Structured Clustering (Personal WWL Score Variances) (In-the-Wild Study)

Table 7.8: Two WWL-based Clusters in In-the-Wild User Study

Cluster name	Users	Mean WWL Stdev.
"WWL-sensitive users"	13	21.38
"WWL-insensitive users"	14	8.19

The average NASA-TLX WWL scores, for each notification strategies and for each cluster, are illustrated in Figure 7.12.

For the "WWL-sensitive users", the results show the same trend as I observed in the controlled user study. A 33% decrease in perceived workload was observed in my breakpoint strategy ("Breakpoint") results, compared to the perceived workload in the random strategy ("Random"), that emulates how people are currently experiencing interruptions on the standard Android notification system. "Breakpoint" strategy (workload score of 45.46) resulted in only an increase of 33% in perceived workload when compared to the baseline "Disabled" strategy (workload score of 34.22) with no notifications, while "Random" strategy (workload score of 51.07) resulted in an increase of 49%.

A Friedman test revealed a significant effect of notification strategy on the WWL score ($\chi^2(2) = 8.5, p < 0.05$). A post-hoc pair-wise comparison using Wilcoxon rank sum tests with Holm correction showed significant differences between "Disabled" and "Random" ($p < 0.01, \gamma = 0.37$) and between "Random" and "Breakpoint" ($p < 0.05, \gamma = 0.20$),

For "WWL-insensitive users", on the other hand, my Friedman test and pair-wise test with Wilcoxon rank sum tests showed no significant differences between all of three strategies.

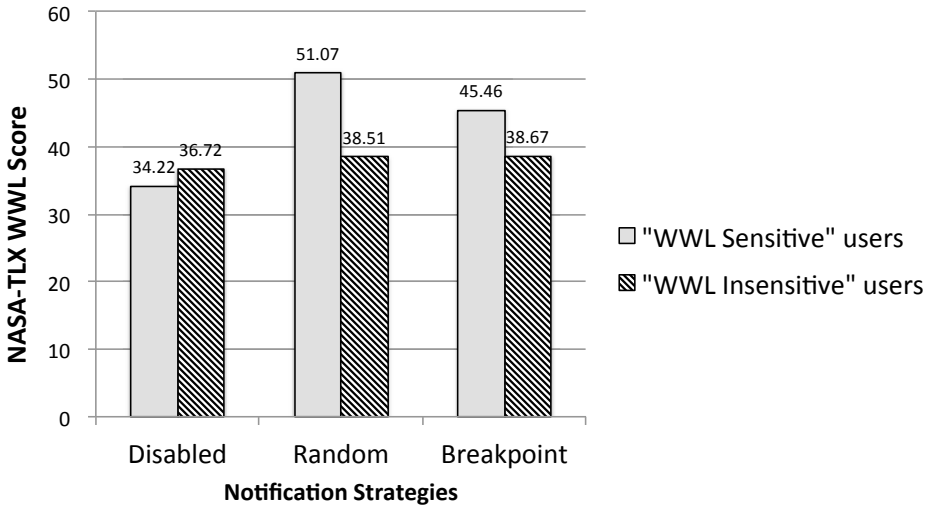


Figure 7.12: NASA-TLX WWL Scores for Each Cluster (In-the-Wild User Study)

7.4.6 Result Analysis: Subjective Frustration

I also conducted another analysis on user's subjective frustration value collected in daily NASA-TLX surveys, since the frustration value looks the key element among 6 different elements in the survey. Figure 7.13 shows the variances of the frustration scores for all participants.

Similarly, for the frustration scores, since differences in the variance among users were observed, I first conducted a hierarchical clustering using the Ward method and Euclidean distance on the variance of each user's NASA-TLX frustration scores. The resulted dendrogram is shown in Figure 7.14.

Similar to the analysis on the WWL score variances, I observe a separation between the observed 2 clusters, thus concluded the size of the clusters to 2. (I also confirmed that a non-hierarchical clustering with K-means algorithm (K=2) with the Hartigan-Wong method output the same clustering result.)

Table 7.9 shows the number of users and the mean personal frustration score standard deviation in each cluster respectively. Also, Table 7.10 shows the comparisons between the WWL-based clustering and the frustration-based clustering. All of 13 participants in the "WWL-sensitive users" cluster are clustered in to "FRU-sensitive users" cluster. On the other hand, 10 out of 14 users in "WWL-insensitive users" are clustered in to "FRU-insensitive users" while other 4 users are clustered in to "FRU-sensitive users".

Figure 7.15 shows the average frustration scores for the different notification strategies, for the two clusters respectively.

For the "FRU-sensitive cluster", I observe the same trend as I saw in my WWL score analysis. A 33% decrease in frustration was observed in my breakpoint strategy ("Breakpoint") results, compared to the perceived workload in the random strategy ("Random").

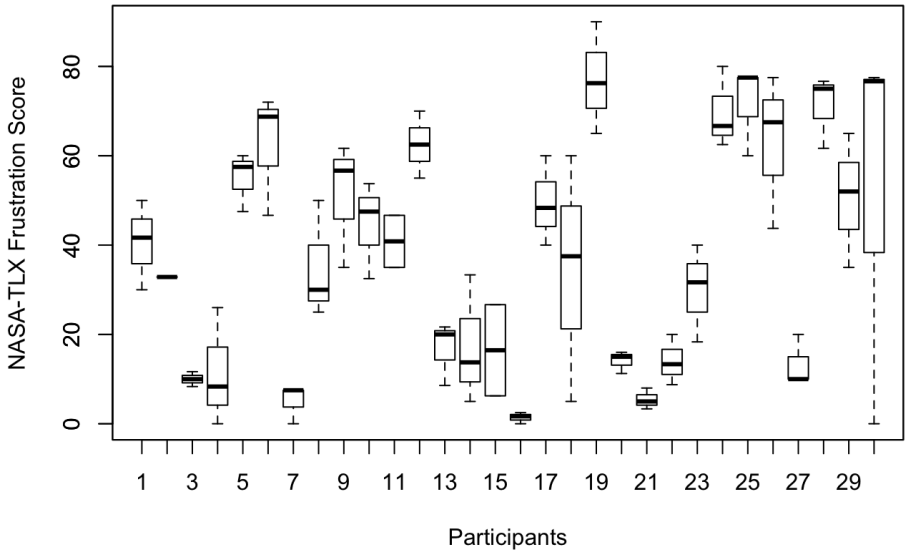


Figure 7.13: Variance of Frustration Scores

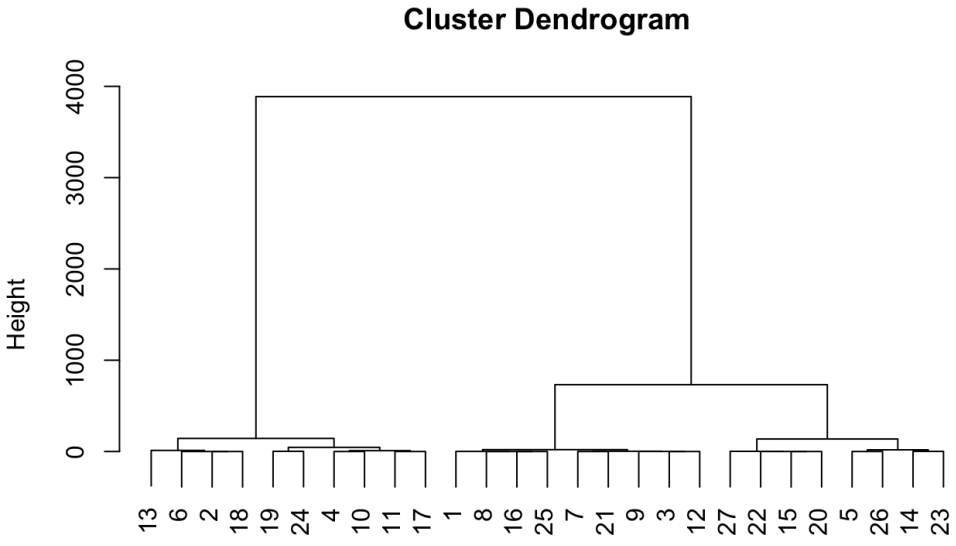


Figure 7.14: Dendrogram from Structured Clustering (Personal Frustration Score Variances) (In-the-Wild User Study)

"Breakpoint" strategy (frustration score of 50.74) resulted in only an increase of 27% in perceived workload when compared to the baseline "Disabled" strategy (perceived workload score of 39.77) with no notifications, while "Random" strategy (perceived workload

Table 7.9: Two Frustration Score-based Clusters in In-the-Wild User Study

Cluster name	Users	Mean Frustration Stdev.
"FRU-sensitive users"	17	25.29
"FRU-insensitive users"	10	7.72

Table 7.10: Comparisons between Two Clustering Analysis

	FRU-sensitive users	FRU-insensitive users
WWL-sensitive users	13	0
WWL-insensitive users	4	10

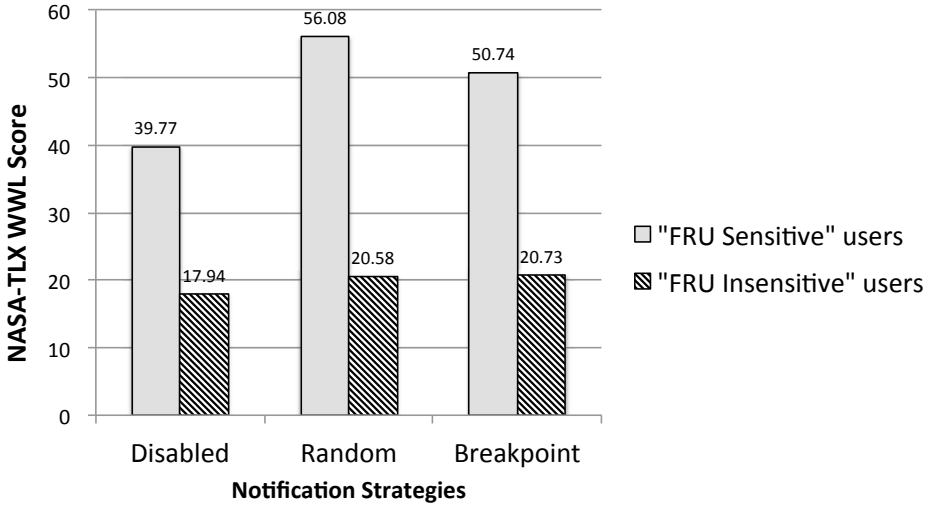


Figure 7.15: Frustration Scores for Each Cluster

score of 50.74) resulted in an increase of 41%. A Friedman test revealed a significant effect of notification strategy on the WWL score ($\chi^2(2) = 4.7, p < 0.05$). A post-hoc pair-wise comparison using Wilcoxon rank sum tests with Holm correction showed significant differences between "Disabled" and "Random" ($p < 0.05$, $\gamma = 0.33$), between "Disabled" and "Breakpoint" ($p < 0.05$, $\gamma = 0.22$), and between "Random" and "Breakpoint" ($p < 0.05$, $\gamma = 0.17$).

On the other hand, for the "FRU-insensitive users", no significant differences were observed between all of three strategies, by my Friedman test and pair-wise test with Wilcoxon rank sum tests.

7.4.7 Result Analysis: Response Time for the First Pop-up

Figure 7.16 shows my next analysis on the response time to the first pop-up. The response time is the time differences from when the first pop-up is shown on the smartphone screen to when it was answered by the user. After the user study, I obtained 1130 data points for the "Random" strategy and 1032 data points for the "Breakpoint" strategy. The average response time was 3.18 seconds in "Random" and 2.77 seconds in "Breakpoint" respectively. My Wilcoxon Signed-rank test showed that there is a significant effect of strategy ($W = 343$, $Z = -3.19$, $p < 0.05$, $\gamma = 0.37$).

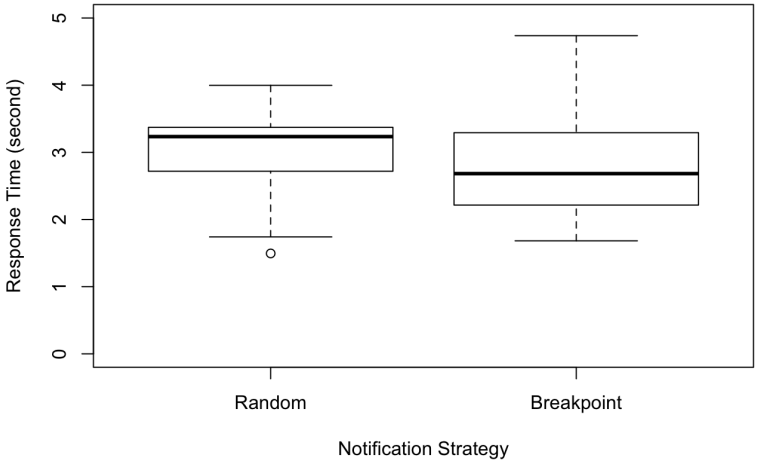


Figure 7.16: Response Time to the First Pop-up

7.4.8 Result Analysis: Response Time for the Second Pop-up

Next I analyzed response time for the corresponding second pop-up questions. Again, the response time is the time differences from when the second pop-up is shown on the screen to when it was answered by the user. Figure 7.17 shows the results. The average response time in "Random" strategy is 5.97 seconds while the average response time in "Breakpoint" strategy is 5.88 seconds. My Wilcoxon Signed-rank test did not show significant difference the response time values of the strategies. Also, the same types of tests combined with the clustering (either WWL or frustration scores) did not show any significant difference.

From this analysis result, my hypothesis is that, since the target of user's attention was already switched from the user's primary task to the interruption at the timing of the first pop-up, regardless of the type of the notification strategies used, the response time values for the second pop-up are not significantly different between the notification strategies (of the first pop-up).

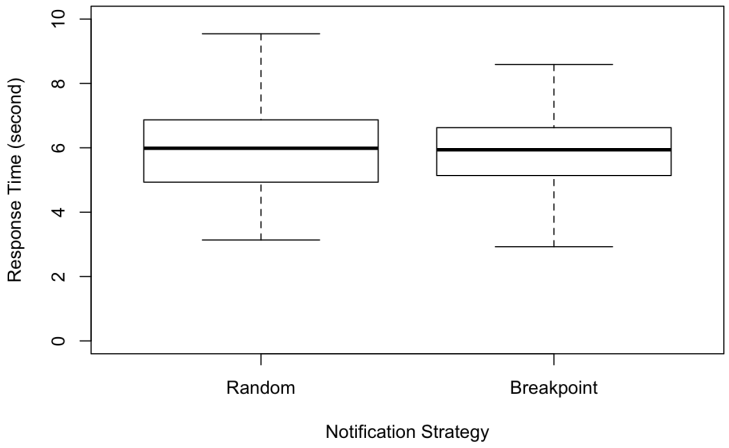


Figure 7.17: Response Time to the Second Pop-up

7.4.9 Result Analysis: Correct Answer Rate for the Second Pop-up

Another analysis on the second pop-up was on the correct answer rate for the second pop-up screen. Figure 7.18 shows the results. The correct answer rate is 87.0% in "Random" strategy and 87.8% in "Breakpoint" strategy. However, my Wilcoxon Signed-rank test did not show significant difference the response time values of the strategies. Furthermore, the same types of tests combined with the clustering (either WWL or frustration scores) did not show any significant difference. This analysis result supports my hypothesis on the target of user's attention mentioned above.

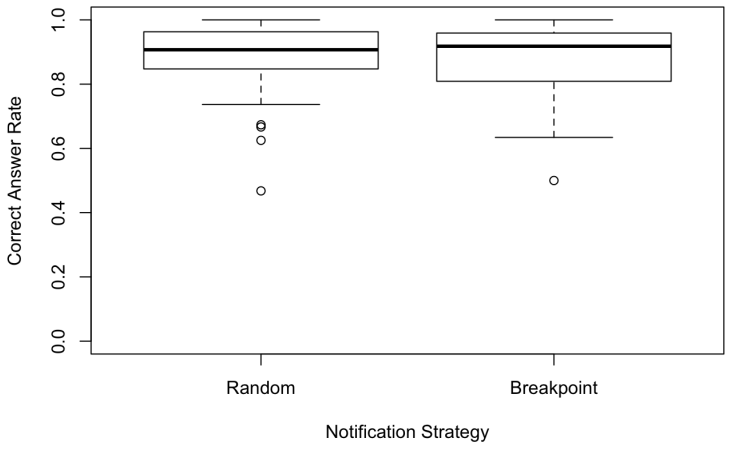


Figure 7.18: Correct Answer Rate in the Second Pop-up

7.4.10 Post-Experiment Survey

After the 16-day experiment has finished, I conducted an instant post-experiment survey for each user, at the end of the user study. Table 7.11 and 7.12 summarizes the results. I asked each participant following 5 questions.

1. Did you realize any differences between several different notification strategies that changed daily, excluding "No notification" strategy?
2. Do you think the differences in Q (1) affected your mental workload?
3. Do you think the differences in Q (1) affected your response time to answer the quiz?
4. With my software, did you see any difference in your phone's battery life?
5. With my software, did you observe any performance degradation in your phone?

For questions 1 to 3, I asked the participants to answer each question by using 5-level likert scale (1 – Strong disagree, 2 – Disagree, 3 – Neutral, 4 – Agree, and 5 – Strongly agree). For questions 4 and 5, I asked the participants to answer each question by using another 5-likert scale (1 – Not at all influential, 2 – Slightly influential, 3 – Somewhat influential, 4 – Very influential, 5 – Extremely influential).

Table 7.11 summarizes the answers of question 1 to 3. For the question (1) "Did you realize any differences between several different notification strategies that changed daily, excluding "No notification" strategy?", the answer with the biggest number of the participants was "Disagree", while the answer with the second biggest number of the participants was "Agree". From this result, I hypothesize that this answer may be related to the clusters I observed in my NASA-TLX score analysis. However, I could not further analyze these survey answers in terms of possible matching against the cluster I previously generated, since there were several data inconsistencies in the user ID field of the survey answer data. At the survey, I asked each participant to input her/his own user ID manually. However, I eventually found that several participants have input the wrong user ID number, then it was not possible to analyze the data, referring their user ID values.

For the question (2) "Do you think the differences in Q (1) affected your mental workload?", more than half of the participants (16 out of 30) agrees or strongly agree, while 7 participants disagrees or strongly disagree. The same trend was observed for the question (3) "Do you think the differences in Q (1) affected your response time to answer the quiz?". 18 participants agreed or strongly agreed to the question, while 5 participants disagreed or strongly disagreed. Again, I could not analyze the results further on the relationships with the observed clusters, due to the inconsistent data on the user IDs.

Table 7.12 summarizes the answers of question 4 and 5. For the question (4) "With our software, did you see any difference in your phone's battery life?", 10 out of 30 participants answered that it was not influential at all. Although 20 participants were aware of some level of change in power consumption, the total number of users who answered "Not at all

Table 7.11: Summary of the Post-Experiment Survey (1)

Question	"Strongly disagree"(1)	"Disagree"(2)	"Neutral"(3)	"Agree"(4)	"Strongly agree"(5)	Average	Std. Dev.
1	6	10	5	7	2	2.6	1.2
2	3	4	7	15	1	3.2	1.1
3	2	3	7	15	3	3.5	1.0

- *Question (1): Did you realize any differences between several different notification strategies that changed daily, excluding "No notification" strategy?*
- *Question (2): Do you think the differences in Q(1) affected your mental workload?*
- *Question (3): Do you think the differences in Q(1) affected your response time to answer the quiz?*

influential" (10) and "Slightly influential" (10) covers 20, which is 2/3 of the participants. The result was quite promising for us in terms of Attelia I's power efficiency.

Table 7.12: Summary of the Post-Experiment Survey (2)

Question	"Not at all influential"(1)	"Slightly influential"(2)	"Somewhat influential"(3)	"Very influential"(4)	"Extremely influential"(5)	Average	Std. Dev.
4	10	10	6	4	0	2.1	1.0
5	14	9	6	0	1	1.8	1.0

- *Question (4): With our software, did you see any difference in your phone's battery life?*
- *Question (5): With our software, did you observe any performance degradation in your phone?*

7.5 Summary

This chapter addressed my research on detecting breakpoints during user's mobile experience (interaction with a device) on a single mobile device, introducing the detail design, implementation, and evaluation results of Attelia I.

A controlled user study showed that notifications at detected breakpoint timing resulted in 46% lower perceived workload compared to randomly-timed notifications. Furthermore, my in-the-wild user study with 30 participants for 16 days further validated Attelia's value, with a 33% decrease in perceived workload compared to randomly-timed notifications.

Chapter 8

Breakpoint Detection on Multiple Devices

This chapter describes “Attelia II”, the second prototype of Attelia. Attelia II is cable of detecting breakpoints in a user’s daily life comprehensively, both during user’s active manipulation and inactive periods. Attelia II also addresses breakpoint detection in a combination of multiple mobile and wearable devices. This chapter details the design, implementation, and evaluation of Attelia II.

8.1 Design of Attelia II

Encouraged by my promising results with Attelia I, this research builds the second prototype, Attelia II, on my past work in two novel and important ways. First, this research address how breakpoint detection can be applied in the **multi-device** (i.e., smartphones and smart watches) ubiquitous computing environments that users are often in, and use these devices to detect breakpoints. Second, Attelia I only detected breakpoints during active interaction with a smartphone. Attelia II extends the breakpoint detection to cover all aspects of a user's daily life, including the period the devices are carried or worn but not actively manipulated. I demonstrate the impact of this increased coverage on users' perceived workload.

Attelia I has the following three distinctive features:

- Attelia II detects breakpoints on a combination of user's mobile and wearable devices, without the use of an external server or any psycho-physiological sensors.
- Attelia II detects such timings in real-time (not post-hoc) so that it can be used to adapt notification timings at run-time.
- Attelia II's detection can be applied to a wide range of applications installed on users' smartphones, not requiring any modifications in to the applications.

8.1.1 Two Types of Breakpoints as Temporal Targets for Interruption

Referring to the results from my previous work and other related research, Attelia II uses breakpoints [63] as a temporal target for sensing an opportune moment for delivering interruptive notifications with reduced user perceived workload.

In order to cover user's comprehensive everyday life in ubiquitous computing, Attelia II introduces two different notions of breakpoints, namely **User Interaction-based Breakpoint** and **Physical Activity-based Breakpoint**.

User Interaction-based Breakpoint

While the user is manipulating a device that he is carrying or wearing, there is an application that is the target of his manipulation. Thus, for the period when the device is actively being manipulated, I focus on the interaction between the user and the application and use that information for detecting a user's breakpoints.

Although the application itself is one possible source of knowledge about breakpoints, using knowledge from the internals of any specific application is not feasible nor scalable, given the huge number of applications available and the fact that application developers would need to expose internal information at development time. Instead, we collect runtime status events from the operating system and executing applications, and use them to identify relationships to ground truth values of interruptive overload provided by users, during a training phase. During this training phase, users indicate when they are interruptible

by pressing an always-present button on their interface (see Figure 8.3.) This training data is provided to a J48 classifier running on the mobile device. Note that the User Interaction-Based Breakpoint Detection described here is the same as what was presented in Attelia I.

Physical Activity-based Breakpoint

In my daily lives with smartphones and smart watches, there is a significant amount of time when I just carry or wear them but do not actively use (manipulate) them. For example, in Melissa's scenario, she wears her smart watch and carries her smartphone in her pocket but does not actively manipulate them while moving from the lab to the kitchen, getting coffee and returning to the lab. Another example is when a user is just reading a book, sitting on a sofa, and wearing his smart watch. To comprehensively address information overload in a user's daily life, this type of situation needs to be handled, by finding an opportune moment to deliver notifications while users are not actively manipulating their devices.

To this end, I focus on transitions in a user's physical activity, such as "when a user stands up" or "when a user stops running". I specifically hypothesize that when a person changes her activity from a high energy state to a lower energy state, that timing can be strongly considered as her breakpoint. (Later I validate this hypothesis with input gathered from users.) Concretely on the mobile and wearable devices, Attelia II declares a physical activity-based breakpoint when such a change in the user's activity is detected, using activity recognition mechanisms built on top of the hardware sensors already available on mobile platforms, such as the accelerometer or GPS.

8.1.2 Mobile Sensing to Real-Time Breakpoint Detection

In order to realize real-time detection on multiple mobile and wearable devices, Attelia II has its own architecture for overall breakpoint detection shown in Figure 8.1.

(1) On each device, both the "User Interaction-based Breakpoint Detection" (while the device is being actively manipulated) and "Physical Activity-based Breakpoint Detection" (while the device is not being manipulated) will be running, according to the current device usage.

(2) Each detection component executes its own local binary classifier to detect breakpoints at a configured periodicity and outputs the binary value. Those local classification outputs along with the device usage status information will be exchanged across a user's multiple devices via an "Inter-Device Breakpoint Sharing" layer.

(3) An installed "Combinational Breakpoint Detection" algorithm reads the current values of all underlying local breakpoint detectors and device usage statuses, and generates a **final decision on the user's breakpoint status** across devices, based on the selected "Combinational Detection Model".

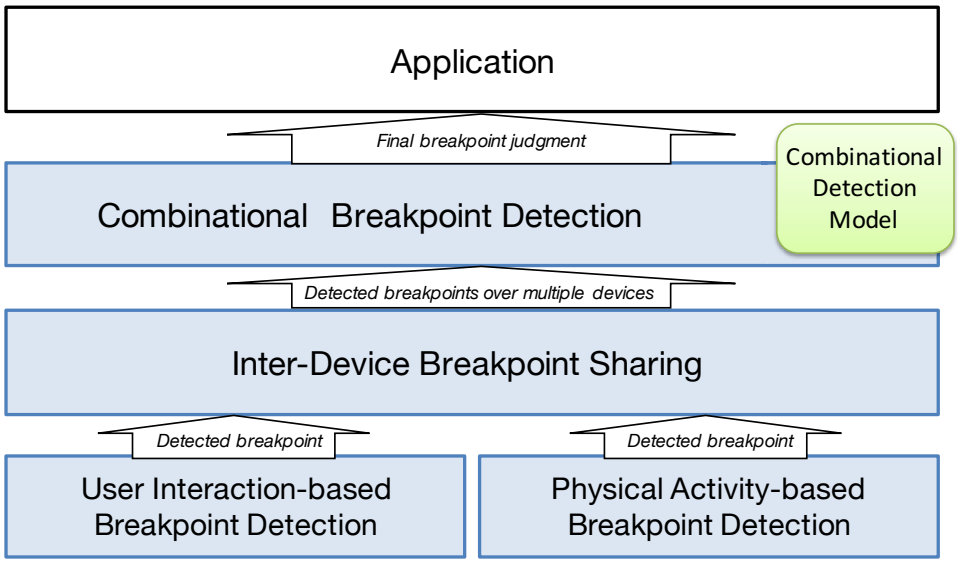


Figure 8.1: Attelia II Layered Breakpoint Detection Architecture

8.2 Attelia II System

Based on the design above, here I describe the system architecture of Attelia II implemented on the Android platform in this section. Figure 8.2 shows the system structure of Attelia II prototype implemented on the generic Android 4.3 (and above) platform and Android Wear 5 (and above) platform. The current prototype runs on a variety of Android devices including smartphones, tablets, notebooks, smart cameras, and smart watches as shown in Figure 8.3.

Attelia II is implemented as a middleware service for the Android platform and runs on each device of the user. The middleware implementation allows the service to be distributed through the Google Play store and contributes to the deployability of the system to end users.

Attelia II uses several underlying components inside the Android platform which are illustrated in "Underlying Components" layer in Figure 8.2. Each individual breakpoint detector on each device reads a data stream from the underlying systems, such as activity recognition results or the UI event stream and detects breakpoints, by using its own feature extraction and classification (powered by Weka [56]) logic, respectively. (More detailed information on the underlying components used in each platform are summarized in Table 8.1 and explained in the next section.)

These detection results ("Detected breakpoints" in Figure 8.2) are fed into "Inter-Device Breakpoint Sharing" component and exchanged among multiple devices in over Bluetooth-based Personal Area Network (PAN). When breakpoint(s) is detected by at least one of the low-level breakpoint detectors, the "Combinational Breakpoint Detector" component combines these results by following the definition in a configured "Combinational Detection Model" and produces a final breakpoint judgment.

Each device runs an identically selected Combinational Breakpoint Detector, that has

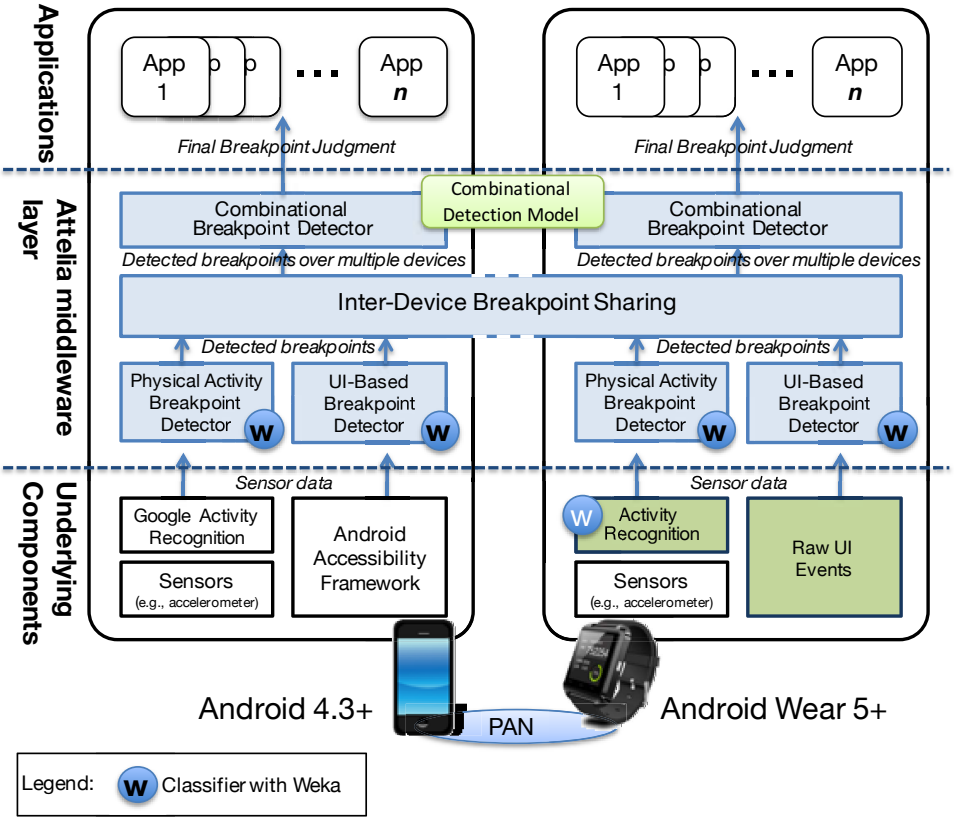


Figure 8.2: Attelia II System Architecture

access to the same information as the others (e.g., which detectors detected a breakpoint and which devices are actively being used). When they make a final judgment that a breakpoint has occurred, they use the device usage information to determine which device should deliver any deferred notifications. For example, if the phone is being used, the phone should receive the notifications since it already has the user's attention.

Table 8.1: Breakpoint Detection Mechanisms in Attelia II

	Generic Android platform	Android Wear platform
User Interaction-based Detector	Accessibility Framework [28]	Linux Input Subsystem [3]
Physical Activity-based Detector	Google Play Services Location APIs [30]	Original accelerometer -based activity recognition

8.2.1 User Interaction-based Breakpoint Detection

Table 8.1 shows the list of mechanisms used for each detector.

Figure 8.3: Attelia II on Diverse Devices: Notebook, Phone, Tablet, Camera and Watch

Generic Android Platform

On the generic Android platform (version 4.3 and above), Attelia II uses the same user interaction-based breakpoint detection mechanism as Attelia I presented in Chapter 7. Using the Android Accessibility Framework [28], the detector reads the UI event stream, extracts feature vectors, and executes a J48 classifier on the Weka [56] machine learning engine every 2.5 seconds, while the device is being actively used (more specifically, while the device's screen is on and there is any UI Event fired in this time frame).

Android Wear Platform

On the Android Wear 5 platform, on the other hand, I implemented a different breakpoint detector which uses the Linux Input Subsystem [3] since the Android Accessibility Framework is not provided as of version 5.0.1.

Linux Input Subsystem is a standardized way of manage all USB input devices in Linux. From the perspective of a system that uses this subsystem, available input devices on the local device are abstracted as device files under `/dev/input/` on the Linux file system. For example, Figure 8.4 shows the list of device files on Sony SmartWatch3. By opening the device file and reading the content, applications can easily detect the input to a specific input device. Attelia II particularly uses `/dev/input/event1` to get user's tapping and gesture inputs on to the touch screen of the Android Wear smart watch.

Figure 8.4: List of Linux Input Device Files on Sony SmartWatch3

```
# adb shell su -- getevent -lp

add device 1: /dev/input/event2 <------ Power Button
  name:     ``bcmpmu_on''
  events:
    KEY (0001): KEY_POWER
  input props:
    <none>

add device 2: /dev/input/event3
  name:     ``bcm_headset''
  events:
    KEY (0001): KEY_VOLUMEDOWN        KEY_VOLUMEUP          KEY_MEDIA
  input props:
    <none>

add device 3: /dev/input/event1  <----- tapping, gesture, etc.
  name:     ``synaptics_dsx''
  events:
    KEY (0001): KEY_POWER             KEY_SLEEP             BTN_TOOL_FINGER       BTN_TOUCH
    ABS (0003): ABS_X                 : value 0, min 0, max 320, fuzz 0, flat 0, resolution 0
                ABS_Y                 : value 0, min 0, max 320, fuzz 0, flat 0, resolution 0
                ABS_MT_SLOT           : value 0, min 0, max 4, fuzz 0, flat 0, resolution 0
                ABS_MT_TOUCH_MAJOR    : value 0, min 0, max 15, fuzz 0, flat 0, resolution 0
                ABS_MT_TOUCH_MINOR    : value 0, min 0, max 15, fuzz 0, flat 0, resolution 0
                ABS_MT_POSITION_X     : value 0, min 0, max 320, fuzz 0, flat 0, resolution 0
                ABS_MT_POSITION_Y     : value 0, min 0, max 320, fuzz 0, flat 0, resolution 0
                ABS_MT_TRACKING_ID    : value 0, min 0, max 65535, fuzz 0, flat 0, resolution 0
  input props:
    INPUT_PROP_DIRECT

add device 4: /dev/input/event0
  name:     ``alp''
  events:
    ABS (0003): ABS_MISC              : value 0, min 0, max 65528, fuzz 0, flat 0, resolution 0
  input props:
    <none>

could not get driver version for /dev/input/mouse0, Not a typewriter
could not get driver version for /dev/input/mice, Not a typewriter
```

Due to the nature of available Android smart watches and the fact that most current Android Wear products mainly support checking (Android's) notifications, I take any manipulation on the watch screen as an indication that the user is at a breakpoint. Thus, the current breakpoint detector implementation looks for breakpoints every 2.5 seconds, if more than one tap-related event comes from the underlying Linux Input Subsystem in this time window.

8.2.2 Physical Activity-based Breakpoint Detection

Physical Activity-based breakpoint detection is based on **a transition** in a user's physical activity, such as when she stops walking. This breakpoint detector relies on underlying activity recognition that generates labels such as walking, running, or still, and detects breakpoints according to changes in activity.

To confirm my earlier hypothesis about detecting breakpoints in during changes in physical activities (i.e., that breakpoints exist when moving from a high-energy to a low-energy activity), I conducted a survey. The survey asked participants to rate, using a 10-point Likert scale, the likelihood of a breakpoint when transitioning between each pair of the following physical activities: bike-ride, running, walking, working at a desk, and being still. Table 8.2

shows the summary of the results from 26 university students.

Based on these results, I built a model for detecting physical activity-based breakpoints, that reads a series of user activity labels from the underlying activity recognition system and concludes a breakpoint if an activity change whose corresponding value in Table 8.2 is greater than 5.0.

Table 8.2: Ground Truth on Physical Activity Change Breakpoint

		To				
		onbike	running	walking	working	still
From	onbike		4.7 (3.1)	**6.8** (2.5)	4.9 (3.4)	**6.4** (3.0)
	running	4.7 (3.0)		**8.2** (1.4)	4.5 (3.3)	**7.0** (2.6)
	walking	4.3 (3.0)	5.0 (2.9)		**5.3** (3.3)	**7.4** (2.3)
	working	4.8 (3.5)	**5.4** (3.1)	**6.9** (2.6)		**5.8** (3.6)
	still	4.7 (3.3)	**5.1** (3.1)	**7.3** (2.3)	3.8 (2.9)	

Each number shows the average (and standard deviation) using a 10-point Likert scale.
Values in bold indicate those used by the breakpoint detector.

Generic Android Platform

On the generic Android platform, physical activity-based breakpoint detection is built on top of the Activity Recognition API of Google Play Service Location APIs [30]. Using various types of physical sensors and the GPS in the Android devices, this API returns the device's current activity, such as "STILL", "IN_VEHICLE", "RUNNING", or "ON_BICYCLE". Detail of this API by Google is not opened for the public, but it is considered to be using multiple types of sensors, such as accelerometer and GPS, inside.

The frequency with which the API returns the activity depends on the Android platform version. On Android 4, it returns relatively periodically, such as once every few seconds. On the other hand, the frequency is very variable on Android 5 platform, depending on the current activity. When the phone is placed and fixed on the desk, for example, the intervals of the activity updates from the API will be several minutes probably because the API detects the phone is not on user's body.

Since the API is provided by Google, the provider of the Android platform itself, I assumed the API has a certain degree of quality as a product and optimization in implementation and introduced it to the Attelia system. On the generic Android platform devices, such as smartphones and tablets, Attelia II opens an instance of this API and uses a stream of activity labels output from the API. The activity labels will be fed into my original breakpoint classifier which refers the values in Table 8.2.

Android Wear Platform

On the Android Wear 5 platform, I implemented my own activity recognition on the smart watch since the Activity Recognition API described above is not supported. Following my own past research on smartphone-based activity recognition [66], I built an accelerometer-based activity recognizer as follows. The system's overview with real-time mobile sensing and machine learning technique is similar to what has been already presented in Section 6.2.

Sensor hardware and data: My implementation uses the Sony SmartWatch3 SWR50 [79] with Android Wear 5.0.1. Following my previous work and other activity recognition research, my system uses data from the accelerometer. On SWR50, applications can read acceleration data with a frequency of 50Hz. Using both High Pass Filter (HPF) and Low Pass Filter (LPF), value of gravity force and very high frequency wave in raw acceleration values will be filtered out, before further data processing.

Feature vector: Table 8.3 overviews the features used in the system. The system uses a set of 22 commonly used time-domain and frequency features, widely used for accelerometer-based activity detection. Based on my past experience, the length of a time frame for feature extraction is set to 3 seconds.

Table 8.3: Selected Features Used for Activity Recognition

Feature Type	Features
Time Domain	Mean ($\bar{x}, \bar{y}, \bar{z}$) Magnitude of Mean ($\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}$) Variance $\{var(x), var(y), var(z)\}$ Correlation $\{corr(x,y), corr(y,z), corr(x,z)\}$ Covariance $\{cov(x,y), cov(y,z), cov(x,z)\}$
Frequency Domain	Energy ($\frac{\sum_{j=1}^{N}(m_j^2)}{N}$), m_j is FFT component Entropy ($-\sum_{j=1}^{n}(p_j * log(p_j))$, p_j is FFT histogram

Training of the classifier model: Due to the reduced capability of Android Wear devices, my current implementation only classifies "still", "walking", and "running". I trained a J48 decision tree classifier model on the Weka [56] engine, with ground truth data collected from 10 people. The cross validation result is shown in Table 8.4.

Table 8.4: Confusion Matrix: Cross Validation of Activity Recognition

		Classified As		
		still	walking	running
Ground truth	still	92.2%	7.7%	0.0 %
	walking	4.3%	95.1%	0.6 %
	running	5.1%	5.3%	89.8%

8.2.3 Inter-Device Communication

Attelia II shares the breakpoint detection events and device-usage status across multiple devices via Bluetooth-based Personal Area Network (PAN) among the devices. Here Attelia II basically assumes that the user is wearing and carrying devices that are within the range of Bluetooth wireless communication.

When a local breakpoint detector detects a breakpoint, attributes about the breakpoint, including its timestamp and detector type, will be sent to other devices in real-time. Attelia service on each device keeps track of the list of currently-detected breakpoints (sent from all devices) for the last several t seconds. (Currently the value of t is set to 10 seconds.) As the result such communication between devices, Attelia II has an assumption that the Attelia II service on user's each device basically share the same view on the current conditions of the breakpoint detections in all devices.

Also for device-usage status, Attelia II sends a "DEVICE IN_USE" message to remote devices when the screen of the local device turns on, and a "DEVICE NOT_USED" message when the screen turns off. Again, as the result, Attelia II services on all of user's devices basically share the same view on the current device-usage conditions of all devices. This simple implementation covers most situations in which a user is manipulating target devices, since most of the time the screen is on when the user is interacting with the a device.

8.2.4 Combining Breakpoint Detection

In combinational breakpoint detector component, a final judgment on the user's breakpoint detection will be processed based on (1) current condition of all underlying breakpoint detections and (2) pre-configured combinational detection model.

Figure 8.5 shows an example situation on one of the user's devices. A table on the left stores the shared view on the current conditions of the breakpoint detections in all devices. In this example, currently physical-activity breakpoint is detected both on the watch and the smartphone. Another table on the right illustrates the concrete example of the combinational detection model. In this particular example, the model specifies that physical-activity breakpoint on the watch and physical-activity breakpoint on the phone are needed to declare the final conclusive breakpoint.

Every time a breakpoint detection event comes from any user interaction-based or physical activity-based breakpoint detector (of any device, either from the local device or the remote device), the new event will be firstly stored into the left table as a newest data. As already mentioned, any breakpoint detected by an individual detector within the last 10 seconds is considered to be a "current" breakpoint. Next, combinational breakpoint detector component makes the final decision on the "conclusive breakpoint", by comparing the both tables and checking if the current condition satisfies what the model requires.

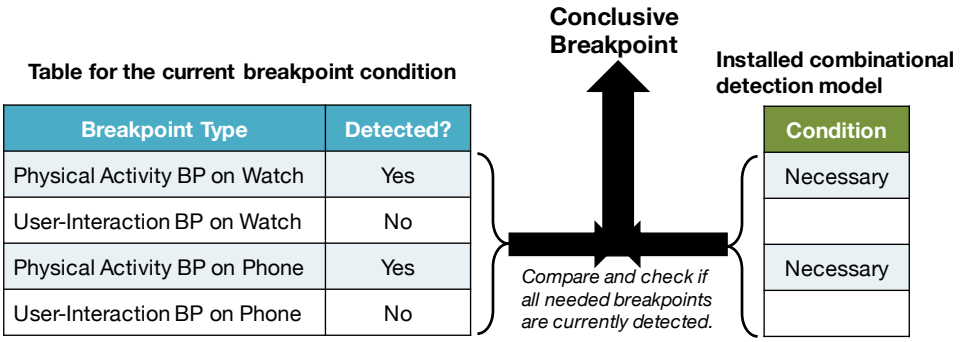


Figure 8.5: Combinational Breakpoint Detection on Each Device

8.3 Evaluation: In-the-Wild User Study

Using the Attelia II prototype, I conducted an in-the-wild user study with 41 participants for 1 month to evaluate how Attelia II performs in users' multi-device environments.

The objectives in this study were as follows:

1. The Attelia I system used only User Interaction-based breakpoint detection on smartphones. I investigated whether the addition of Physical Activity-based breakpoint detection on smartphones would result in reduced workload perception when dealing with notifications.

2. I wanted to understand the value of having breakpoint detection on a worn smart device. So, on the smart watch alone, I compared the impact of performing breakpoint detection for delivering notifications compared to random delivery timings.

3. As there are different possible ways to combine the different detectors for making a final decision about whether a breakpoint has occurred, I compared different combinations of watch and phone breakpoint detectors to each other, to random delivery and to Attelia I.

8.3.1 Participants

41 (31 male and 10 female) participants were recruited for the study. The participants were university students and staff members, with ages between 19 and 26. 24 participants came from computer science and information technology related departments, while the other 17 came from other schools, such as economics, psychology, or social sciences. All of the participants were smartphone (Android OS version 4.3 or above) users in their daily lives. None had a smart watch, so I provided each with a Sony SmartWatch3 device to use during the study. Subjects were paid $100 for their participation, and were eligible to win 1 of 2 smart watches via a lottery.

8.3.2 Overview of the Experiment Procedure

My experimental procedure consisted of three parts. (1) At the beginning of the study, each participant received instructions for the study, signed my consent form for participation, and received a Sony SmartWatch3. I paired the watch to the participant's phone and installed Attelia II on both devices.

(2) Starting from the next morning, the data collection and breakpoint detection began and lasted for 31 days. Every day, each user experienced Attelia's interruptive notifications, whose timings are based on a randomly selected breakpoint detection model from those models shown in Table 8.5. Attelia II contained definitions of all these models, but these were hidden inside Attelia, thus users did not know which model they were being exposed to each day. Everyday, both devices were set to the same chosen model.

To explore my third objective, I split the 31-day experiment into 2 phases. Table 8.6 shows the number of days that each model was configured to be selected during each phase. During the first phase, a special "comparison" mode, described later, was configured to collect data efficiently on multiple different combinations of models. During the 2nd phase, the model was changed randomly, but evenly, among the specified 5 models everyday, to prevent ordering effect.

(3) After 31 days, participants filled out the post-experiment survey, uninstalled the Attelia service, returned the watch (except for the lottery winners), and were paid.

8.3.3 Experimental Setup

Combinational Breakpoint Detection Model

In order to achieve the objectives described above, I created a series of combinational breakpoint detection models as shown in Table 8.5. Each strategy has a different set of underlying detectors to be used for the combinational detection.

The **Random** model does not use any detectors and displays notifications using random timings. This model emulates what people are currently interrupted by notifications.

Phone_UI and **Phone_UI_Act** are prepared for the first objective. Phone_UI is actually the Attelia I system, which uses only a UI-based detector on the phone. This model delivers notifications at the breakpoint timings detected by the UI-based detector while the device is manipulated (the screen is on), and shows notifications in the random timing while the device is not used (the screen is off). On the other hand, Phone_UI_Act adds the use of the physical activity-based detector. The difference between the two models is that Phone_UI_Act delivers notifications at the breakpoint timings detected by activity-based detector while the device is not used, instead of the random timings.

Watch_UI_Act, along with **Random**, describes the conditions for the second objective. Watch_UI_Act delivers notifications at the breakpoint timings detected by the UI-based detector while the watch is manipulated (the screen is on), and delivers notifications at the breakpoint timings detected by the watch's activity-based detector while the watch is not used (the screen is off).

Table 8.5: Combination Breakpoint Detection Models

Model Name	Detectors used for combinational breakpoint detection			
	Watch		Phone	
	UI-based	Activity-based	UI-based	Activity-based
Random	None of detectors used. In random timings.			
Phone_UI	Not used	Not used	Used	Not used
Phone_UI_Act	Not used	Not used	Used	Used
Watch_UI_Act	Used	Used	Not used	Not used
Combo(c)	Used	Not used	Used	Not used
Combo(d)	Used	Not used	Not used	Used
Combo(e)	Not used	Used	Used	Not used
Combo(f)	Not used	Used	Not used	Used
Combo(g)	Not used	Used	Used	Used
Combo(h)	Used	Not used	Used	Used
Combo(i)	Used	Used	Not used	Used
Combo(j)	Used	Used	Used	Not used
Combo(k)	Used	Used	Used	Used
Combo(x)	OR(Combo(h), (g), (f), (j), (d))			

Table 8.6: Phase, Used Model and Duration during the 31 Day User Study

Phase	Phase 1 (14days)	Phase 2 (17 days)				
Model	*(special "comparison" mode)*	Random	Phone_UI	Phone_UI_Act	Watch_UI_Act	Combo(x)
Duration (days)	14	3	3	3	3	5

Combo(c) through **Combo(x)** are the models which involve multiple detectors across devices and were compared for the third objective. These "Combo" models internally use "AND" logic over multiple underlying detectors to make their final breakpoint decision. I used these models to explore whether multiple agreements amongst individual breakpoint detectors might perform better than the individual ones.

Interruptive Notifications

The interruptive notification took the form of a full-screen Experience Sampling Method (ESM) question that asked users to indicate whether the current moment was a good time to be interrupted, using a 5-point Likert scale (1=strongly disagree to 5=strongly agree). This custom notification was employed due to the limitation on Android OS where third party software cannot control timings of Android's official notification system. If the user was actively manipulating the smartphone, then the notification was delivered on the phone.

Otherwise, I defaulted to delivery on the watch. All the notifications were treated equally without concept of "importance".

The minimum interval between two consecutive notifications was set to 1500 seconds, the maximum interval to 1800 seconds, and the daily maximum number of notifications to 20. The study software also was configured to only send notifications between 8AM to 8PM daily. The parameter values were carefully chosen after interviewing prospective participants about their daily lives, to acquire a sufficient number of data samples without overburdening them.

Measurement

When the user's model detected a breakpoint, an Experience Sampling Method (ESM) notification was delivered to the user. If the user was actively manipulating the smartphone, then the notification was delivered on the phone. Otherwise, I defaulted to delivery on the watch. The notification asked users to indicate whether the current moment was a good time to be interrupted, using a 5-point Likert scale (1=strongly disagree to 5=strongly agree). In addition, each night, users were given the NASA-TLX [33] survey, a validated instrument for assessing user workload. They were asked to answer the survey on the Web, and to consider their experience with the current day's notification delivery strategy provided by Attelia II.

8.3.4 Collected Data

Analyzing all the collected data uploaded to my server during 31 days, the average daily duration of device operation, during the times when each of two UI-based breakpoint detectors were active, was 227 minutes on the phone and 1.4 minutes on the watch. When I group operations that are separated by less than 60 seconds, the per-user daily average number of device operations are 174 on the phone, and 9.5 on the watch. Also, the average number of displayed notifications for each user was 10.5 times per day, with 7.3 notifications of these being attended to by the user.

8.3.5 Result: Value of Physical Activity-based Breakpoint Detection

My first experiment was to investigate whether the addition of physical activity-based breakpoint detection to the already existing UI-based detection on smartphones, would reduced user's workload perception when dealing with notifications. We evaluated these two approaches (Phone_UI and Phone_UI_Act) and "Random" for each user, over a period of 9 days, and compared the resulting workloads.

Figure 8.6 shows the average TLX Weighted Workload (WWL) scores among the models. The Phone_UI_Act model results in significantly lower workload perception, compared to the Phone_UI (Attelia I) model and Random, which approximates how people are currently interrupted by notifications. When compared to the baseline (Random), Phone_UI_Act

had a lower score by 12.2 (i.e., reduced workload), while Phone_UI reduced the workload score only by 7.1. The relative gain (or reduction in workload perception) from using the Phone_UI_Act model is 71.8%, compared to the Phone_UI (Attelia I) model.

A Friedman test revealed a significant effect of notification strategy on the WWL score ($\chi^2(4) = 18.5, p < 0.01$). A post-hoc pair-wise comparison using Wilcoxon rank sum tests showed the significant differences between Random and Phone_UI ($p < 0.05$, $\gamma = 0.28$), between Random and Phone_UI_Act ($p < 0.05$, $\gamma = 0.50$), and between Phone_UI and Phone_UI_Act ($p = 0.05$, $\gamma = 0.24$). Therefore, I can confirm my first hypothesis that adding physical activity breakpoint detection to smartphones is an improvement over just having UI-based breakpoint detection.

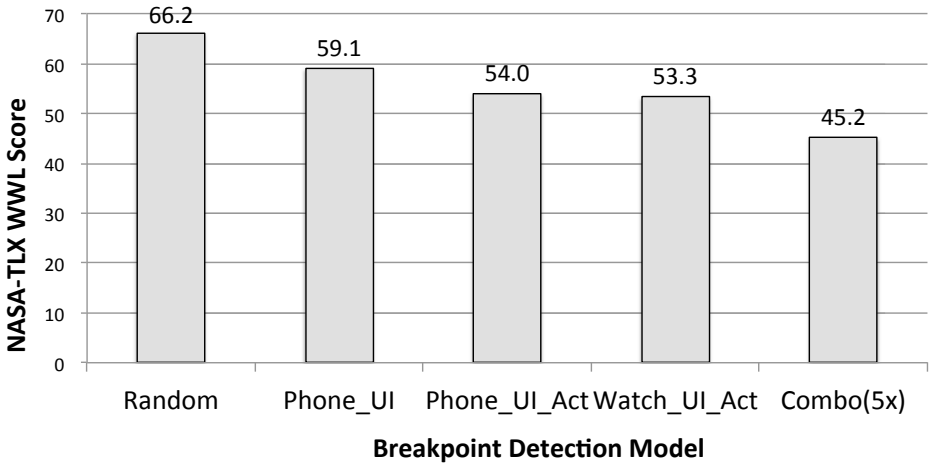


Figure 8.6: NASA-TLX WWL Scores

Figure 8.7 shows the average ESM interruptibility (5-point Likert scale) scores among the models. The Phone_UI_Act model was rated as providing more appropriate interruptions, when compared to both the Phone_UI (Attelia I) model and Random.

8.3.6 Result: Attelia II on the Smart Watch

My next experiment was to investigate whether Attelia II on the watch would reduce user's workload perception. Taking a similar approach as for the previous experiment, I evaluated the Watch_UI_Act model for each user, over a period of 3 days and compared the resulting workload to that of the Random model from the previous experiment.

In Figure 8.6 the Watch_UI_Act model results in significantly lower workload perception, compared to the Random model (a reduction in workload score of 12.8, or 19.4%) A pair-wise comparison using Wilcoxon rank sum tests showed the significant differences between Random and Watch_UI_Act ($p < 0.05$, $\gamma = 0.35$).

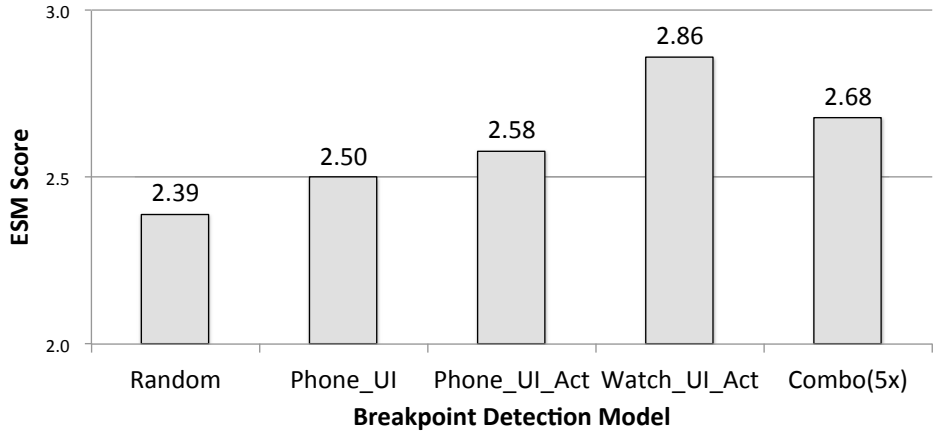


Figure 8.7: ESM Scores

8.3.7 Result: Inter-Device Combinational Models

The final experiment was to investigate the power of inter-device combinational breakpoint detection. Since the number of combinational models ("Combo" models in Table 8.5) are large, I split the experiment into two phases.

Phase 1: Choosing the Best Model

For the first phase of the study, which lasted for 14 days, my goal was to evaluate the accuracy of the different combinational models: Combo(c) through Combo(k). To do so, we configured Attelia into a special "comparison" mode. We set all four breakpoint detectors to be active (phone UI, phone physical activity, watch UI, watch physical activity). Whenever any of the detectors detected a breakpoint, an ESM-based notification was delivered to the user. The notification asked users to indicate whether it was delivered at an interruptible moment, using a 5 point Likert-scale. In addition, we examined the state of the other three breakpoint detectors. With the state of all four breakpoint detectors, and the ESM value, we could assess the value of all 9 combinational models. For example, consider a situation when a user is interacting with her smartphone and the smartphone's UI-based breakpoint detector is triggered. In addition, her watch's UI-based breakpoint detector was not triggered because she was not manipulating it, but both the watch's and the phone's physical activity-based detectors were triggered because she transitioned from walking to being still. This combination of detectors corresponds to the Combo(g) model. Using the gathered ESM response, we can assess whether this combination accurately detected a breakpoint.

Note that we capped the daily number of breakpoints to 20. My goal was to acquire 2 ESM responses for each of the 9 combinational models and the Random model each day. To ensure that we achieved this goal, if the Combinational Detector finds a model that can be evaluated at this moment (e.g., Combo(h)), and if the model has not already had its two ESM responses, only then will the ESM notification will be displayed. Otherwise, no

notification is delivered. To collect 2 responses for the Random model, "Random"-based notifications were randomly triggered twice daily.

Table 8.7 summarizes the results. For each user and for each Combo model, we calculated the average ESM score (5-point Likert scale) and compared it to the average ESM score for the Random model. We define the difference between these two scores as the "gain" value. We then looked at which model provided the biggest gain for each user. Row (1) shows, for each model (Combo(c) through (j)), the gain averaged across the users for whom that model had the highest gain. We can clearly see that, for the models that use more of the underlying detectors, (i.e., (g), (h), (i), and (j)), the gain is higher. Note that Combo(k) notification, which reflects the situation where all of 4 detectors detecting breakpoints, did not occur during our study. However, as Row (2) shows, we also observed that the number of answered ESMs (and correspondingly, the number of delivered ESMs - Row 2 - is quite small for those models. This is expected as it is less likely that 3 or more detectors will be active at any given time.

With these results, we chose the best 5 models (those with the highest average gain), and combined them with a disjunction (i.e., a logical OR). We label this new model as **Combo(x)** and we use this in phase 2 of my user study where we compare it to the non-combinational models described earlier.

Table 8.7: ESM Score Results on "Combo" Models

Model	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	Random
(1)Average gain across highest-gainers	1.28	**1.46**	1.17	**1.45**	**1.62**	**1.72**	1.40	**1.47**	(N/A)	(1.00)
(2)Number of displayed ESM	300	46	194	170	68	8	6	72	0	1043
(3)Number of answered ESM	249	34	165	137	54	7	6	58	0	647

Phase 2: Power of the Best Model

After the first phase ended and we successfully created the model Combo(x), we began phase 2 which lasted for 17 days. We evaluated this model with my other models: Random, Phone_UI, Phone_UI_Act, and Watch_UI_Act. Note that experiments 1 and 2 were conducted during phase 2, and it is those results that we compare to Combo(x). All users experienced all models for at least 3 days, in a random ordering.

Again, Figure 8.6 shows the comparisons of average TLX WWL scores across the models. The Combo(x) model has a significantly lower workload than the other models, including Phone_UI_Act and Watch_UI_Act. Combo(x) amazingly resulted in an 295.1% more reduction in workload perception, compared to Phone_UI (Attelia I). When compared to the baseline (Random), Combo(x) resulted in a score that was 21.0 points lower, while Phone_UI resulted only in a reduction of 7.1. A post-hoc pair-wise comparison using Wilcoxon rank sum tests showed the significant differences between "Random" and "Combo(x)" ($p < 0.01$, $\gamma = 0.71$), between "Phone_UI" and "Combo(x)" ($p < 0.01$, $\gamma = 0.51$), and between "Phone_UI_Act" and "Combo(x)" ($p = 0.05$, $\gamma = 0.25$).

8.3.8 Discussion: ESM Scores

Figure 8.7 shows the results of the average ESM scores for each breakpoint detection model.

The average scores across the models show promising values in terms of users' self-reported interruptibility. However, there were no statistical significance between the different models. I hypothesize that the responses for the Watch_UI_Act model has a higher (not significant) mean is that participants were interacting with their smart watch (which mostly just supports application notifications), and at that times, they were interruptible. This happened relatively infrequently as the watch was used relatively infrequently, thus not impacting the ESM score for Combo(x). Despite the lack of significance in these results, the results from the NASA-TLX survey clearly shows that users' workload perception was reduced by each of the different approaches in Attelia II. From these results I argue that, although users sometimes might not be aware of the direct value of Attelia II's breakpoint timing-based notification delivery, their overall workload perception was surely affected by my system.

8.4 Summary

This chapter described the research on detecting breakpoints in "multi-device" environment with covering user's comprehensive daily life. Attelia II extended breakpoint detection of Attelia I to cover all aspects of a user's daily life, including the period the devices are carried or worn but not actively manipulated, by introducing a combination of two types of breakpoints, namely **User Interaction-based Breakpoint** and **Physical Activity-based Breakpoint**. Attelia II also addressed how breakpoint detection can be applied in the **multi-device**.

My in-the-wild user study demonstrated the value of Attelia II in a multi-device environment. I showed that an introduction of physical activity-based breakpoint detection in addition to Attelia I's UI event-based breakpoint detection resulted in a 71.8% greater reduction in users' workload perception. Using both of these detectors on the smart watch resulted in a reduction of 19.4% in user workload perception, compared to random notification delivery. Finally, my best multi-device combinational breakpoint detection model "Combo(x)" resulted in a 31.7% lower workload perception compared to the random case, which is a greater reduction than from Attelia I by 295%.

Chapter 9

Conclusion

This dissertation addressed the problem of interruption overload, a situation in which the user's attention resource with limited capacity will be negatively impacted by an excessive amount of ill-timed interruptive notifications. Because users are surrounded by an increasing number of computer devices, applications, services, and other connected users all-day long in the age of ubiquitous computing, users are receiving more interruptions from applications in the background of their primary task. Meanwhile, human attention is well known to have a limited amount of capacity according to past research in the field of cognitive psychology. Under such a situation, the attention resource of human users is one of the most precious resources in the computing area.

Toward the capability of attention-awareness in computer systems, this research addressed the most fundamental part of attention-awareness, which is attention sensing. Employing the concept of breakpoint, my research system Attelia was developed to detect such timing in real-time, solely on a combination of mobile and wearable devices, without needing external psycho-physiological sensors, and without modifications to existing mobile operating systems or diverse applications.

The extensive user study on Attelia I and II validated the effectiveness of Attelia. Notifications delivered to the participants in the detected breakpoint timing significantly lowered their subjective workload perception. Furthermore, a combination of two types of breakpoint detection, namely, user interaction-based detection and physical activity-based detection, resulted in even more improvements in the reduction of workload perception.

9.1 Future Work

This section presents the avenues for future work on Attelia, and further research into attention-awareness.

Future Work on Attelia

First, further investigation on "insensitive" users ("WWL-insensitive users" and "FRU-insensitive users") is my first research topic of interest. Such an opportunity includes the possible real-time detection of which cluster the user belongs to, and an investigation into other possible notification adaptation schemes so that their workload perception can be lowered.

A second research opportunity is further system improvement using the model personalization technique. In this paper, I used a single model (decision tree), which was commonly used for all participants. For example, active learning and a longitudinal user study are possible next steps.

Classifying multiple levels of breakpoints is another opportunity to improve the breakpoint detection capability of Attelia. Iqbal *et al.* found that there are at least three granularities of breakpoints that users detect reliably, "coarse," "medium," and "fine" [44]. The current Attelia implementation classifies a breakpoint in a binary manner, without considering the granularity. Multi-level breakpoint detection, possibly using the information from the source detector, such as the concrete breakpoint type, points to an interesting future research opportunity.

Deployment of the Attelia service to a real notification system on a smartphone operating system, including Android OS, is yet another challenging task. My user study with the current implementation used my own artificial interruptive notifications due to the access limitations of the real notifications of the real applications inside Android OS. Meanwhile, Attelia is ready to export its "interruptibility API" based on the results of a real-time breakpoint detection. Other applications can utilize Attelia's API through a standard Android IPC mechanism.

Towards the realization of a real-world attention-aware notification system, introducing a concept of "priority" both on user's demand side and on information provider's supply side for the "brokering" between them can be another next step of the research. Also, a concept of "topic relevance" between user's pursuing task and interruptive notifications can be another elements for such brokering. For such functionalities, sensing of "user's attentional target" from user's context information and/or application's content information, as well as the mechanisms for priority handling and negotiation between the demand and supply sides, are the immediately building blocks to be explored. In the current Attelia system, all notifications are handled with equal priority, and the relevance of the content between the user's primary task and the interruptive notification has not yet been handled.

Next, a more detailed analysis on the user's interruptibility with regard to changes in physical activity serves as an interesting research opportunity, given that smartphone platforms come with activity recognition APIs. Initially, I had a hypothesis that people will have a breakpoint when changing from a "high-energy" state to a "low-energy" state, such as when they "stop walking." However, our self-reported data in Table 8.2 shows that there are breakpoints when going from low states to high states, such as being "still" to "walking."

Further introduction and opportunistic combinations of other wearable devices, such as

smart glasses [52] with gaze-tracking or blink-recognition features, is yet another research opportunity. Under the use of multiple devices, in which some of the devices have their own dedicated sensors and some do not, I need to investigate the value in having subsections of my system focus on the more impoverished devices.

Determining which device to send notifications to is another research challenge. In my user study, for simplicity, I configured the notification destination such that, if the user is actively manipulating their smartphone, then a notification is delivered on the phone. Otherwise, I defaulted to delivery on their smart watch. I may apply other display techniques developed by related studies in a multi-display context [22, 27].

Currently, our "workload perception" measurement using a nightly NASA-TLX survey has a limitation in terms of the temporal distances between the actual interruptive notification and the survey. Although we asked the participants to review their notification experience during the day in terms of the timing (not in terms of the number of notifications), their subjective evaluation at the end of the day may be influenced by several non-experiment-related aspects of their lives. Meanwhile, conducting such a survey during the day can itself be another possible workload for users. A lightweight but efficient survey methodology should be investigated as future work.

Although my extensive in-the-wild user studies, which lasted for up to one month (respectively), proved the effectiveness of my proposal, further long-term and large-scale evaluations with diverse subjects remain as future work. In my user studies, the participants were recruited from university communities. For example, using a popular crowdsourcing service such as Amazon Mechanical Turk, a further larger-scale evaluation with people of versatile demographics will be enabled. For the longer-term evaluation, several types of techniques to keep participants' active commitment for the experiment, including gamification techniques, are considered to be a useful supporting mechanism.

Longer Term Research

In the longer term, studies on attention-awareness have only begun with this research on attention sensing. Various attention-aware adaptations in computing other than the adaptive delivery of interruptive notifications have yet to be addressed. Other functionalities in attention-awareness, such as attention prediction or attention management, are also areas of future work.

Considering an even longer period of a user's internal state, attention can be classified as a rather short-term state. For example, what a user "wants to do" in the longer term is considered to be their **intention.** In the same context of research on awareness in computing toward a user's internal states, "intention-awareness" is yet another big research challenge, starting from the formal conceptualization and establishment of relationships with the area of attention.

9.2 Epilogue

Ever since the preciousness of human attention resource in computing was first mentioned by Herbert A. Simon in 1969 [76] (published in 1971), attention-aware computing has been a research issue for nearly half a century. As emerging mobile and wearable devices and their numerous applications are taking gradual steps into the realm of ubiquitous computing in the real world, one ultimate aspect of ubiquitous computing with disappearing technology is making the use of a *"computer as refreshing as taking a walk in the woods"* [89]. I am confident that this dissertation will be a significant step toward the realization of such human-centered computing and that it will contribute to the field of computer science in the long term.

Bibliography

[1] A. R. Abdel-Khalik. The effect of aggregating accounting reports on the quality of the lending decision: An empirical investigation. *Journal of Accounting Research*, pages 104–138, 1973.

[2] P. D. Adamczyk and B. P. Bailey. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 271–278, New York, NY, USA, 2004. ACM.

[3] T. L. K. Archives. Programming input driver. https://www.kernel.org/doc/Documentation/input/input-programming.txt.

[4] B. P. Bailey and J. A. Konstan. 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):685 – 708, 2006.

[5] B. P. Bailey, J. A. Konstan, and J. V. Carlis. Measuring the effects of interruptions on task performance in the user interface. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 2, pages 757–762 vol.2, 2000.

[6] B. P. Bailey, J. A. Konstan, and J. V. Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *IFIP TC.13 Conference on Human Computer Interaction (Interact 2001).*, pages 593–601. IOS Press, 2001.

[7] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive computing*, pages 1–17. Springer, 2004.

[8] J. Beatty and B. Lucero-Wagoner. The pupillary system. *Handbook of psychophysiology*, 2:142–162, 2000.

[9] J. B. Begole, N. E. Matsakis, and J. C. Tang. Lilsys: Sensing unavailability. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 511–514, New York, NY, USA, 2004. ACM.

[10] BI Intelligence. Here comes the internet of things. https://intelligence.businessinsider.com/the-internet-of-things-2013-10, Aug. 2014.

[11] M. Böhmer, C. Lander, S. Gehring, D. P. Brumby, and A. Krüger. Interrupted by a phone call: Exploring designs for lowering the impact of call notifications for smartphone users. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3045–3054, New York, NY, USA, 2014. ACM.

[12] D. E. Broadbent. *Perception and communication*. Elsevier Science Ltd, 1958.

[13] 羽倉淳一郎, 鈴木邦夫, and 松尾健彦. 自動車の電動化と高性能化を支える車載用モータドライバ IC (特集 進化する車載用半導体技術). 東芝レビュー = *Toshiba review*, 69(8):12–15, Aug. 2014.

[14] E. G. Chewning and A. M. Harrell. The effect of information load on decision makers' cue utilization levels and decision quality in a financial distress decision task. *Accounting, Organizations and Society*, 15(6):527–542, 1990.

[15] S. Cohen. Aftereffects of stress on human performance and social behavior: a review of research and theory. *Psychological bulletin*, 88(1):82, 1980.

[16] F. J. Corbató, M. Merwin-Daggett, and R. C. Daley. An experimental time-sharing system. In *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, AIEE-IRE '62 (Spring), pages 335–344, New York, NY, USA, 1962. ACM.

[17] M. Csikszentmihalyi and R. Larson. Validity and reliability of the experience-sampling method. *The Journal of nervous and mental disease*, 175(9):526–536, 1987.

[18] M. Czerwinski, E. Cutrell, and E. Horvitz. Instant messaging: Effects of relevance and timing. In *People and computers XIV: Proceedings of HCI*, volume 2, pages 71–76. British Computer Society, 2000.

[19] M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 175–182, New York, NY, USA, 2004. ACM.

[20] J. A. Deutsch and D. Deutsch. Attention: some theoretical considerations. *Psychological review*, 70(1):80, 1963.

[21] A. K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, Jan. 2001.

[22] J. Dostal, P. O. Kristensson, and A. Quigley. Subtle gaze-dependent techniques for visualising display changes in multi-display environments. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 137–148, New York, NY, USA, 2013. ACM.

[23] J. E. Fischer, C. Greenhalgh, and S. Benford. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 181–190, New York, NY, USA, 2011. ACM.

[24] Fitbit Inc. Fitbit trackers. http://www.fitbit.com/, 2014.

[25] Flurry. App install addiction shows no signs of stopping. http://flurrymobile.tumblr.com/post/115194583975/app-install-addiction-shows-no -signs-of-stopping, Dec. 2014.

[26] T. K. Fredericks, S. D. Choi, J. Hart, S. E. Butt, and A. Mital. An investigation of myocardial aerobic capacity as a measure of both physical and cognitive workloads. *International Journal of Industrial Ergonomics*, 35(12):1097–1107, 2005.

[27] J. E. Garrido, V. M. R. Penichet, M. D. Lozano, A. Quigley, and P. O. Kristensson. Awtoolkit: Attention-aware user interface widgets. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 9–16, New York, NY, USA, 2014. ACM.

[28] Google Inc. Designing for accessibility - Android Developers. https://developer.android.com/intl/ja/guide/topics/ui/accessibility/index.html.

[29] Google Inc. Google alert. https://www.google.com/alerts.

[30] Google Inc. Making your app location-aware - Android Developers. https://developer.android.com/intl/ja/training/location/index.html.

[31] Google Inc. The new multi-screen world | think with google. http://www.google.com/think/research-studies/the-new-multi-screen-world-study.html, Aug. 2012.

[32] E. Haapalainen, S. Kim, J. F. Forlizzi, and A. K. Dey. Psycho-physiological measures for assessing cognitive load. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 301–310, New York, NY, USA, 2010. ACM.

[33] S. G. Hart and L. E. Staveland. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139 – 183. North-Holland, 1988.

[34] J. Ho and S. S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 909–918, New York, NY, USA, 2005. ACM.

[35] E. Horvitz and J. Apacible. Learning and reasoning about interruption. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 20–27, New York, NY, USA, 2003. ACM.

[36] E. Horvitz, P. Koch, and J. Apacible. Busybody: Creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 507–510, New York, NY, USA, 2004. ACM.

[37] M. Hoshiba, O. Kamigaichi, M. Saito, S. Tsukada, and N. Hamada. Earthquake early warning starts nationwide in japan. *Eos, Transactions American Geophysical Union*, 89(8):73–74, 2008.

[38] S. Hudson, J. Fogarty, C. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J. Lee, and J. Yang. Predicting human interruptibility with sensors: A wizard of oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 257–264, New York, NY, USA, 2003. ACM.

[39] C. S. Ikehara and M. E. Crosby. Assessing cognitive load with physiological sensors. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 295a–295a. IEEE, 2005.

[40] Information Processing Society of Japan. Historical computers in japan: National aerospace laboratory of japan, numerical wind tunnel. http://museum.ipsj.or.jp/en/computer/super/0020.html.

[41] S. T. Iqbal, P. D. Adamczyk, X. S. Zheng, and B. P. Bailey. Towards an index of opportunity: understanding changes in mental workload during task execution. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 311–320. ACM, 2005.

[42] S. T. Iqbal and B. Bailey. Leveraging characteristics of task structure to predict the cost of interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 741–750, New York, NY, USA, 2006. ACM.

[43] S. T. Iqbal and B. P. Bailey. Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1489–1492, New York, NY, USA, 2005. ACM.

[44] S. T. Iqbal and B. P. Bailey. Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 697–706, New York, NY, USA, 2007. ACM.

[45] S. T. Iqbal and B. P. Bailey. Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks. *ACM Transactions on Computer-Human Interaction*, 17(4):15:1–15:28, Dec. 2010.

[46] S. T. Iqbal and E. Horvitz. Notifications and awareness: A field study of alert usage and preferences. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pages 27–30, New York, NY, USA, 2010. ACM.

[47] S. Ishimaru, K. Kunze, K. Kise, J. Weppner, A. Dengel, P. Lukowicz, and A. Bulling. In the blink of an eye: Combining head motion and eye blink frequency for activity recognition with google glass. In *Proceedings of the 5th Augmented Human International Conference*, pages 15:1–15:4. ACM, 2014.

[48] J. Jacoby, D. E. Speller, and C. K. Berning. Brand choice behavior as a function of information load: Replication and extension. *Journal of consumer research*, pages 33–42, 1974.

[49] J. Jacoby, D. E. Speller, and C. A. Kohn. Brand choice behavior as a function of information load. *Journal of Marketing Research*, pages 63–69, 1974.

[50] W. James. *The Principles of Psychology (Volume 1 of 2)*. Digireads. com Publishing, 2004.

[51] Jawbone. Jawbone UP. http://www.jawbone.com/, 2014.

[52] JINS Co., ltd. JINS MEME, https://www.jins-jp.com/jinsmeme/en/. https://www.jins-jp.com/jinsmeme/en/, 2014.

[53] D. Kahneman. *Attention and effort*. Prentice-Hall, Inc., 1973.

[54] E. Kim, S. Helal, and D. Cook. Human activity recognition and pattern discovery. *Pervasive Computing, IEEE*, 9(1):48–53, 2010.

[55] J. G. Kreifeldt and M. E. McCarthy. Interruption as a test of the user-computer interface. In *JPL Proceeding of the 17 th Annual Conference on Manual Control*, pages 655–667, 1981.

[56] Machine Learning Group at the University of Waikato. Weka 3: Data mining software in java. http://www.cs.waikato.ac.nz/ml/weka/.

[57] N. K. Malhotra, A. K. Jain, and S. W. Lagakos. The information overload controversy: An alternative viewpoint. *The Journal of Marketing*, pages 27–37, 1982.

[58] J. T. Milord and R. P. Perry. A methodological study of overloadx. *The Journal of General Psychology*, 97(1):131–137, 1977.

[59] Monsoon Solutions Inc. Monsoon power moniter. http://www.msoon.com/LabEquipment/PowerMonitor/.

[60] L. Mulder. Measurement and analysis methods of heart rate and respiration for use in applied environments. *Biological psychology*, 34(2):205–236, 1992.

[61] H. Müller, M. Pielot, and R. de Oliveira. Towards ambient notifications. *Peripheral Interaction: Embedding HCI in Everyday Life*, page 21, 2013.

[62] NASA. NASA Task Load Index (TLX) v. 1.0 manual. http://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX.pdf, 1986.

[63] D. Newtson and G. Engquist. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology*, 12(5):436–450, 1976.

[64] NorthCube AB. Sleep cycle. http://www.sleepcycle.com/, 2014.

[65] B. O'Conaill and D. Frohlich. Timespace in the workplace: Dealing with interruptions. In *Conference Companion on Human Factors in Computing Systems*, CHI '95, pages 262–263, New York, NY, USA, 1995. ACM.

[66] T. Okoshi, Y. Lu, C. Vig, Y. Lee, R. K. Balan, and A. Misra. Queuevadis: Queuing analytics using smartphones. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, pages 214–225, New York, NY, USA, 2015. ACM.

[67] F. Paas, J. E. Tuovinen, H. Tabbers, and P. W. Van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational psychologist*, 38(1):63–71, 2003.

[68] V. Pejovic and M. Musolesi. InterruptMe : Designing Intelligent Prompting Mechanisms for Pervasive Applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 395–906, New York, NY, USA, 2014. ACM.

[69] Pew Research Center. Social media update 2014. http://www.pewinternet.org/2015/01/09/social-media-update-2014/, Jan. 2015.

[70] K. Ryu and R. Myung. Evaluation of mental workload with a combined measure based on physiological indices during a dual task of tracking and mental arithmetic. *International Journal of Industrial Ergonomics*, 35(11):991–1009, 2005.

[71] salesforce.com. The 2014 mobile behavior report. http://www.exacttarget.com/2014-mobile-behavior-report.

[72] Samsung Electronics Co., Ltd. Galaxy nexus. http://www.samsung.com/us/mobile/cell-phones/GT-I9250TSGGEN.

[73] Y. Shi, N. Ruiz, R. Taib, E. Choi, and F. Chen. Galvanic skin response (gsr) as an index of cognitive load. In *CHI'07 extended abstracts on Human factors in computing systems*, pages 2651–2656. ACM, 2007.

[74] M. D. Shields. Some effects on information load on search patterns used to analyze performance reports. *Accounting, Organizations and Society*, 5(4):429–442, 1980.

[75] R. M. Shiffrin and W. Schneider. Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory. *Psychological review*, 84(2):127, 1977.

[76] H. A. Simon. Designing organizations for an information-rich world. *Computers, communication, and the public interest*, 37:40–41, 1971.

[77] J. Smith and N. Dulay. Ringlearn: Long-term mitigation of disruptive smartphone interruptions. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 27–35, March 2014.

[78] D. Snowball. Some effects of accounting expertise and information load: An empirical study. *Accounting, Organizations and Society*, 5(3):323–338, 1980.

[79] Sony Mobile Communications Inc. SmartWatch 3 SWR50. http://www.sonymobile.com/global-en/products/smartwear/smartwatch-3-swr50/, 2014.

[80] C. Speier, J. S. Valacich, and I. Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30(2):337–360, 1999.

[81] R. J. Sternberg and K. Sternberg. *Cognitive Psychology*. Cengage Learning, 6 edition, 2012.

[82] N. M. Su and G. Mark. Communication chains and multitasking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 83–92, New York, NY, USA, 2008. ACM.

[83] G. H. H. ter Hofte. Xensible interruptions from your mobile phone. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 178–181, New York, NY, USA, 2007. ACM.

[84] A. Toffler. *Future shock*. Bantam, 1990.

[85] TOP500.org. 20 years / top #1 systems / numerical wind tunnel: National aerospace laboratory of japan | top500 supercomputer sites. http://www.top500.org/featured/systems/numerical-wind-tunnel-national-aerospace -laboratory-of-japan/.

[86] A. Treisman. Monitoring and storage of irrelevant messages in selective attention. *Journal of Verbal Learning and Verbal Behavior*, 3(6):449–459, 1964.

[87] Trendstream Limited. Multi-device owners - gwi trends q1 2015. https://app.globalwebindex.net/products/report/multi-device-owners-q1-2015, 2015.

[88] M. Weiser. Ubiquitous computing. http://www.ubiq.com/hypertext/weiser/ UbiHome.html, Mar. 1996.

[89] M. Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing Communication Review*, 3(3):3–11, July 1999.

[90] M. Weiser and J. S. Brown. The coming age of calm technology. In *Beyond calculation*, pages 75–85. Springer, 1997.

[91] G. F. Wilson. An analysis of mental workload in pilots during flight using multiple psychophysiological measures. *The International Journal of Aviation Psychology*, 12(1):3–18, 2002.

[92] Yahoo Aviate. How android users interact with their phones. http://yahooaviate.tumblr.com/image/95795838933, Aug. 2014.

[93] F. R. Zijlstra, R. A. Roe, A. B. Leonora, and I. Krediet. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology*, 72(2):163–185, 1999.