Efficient Context-aware Real-time Processing of Personal Data Streams

Zur Erlangung des akademischen Grades eines
Doktors *der Ingenieurwissenschaften*

(*Dr.-Ing.*)

von der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Dipl.-Inform. Yongchun Xu

Tag der mündlichen Prüfung:	................24.07.2014…...............
Referent:	...........Prof. Dr. Rudi Studer…...
Korreferent:	...........Prof. Dr. Thomas Sezter..
Karlsruhe	2014

To

Yibing & Caecilia

Abstract

During the past decade, we have witnessed a rapid expansion of mobile devices into people's life. This development has tremendously impacted and changed the way we live our life. Mobile devices are increasingly penetrating into people's lives not only as efficient and convenient tools for communication and entertainment, but also as indispensable companions to many people. Meanwhile, thanks to the exponential advancement in technology in general and in sensor technology in particular through cheaper embedded sensor devices, electronic sensors, especially the wearable physiological sensors, are becoming more affordable and readily available to be used in our daily life. On the other hand, people have become more conscious of improving life quality by living a healthy life style. As such, any technologies, which relate to helping people achieve a better life style, have gained great attention and popularity.

Taking advantage of the advancement in both mobile technology and sensor technology, mobile devices offer a unique niche in applying these technologies to people's daily life and provide a new possibility of helping ordinary people be more proactive in monitoring and maintaining good or excellent health status. Unfortunately, the shortcomings of processing of such a large amount of real-time personal data in many mainstream applications remain the same: real-time data is processed in isolation and statically, which also means that real-time data are not integrated with other data, especially individual user's data, and the processing method is not dynamically adapted to the changes of users' situations. This has resulted not only in poor performance but also in inaccurate or outdated misleading information generated from these applications.

In order to overcome the shortcomings of existing solutions and to achieve an efficient processing of real-time sensor data on mobile devices for various use cases, we propose a framework for the development of innovative mobile applications that are context-aware in processing of real-time personal data streams by taking into account the resource limitation on mobile devices. In this thesis we first review the related literature and existing approaches, and set up the requirements for mobile-based event processing system based on the analyses of mobile device limitations and real-time personal data processing. Then we present an innovative event-driven hybrid software architecture for data-intensive mobile applications, in order to extend processing capability by using an additional backend server. We develop context-aware monitoring models with the purpose of relating real-time personal data with personal context and domain knowledge. The approach of data collaboration enables collaborative personal data processing. We develop methods for resource-aware dynamic pattern distribution to achieve a more efficient pattern distribution regarding available resources of mobile devices. The semantic-based dynamic pattern management uses semantic technologies to manage patterns and event resources and to achieve real-time adaptation to event resource changes. We also present evaluation results regarding the system performance, limitations, and several use cases. Finally, we conclude by summarizing the results of the thesis and outlining our view of the future work.

Contents

List of Figures

List of Tables

List of Algorithms

PART I

Foundation

1 Overview and Motivation

In this chapter we provide an overview and motivation of this thesis. Firstly we describe the main motivation of this thesis including the analysis of the issues in mainstream mobile applications for real-time personal data processing. Based on this analysis we define the research questions, which aim to find out an innovative solution to real-time personal data processing. We then introduce our approach with listing the main contributions of this thesis. In the fourth section we describe how the thesis is organized. The last section lists all publications that are basis for this thesis.

1.1 Motivation

Since Mark Weiser's vision [Weis91] over two decades ago of how the world would change with the introduction of ubiquitous computing, there has been significant progress towards his vision. During the past decade, the growing success of the new generation mobile technologies leads to an expansion of mobile devices, including smartphones and tablets. We have witnessed skyrocketing increases in the sales and uses of smartphones and tablets. For example, in the third quarter of 2013, 211.6 million android devices were sold, representing a growth of 51.3% compared to the same quarter over a year ago [IDC13a]. These numbers reflect a trend of ever increases in sales, since the introduction of new generation mobile devices. Mobile devices are increasingly penetrating into people's lives as efficient and convenient tools for communication and entertainment, and have become a ubiquitous and indispensable companion for many people. Furthermore the rapid development in the mobile technologies, ease of use and falling cost accelerate the permeation of mobile devices. In the end, this development has ensured that mobile devices have become an integral part of our daily life.

Meanwhile, due to the popularization and wider availability of sensor technology through cheaper embedded sensor devices, electronic sensors especially the wearable physiological sensors are used more and more in our daily life rather than only in lab environment [Fran13, Agga13]. According to [Bbcr13] the global market for sensors reached $68.2 billion in 2012 and is expected to reach almost $120 billion by 2019. For the wearable physiological sensors, several market prediction analyses indicate that there should be about 250 Million Wearable Health & Fitness Sensing Devices by 2017.[1]

Sensors are also widely used in mobile devices. Nowadays almost every mobile device has integrated several sensors such as accelerometer, gyroscope and GPS and additional wearable sensors including physiological sensors such as heartbeat rate sensor or skin conductance sensor that can

[1] http://www.prweb.com/releases/2013/10/prweb11283978.htm

be easily connected to the mobile device through standard communication interface like Bluetooth. In doing so, mobile devices are capable of processing various personal information of people (i.e. physiological information like heart rate, body temperature and etc.) and environmental information around people (i.e. location, air temperature, humidity and etc.).

On the other hand, improving daily life quality has gained lots of attention in the last several years. Health, one of the most important life qualities, is deeply impacted by individuals' lifestyles. Lifestyle such as sleep, socialization and exercise patterns can be directly connected to the presence of health related problems such as, high-blood pressure, stress [Nocc92], anxiety, diabetes and depression [Fox99, GBHF89]. Positive health effects can be observed when these wellbeing indicators (e.g., sleep, physical activity) are kept in healthy ranges. Hence, there has been a need for monitoring the health status of peoples' daily life. Indeed the interest of health monitoring on aspects of daily life is currently being reinforced by the Quantified Self (QS) movement, which aims to improve general well-being through self-tracking. The initiatives like quantified Self website[2] demonstrate the value of psychophysiological measurements (e.g. heart rate, ECG, EEG) for creating self-awareness about threats and opportunities for healthier life styles.

Taking advantage of the combination of mobile technology and sensor technology, mobile devices provide a new possibility of monitoring peoples' health status. In the meantime, the development of mobile devices is also showing the huge potential of enabling more potent real-time **data-intensive applications**, which are dealing with huge amount of mobile data, like in the case of m-Health and m-Commerce. Currently there are already existing mobile applications for personal monitoring, like the Nike+ ecosystem. However, the shortcoming of processing of this data in mainstream applications remains the same: real-time data is processed in isolation and statically, which also means that real-time data are not integrated with other data, especially the user's data, and the processing method is not dynamically adapted to the changes of the user's situation. Hence, there is a big gap between the promises of the existing approaches and the real impacts on the ordinary peoples' life.

Assume following scenario:

Peter is a middle-age (45 years old) employee in an IT company. Because his work is very exciting and challenging and involving meeting with leading-edge research and the necessity of frequent business travels on a regular basis, he is struggling with reducing the time spent abroad to have more time for his wife and his two kids. Consequently, he has poor nutrition habits and lives a sedentary lifestyle due to the constant need of optimizing the available time at the meeting places. After a medical examination he was told that he has high blood pressure and too high levels for the HDL (High-density lipopro-

[2] http://quantifiedself.com/

tein) cholesterol, which is one of the risk factors for developing further cardiovascular diseases. Peter is fully convinced that he should improve his lifestyle behavior, so he decides to do some fitness training such as jogging twice every week.

In order to improve the effectiveness of the training, Peter bought some sensors to monitor his personal vital data while jogging, following friends' advices. The sensors include a pedometer on a running shoe, which counts the precise number of jogging steps, and a wearable sensor with a chest belt, which measures his heart-beating and breathing rate in real-time. Peter also installed a popular fitness mobile application on his smartphone to monitor his training status using the data provided by sensors.

However, Peter is not satisfied with the application he is using, since it only processes the data statically and provides little personalized monitoring. For example, once the air temperature was high and Peter didn't feel well due to heavy breathing during the jogging. Considering his medicine record (risks for cardiovascular diseases), Peter couldn't know whether this is an indication of cardiovascular danger. While Peter should have slowed his pace down, the fitness application may actually encourage him to speed up on the contrary. Or, in a muggy summer morning, Peter ran on usual speed, while his fitness application had alarmed for several times and warn him to slow down, due to his unusual high heart rate. But Peter may feel perfectly fine as he observed other joggers running nearby at the same time without realizing his higher-than-usual heart beatings may be completely different from those healthy joggers. But the application that he is using doesn't provide such functionality to inform him about the comparison of the heart beatings.

Peter wants to find a new fitness App that can provide real-time monitoring that suits to his personal context (i.e. medical history) and actual situation, and links to other joggers in collaboration as well.

Indeed, the current more advanced solutions to processing mobile personal data are fairly limited on a) finding some simple predefined patterns in isolated mobile data, like detecting some arrhythmias in m-Health scenarios based on processing cardio data (ECG) (e.g. CardioDefender[3]) or b) "simple" transferring the personal data to servers in order to visualize it and share with others, especially in the case of fitness data, like in communities of runners (e.g. Run Keeper Portal[4], ca. 10 million users). The main limitation in all these approaches is to treat mobile devices as stand-alone processing units that have a well-defined set of processing tasks and well defined interaction with the environment. However in the todays Big/Fast Data hype, data-driven orientation influences mobile processing as well: the mobile data, put in the context of mobile environment, should drive

[3] http://www.everisthealth.com/content/cardiodefender/summary.htm
[4] http://runkeeper.com/partner

the processing "mode" of a device. Indeed, without context information of users' current situations, many real-time personal data are difficult to interpret in a correct way. For example, if an arrhythmia[5] is happening under the highly humid condition, the heart rate data should be processed in a different way than in the low humidity condition, depending on the intensity of the current physical activity of the patient (walking, stairs climbing, …), where the level of intensity should be calculated based on analyses of historical data (average values) etc.

It is obvious that the nature of such conditions is dynamic, since the current health status of a patient is a very important factor and it can be obtained through some other real-time sensing data, like the breathing-level data. The main issue is to enable the adaptive processing on a mobile device, whereas the adaptation is dynamic and data-driven. Indeed, the list of the causal chains (knowledge) can be quite long and extremely complex. Trying to capture all possible scenarios and hard-coding them in a mobile application is not a practical solution due to a limited processing and storage capacity of a mobile device. Closing the world of possible interesting situations that could be processed leads to constraining the usefulness of application potential of mobile computing. Furthermore, so far the monitoring applications have focused only on one individual person. In other words, the monitoring is done in the isolation. Each user is monitored separately. In many situations, the data collaboration between users can be very important during the monitoring of a specific person. For example, a cardio problem that has been sensed from one patient can be better and more comprehensively analyzed by having some values measured by another patient who is in a similar context. It is clear that an extended, adaptive, context-aware processing and collaborative monitoring is critically needed in order to exploit the full potential of mobile devices.

In addition, while the innovations in such key components as processor, memory and wireless technologies are ever fast paced, it remains widely agreed that mobile applications are still confined by the limited computation resources of mobile devices. To make it even worse, the power source of most mobile devices, the battery, has seen relatively slow improvement in the past decade. Battery capacity is growing only 5 percent annually [Robi09], which has become a major impediment to providing reliable and sophisticated mobile applications to meet the real-time data processing requirements from mobile applications.

In this thesis, we propose a foundation for the development of an innovative mobile application enabling efficient context-aware processing of real-time personal data streams taking into account the resource limitation on mobile devices. The research combines mobile computing, intelligent complex event processing, semantic technologies and real-time big data, in order to achieve an efficient processing of real-time sensor data on mobile devices for various use cases. In the next section we define the concrete research questions that constitute the core of this thesis.

[5] Arrhythmia is a kind of anomaly in the heart's functioning

1.2 Research questions

The various embedded sensors of mobile devices and different kinds of wearable sensors enable the modern mobile devices to collect a large amount of real-time user's personal and environment data. However, the limited resources on mobile devices and the lack of the processing methods, which enable to relate the context regarding the user's current situation to the real-time personal sensing data, impede mobile applications to exploit the full potential of real-time personal sensing data. Therefore, the main **research question** of this thesis is:

> ***How to efficiently process the personal data, especially from mobile devices and wearable sensors in real-time on mobile devices by taking the resource limitation of mobile device into account and provide dynamic adaptivity regarding user's current situation?***

Specifically, the main research question can be divided into following 5 sub-questions:

Q1: Is it possible to process real-time personal data on mobile devices?

Firstly it should be found out, whether it is possible to process real-time personal data on modern mobile devices. Furthermore, if it is possible, which technology is most appropriate to process real-time personal data on mobile devices?

Q2: How to achieve efficient real-time data processing regarding the resource limitation on mobile devices?

Considering the resource limitation on mobile devices, an efficient method to process personal real-time data on mobile devices should be developed. At the same time, it should not affect the performance of other mobile applications.

Q3: How to achieve context-aware processing regarding the user's current situation?

As mentioned, real-time personal data is difficult to interpret without additional information and the context of user's current situation. Hence context awareness should be achieved in the proposed solution in order to realize dynamic adaptation for real-time personal monitoring.

Q4: How to enable data collaboration among different users?

One of the advantages of today's mobile applications is that each mobile application is linked to a large amount of users over different parts of the world. However, the mainstream monitoring applications process the real-time personal data separately without collaborating with the data from other users. In order to exploit the potential of mobile applications, data collaboration among different users for personal data processing should be realized in the proposed solution.

Q5: How to adapt to the real-time changes in sensor/stream availability?

Wearable sensors connect to mobile devices normally using wireless connections such as Bluetooth protocol. However, the quality of wireless connection is not stable and can be affected by many factors such as radio frequency interference, out of communication range and etc. [Gibs12]. In addition the limited battery capacity of wearable constraints the usage time of wearable sensors. Consequently, the availabilities of personal data streams can be interrupted during the run-time due to the disconnection or shutdown of wearable sensors. In order to ensure reliable real-time personal data processing, the proposed solution should be able to adapt to changes and interruptions in sensors/stream availability.

1.3 Research contributions

In order to solve the research questions stated in the previous section, we argue that mobile applications require novel software architecture for real-time personal data processing, which provides dynamic adaptivity, context awareness, data collaboration and resource-aware data processing.

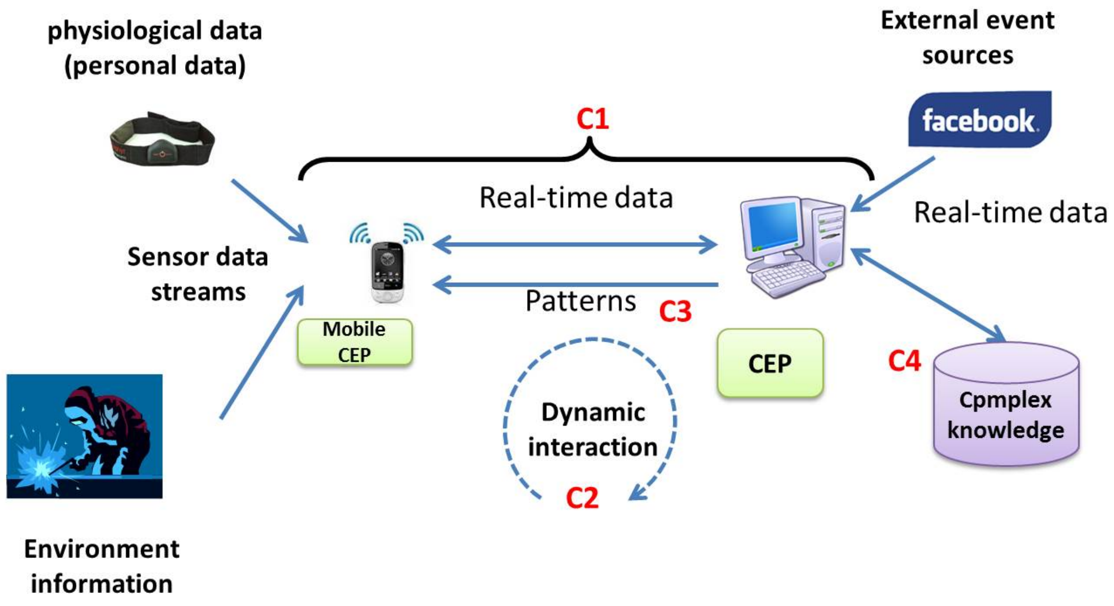


Figure 1-1 Overview of real-time personal data processing

In this research, we firstly develop a hybrid software architecture (as shown in Figure 1-1) for realizing a mobile distributed Complex Event Processing (CEP) system with additional backend server[6], in order to extend the computation capacity of mobile devices, with the purpose of solving the resource limitation problem of mobile devices. Then we develop the model of Monitoring Goal Network (MGN) to realize context awareness in our solution, which enables the correlation between real-time data and various users' context and domain knowledge. The MGN also provides dynamic adaptivity of processing tasks (patterns) according to the changes of the user's situation. Afterwards we use dynamic pattern distribution to achieve resource-aware pattern distribution for efficient data processing on resource limited mobile devices. In order to efficiently manage the patterns and event resources, we model them in semantic and use querying and reasoning to achieve dynamic adaptation to pattern problems in run-time.

The main research contributions are listed as follows:

- **C1: Event-driven hybrid software architecture for mobile applications:** firstly an event-driven hybrid software architecture has been developed for data-intensive mobile applications to extend the computation capacity with purpose of solving the resource limitation problem. The software architecture is based on hybrid architecture combining both the mobile component and its backend server part. It uses Complex Event processing (CEP) technology [Luck01] on both mobile devices and backend server to achieve scalable and reliable real-time personal data processing. Publish/subscribe middleware and Google GCM[7] push service is used to transmit the events between mobile devices and backend server, which enables efficient real-time data exchange across many different devices.

- **C2: Context-aware situation modeling and data collaboration:** Monitoring Goal Network (MGN) is developed to model user's situations and the adaptation to such situations corresponding to ECA (Event-Condition-Action) rules [KnEP00], in order to realize context-aware data processing. MGN models user's situations based on different monitoring goals, whereby different patterns (processing tasks) are used to discern the user's current situation. MGN also relates the real-time detected personal data to additional information (special domain knowledge like health knowledge base or fitness knowledge base) and to user's context (e.g., the user's medical records). In addition, the assigned monitoring goals of users enable the data collaboration among users, by searching for the relevant users, who have the same monitoring goals. The proposed system also provides the possibility of defining custom criteria for searching for relevant users for data collaboration.

[6] The server part of our software architecture can be deployed either on a single server or cloud infrastructure. In the prototype we use a backend server to deploy the server part, so we call it backend server in the rest of this thesis.

[7] http://developer.android.com/google/gcm/index.html

- **C3: Resource-aware dynamic pattern distribution:** In order to achieve efficient and reliable data processing on mobile devices, a resource-aware pattern distribution algorithm has been developed to distribute patterns to both mobile device part and backend server part taking the available resources into account. The distribution algorithm calculates the pattern distribution not only according to the current workload of devices, but also according to the commutation workload caused by different distributions. It also calculates the optimal allocation of event resources for the patterns, with the aim of optimizing the performance for the whole system.
- **C4: Semantic-based dynamic pattern management:** A semantic-based pattern model has been developed for managing the patterns and event resources of users. The available event resources of users are updated dynamically during run-time. By pattern deployment that is based on the information in the pattern model, the proposed system checks the availability of all events requested by the pattern. In case that the requested events are not available, alternative solutions will be provided, via deploying additional patterns to create missing events and the deployment of replacement patterns, which provide the same function as the original patterns. During run-time, if the availability of event resource is changed, the system searches for the replacement in the pattern model, in order to ensure the integrity of the system.

1.4 Thesis organization

The whole thesis is organized in three parts, each containing several chapters.

The first part describes the foundation of the research, including this chapter. It gives an overview and motivation of the thesis, and provides background information. We introduce mobile technology and sensor technology in Chapter 2 and the complex event processing technologies in Chapter 3, which are the most important hardware and software foundation of this thesis. The Chapter 4 provides a literature survey of the related work in different domains and comparing them to our work , including mobile and distributed CEP system, efficient mobile computing, semantic-based pattern and sensor models, and mobile sensing applications and systems.

The second part is devoted to the discussions of the efficient context-aware real-time personal data processing, which describes the main work about an innovative software architecture for personal data processing mobile applications. We start this part by providing analysis of the requirements for personal data processing mobile applications in Chapter 5. The whole software architecture is introduced in Chapter 6, with detailed description of stream management and communication between mobile part and server part. Chapter 7 describes the monitoring goal network (MGN) in detail, which models the user's situation and defines the conditions and adaptivity to deal with such situations. The chapter also describes the data collaboration approach based on monitoring goals defined in MGN. In this chapter we also provide examples that give readers a better understanding about the MGN. Chapter 8 introduces the intelligent resource-aware processing distribution. It shows the pattern distribution model of our solution. The algorithm for pattern distribution based on the current resource of mobile device is described in detail. We again provide examples to explain the inner-work of the algorithm. Chapter 9 introduces the semantic-based pattern model applied in our solution, which is used to manage patterns dynamically in real-time. We describe the mechanism for adaptation of changes in availabilities of data streams.

The third part consists of two chapters: one for evaluation and the other for conclusions. Chapter 10 provides the evaluation of our solution, including performance evaluation and use case evaluation. We evaluate performance of our solution in terms of mobile CEP performance, performance of pattern distribution algorithm and performance of pattern adaptation. In addition, two use case evaluations are provided, one in m-Health domain and another in m-Fitness domain. In the last Chapter 11 we summarize our work of this thesis and provide an outlook for possible future work in this research field.

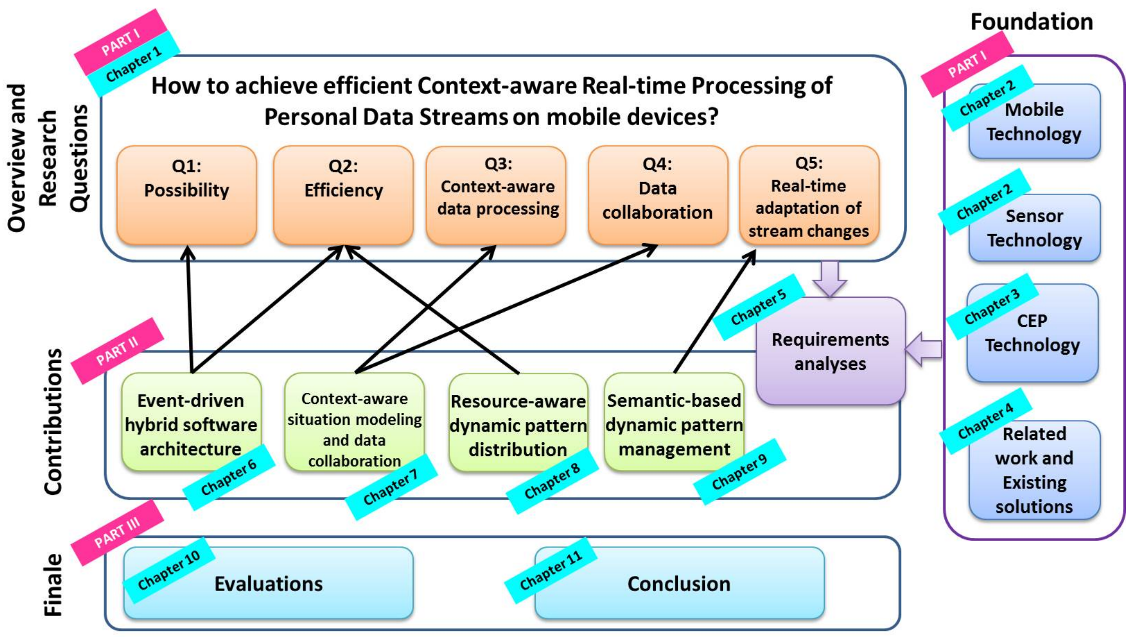

Foundation
PART I
Chapter 1
Overview and Research Questions
How to achieve efficient Context-aware Real-time Processing of Personal Data Streams on mobile devices?
Q1: Possibility
Q2: Efficiency
Q3: Context-aware data processing
Q4: Data collaboration
Q5: Real-time adaptation of stream changes
PART I
Chapter 2
Mobile Technology
Chapter 2
Sensor Technology
Chapter 3
CEP Technology
Chapter 4
Related work and Existing solutions
Chapter 5
Requirements analyses
PART II
Contributions
Event-driven hybrid software architecture
Context-aware situation modeling and data collaboration
Resource-aware dynamic pattern distribution
Semantic-based dynamic pattern management
Chapter 6
Chapter 7
Chapter 8
Chapter 9
PART III
Finale
Chapter 10
Evaluations
Chapter 11
Conclusion

Figure 1-2 Thesis organization

Figure 1-2 illustrates the organization in this thesis. On the top are overview and research questions, which have been introduced in the current chapter. On the right of the figure is related foundation, including mobile technology, sensor technology, CEP technology, and related work and existing solutions, which are described in Chapter 2 to Chapter4.

Based on research questions and foundation we analyze the requirements to our approach in Chapter 5. In the middle are the contributions, which are described in Chapter 6 to Chapter 9. The arrows between contributions and research questions indicate the relationship between them. On the bottom is the finale part, containing evaluation and conclusion chapters.

1.5 Related publications

In this section a list of publications I have published is provided that are related to the core of this thesis. Many topics and discussions regarding methodology, methods, algorithms and framework of this thesis have been published during the time period from November 2010 to May 2014. Below I describe the publications briefly to provide a guide of the development of the approach described in this thesis.

The first raw idea of this thesis was described in [XWSH10]. I discuss the approach about processing sensor data using complex event processing in the AAL (Ambient Assisted Living) domain. Afterwards I introduce semantic technology into CEP-based sensor data processing. An approach of using CEP technology, semantic technology and sensors to monitor the office status for efficient energy consumption was described in [XSSA+11, SMXS+11]. In developing that approach I use an ontology to model the monitoring procedure, patterns and sensors to achieve dynamic adaptation. A demo about this approach was described in [XSMA11].

In the scope of an EU project ARtSENSE[8] I extended my approach by integrating mobile devices and wearable sensors. I combined CEP, semantic technologies and mobile computing to process the sensor data from wearable sensors to conduct the attention of visitors in museum environment in real-time [XSSC+12]. Some demos have been designed and shown in the conferences [XSSS12a, XSSS12b, and XSSS12c].

Finally I integrated my past ideas and develop the current approach for using mobile computing, intelligent complex event processing, semantic technologies and real-time big data to achieve an efficient processing of real-time personal data from sensors on mobile devices. An overview of this approach and some features were initially proposed in [XSSK13]. [SXSS14a] has described some technical features in the scope of a collaborative remote monitoring use case. In addition demos based on this approach were published and shown in [StRX13, SXSS14b] and a tutorial of this approach was given oat DEBS conference (International Conference on Distributed Event-Based Systems 2014) [StSX14].

[8] http://www.artsense.eu/

2 Introduction to Mobile Devices and Wearable Sensor Technologies

Since last decade mobile technologies have brought significant changes in our life. Modern mobile devices,[9] including smartphones and tablets, are becoming increasingly ubiquitous and provide ever richer functionalities through numerous mobile applications. The increasing popularity of mobile devices with their embedded sensors and various wearable sensors leads to an expansion of mobile sensing applications, especial in the m-Health and m-Fitness domains. In this chapter we give an overview of mobile technologies and wearable sensor technologies.

2.1 Mobile Devices

Since iPhone as a new generation smartphone was introduced by Apple in 2007, the mobile device market has been growing at a tremendous rate. The technology has been advancing rapidly and the market research company IDC shows that mobile device sales, including smartphones and tablets, have reached 1.25 billion in 2013 [IDC13a].

Mobile device is a small handheld computing device, which is also called handheld device or handheld computer. According to [Wies03] a mobile device is defined as an extremely portable, self-contained information management and communication device, characterized by the following three aspects:

- It must operate without cables, except temporarily (recharging, synchronizing with a desktop, etc.).
- It must be easily used while in one's hands, not resting on a table.
- It must allow the addition of applications or support Internet connectivity [mobile networks 3G/4G, WIFI and etc.].

Most mobile devices also have high-resolution touchscreen as input device and integrated cameras and sensors such as GPS, Accelerometer and Gyroscope. Thanks to the rapid innovation in mobile technologies, mobile devices have embedded more and more new features. And the lines of delineation between different types of computing devices are becoming increasingly ambiguous, as mobile devices now offer crossover in form and functionality [Kros08].

[9] In this thesis mobile devices indicate handheld computing devices, like smartphones and tablets. Laptops are not included due to the size and different mobility.

In the next subsection we start the introduction to modern mobile devices by reviewing the history of mobile device innovation.

2.1.1 The history of mobile device innovation

In this section we give a short review of the mobile device innovation history. Surprisingly the occurrence of mobile devices happened in the mid-1970s, long before many would have assumed.

The definition of mobile phone was created and used since 1940s [Farl05]. However prior to 1973, mobile telephony was limited to phones installed in cars and other vehicles. The first real handheld mobile phone appeared in 1973 Motorola [CDML+75]. The first call was made on April 3, 1973, when Dr. Martin Cooper, a senior engineer at Motorola, called Dr. Joel S. Engel of Bell Labs and informed him he was speaking via a mobile phone. This first handheld mobile phone weighed 1.1 kg and measured 23 cm long, 13 cm deep and 4.45 cm wide and could offer a talk time of just 30 minutes and took 10 hours to recharge. The first commercial mobile phone was released by Motorola in 1983 known as the Motorola DynaTAC 8000X, providing 30 minutes of talk-time and six hours standby [GiSt87].

In 1992 Nokia releases Nokia 1011, which was the world's first commercially available GSM digital mobile phone [Kobl11]. It measured 195 x 60 x 45mm, weighed 475g and featured a monochrome LCD display and an extendable antenna. It provided 90 minutes talk time, 12 hours stand-by time and was able to receive SMS message.

In 1993, Bellsouth and IBM announced their creation of the Simon personal communicator phone, touted as the world's first smartphone [Kobl11]. Simon was designed to be a cellphone first and a computer second, according to the product's media release. It was a mobile phone, pager, fax machine, and PDA all rolled into one. It included a calendar, address book, clock, calculator, note-pad, email, and a touchscreen with a complete keyboard.

In 2000, the first phones with built-in cameras became publicly available. It was the Sharp J-SH04, introduced by J-Phone and the Sharp Corporation in Japan [Kjel13]. The J-SH04 had an 110,000 pixel resolution (0.1 megapixels), a color LCD screen, one-touch Internet access and a speaker phone.

In 2001 Ericsson T68, as the first mobile phone with a color display, was launched. It provided a passive LCD-STN with a resolution of 101×80 and 256 colors [RDGN04].

In 2003 BlackBerry 6210 was unveiled by Research in Motion (RIM) as part of BlackBerry’s Quark series. The BlackBerry 6210 was the company's first device to offer: E-Mail, testing, a web browser and BlackBerry Messenger service, allowing for web-based communication between BlackBerry users. In the same year the first mobile phone using Palm OS called Treo 180 was available in the market.

In January 2007, Apple launched its first iPhone. The company described the phone as combining three products into one handheld device: a mobile phone, an iPod and a wireless communication device. One of the original iPhone's more revolutionary features was that it allowed users to command the device using only their fingers on a touch screen. Other new functions included a visual voicemail box, touchpad keyboard, a photo library that could be linked to a remote computer and an almost nine-centimeter display for watching movies and television.

On October 22, 2008 the first publicly available smartphone running Android system, the HTC Dream (also known as the T-Mobile G1), was released. It provided a customizable graphical user interface, integration with Google services such as Gmail, a notification system that shows a list of recent messages pushed from apps, and Android Market for downloading additional apps.

In October 2010 the windows phones from Microsoft entered smartphone market. Windows Phone features a user interface based on Microsoft's Windows Phone design system, codenamed Metro.

The history of tablets is shorter than the history of mobile phones. The first tablet was created in 1989 by Jeff Hawkins, called GRidPad[10]. It ran MS-DOS and the military bought a few but consumers mostly ignored it, since it was expensive and heavy.

In 1993 Apple unveiled its first tablet Newton MessagePad, which was also called "personal digital assistant" or PDA, for taking your calendar/todo list and a few apps with you. With a stylus, you could write on it and it would recognize your handwriting [Culb94].

By 1997 Jeff Hawkins was back with PalmPilot [Pogu98], the first affordable PDA. Eventually, the PalmPilot would use touchscreens and become very popular. This device proved that people wanted a third type of mobile device between a mobile phone and a laptop, if it was affordable and was easy to use.

Microsoft introduced its first tablet in 2000 and released in 2002, which had a version of its XP operating system designed for it. Afterwards there are lots of tablets with Windows system on the market, such as LS800 from Motion Computing, which was the smallest tablet at that time. However they were costly and not popular with consumers. They were mostly used in factories, by the military and by other field workers.

In 2010 Apple CEO Steve Jobs introduced the Apple iPad [Appl10], which was the most successful tablet in the history. It runs Apple's iOS and carries a 9.7-inch multi-touch-screen and built-in WiFi.

[10] http://www.computinghistory.org.uk/det/6565/GRidPad-1910/

The iPad provides a user-friendly interface for multimedia playing and performing Internet functions such as web-browsing and emailing. More functions can be extended by downloading and installing relevant apps.

Following upon Apple, also in 2010 many Android tablets from various producers were brought to the market too.

In June 2012 Microsoft announced its new tablet Surface with Windows 8 system and it was released in October.

2.1.2 Market analysis

The rapid innovation in mobile technologies leads to a significant changes in the way we work and live. The smartphones have become a ubiquitous companion for many people and the numerous mobile Applications (Apps) extend the functionality of mobile devices in various domains.

In the fourth quarter of 2010 the shipment of mobile devices has surpassed the shipment of PC [IDC10] for the first time and right after that the significant growth of mobile device market continues. Instead of traditional PCs, mobile devices have become the primary tools through which people access information.

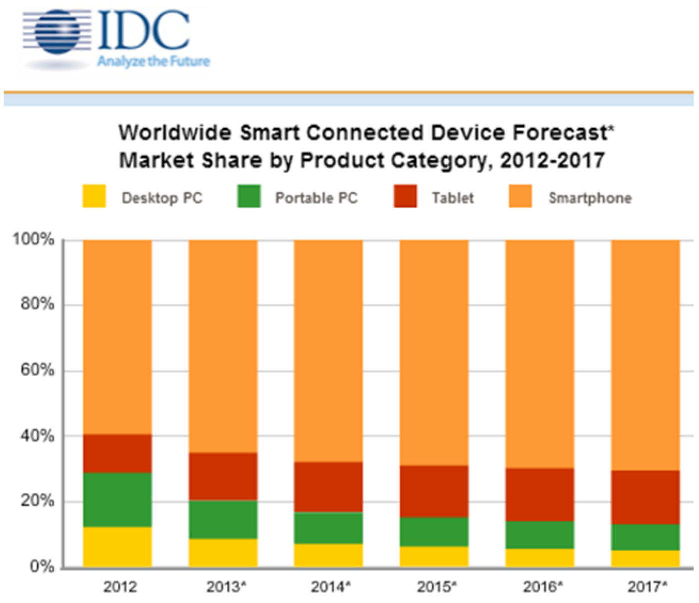


Figure 2-1 Worldwide smart connected device forecast market share by product category, 2012-2017 [IDC13b]

According to the recent market report by IDC [IDC13b] in 2013, the mobile devices including tablet and smartphone have captured the most market shares in value by almost 80%, as shown in Figure

2-1. Contrarily the traditional PCs including desktop PC and Portable PC have lost the most market share and have only now 20% market share. The report has also predicted that the growth of mobile devices from 2013 to 2017 will be about 75%. Table 2-1 summarized the smart connected device market by produce category, unit shipments and market share for 2013, and prediction for 2017.

Table 2-1 Smart Connected Device Market by Product Category, Unit Shipments and Market Share, 2013 and 2017 shipments in millions) [IDC13b]

Product Category	**2013 Unit Shipments**	**2013 Market Share**	**2017 Unit Shipments**	**2017 Market Share**	**2013—2017 Growth**
Desktop PC	134.4	8.6%	123.11	5%	-8.4%
Portable PC	180.9	11.6%	196.6	8%	8.7%
Tablet	227.3	14.6%	406.8	16.5%	78.9%
Smartphone	1,013.2	65.1%	1,733.9	70.5%	71.1%
Total	**1,556**	100%	**2,460.5**	100%	**58.1%**

As introduced in the previous section there are several different operating systems used by mobile devices. However the mobile devices using different operating systems show also different trends in the market. Table 2-2 shows the shipment and market share regarding the different operating systems for third quarter in 2013.

Table 2-2 Top Four Operating Systems, Shipments, and Market Share, Q3 2013 (Units in Millions) [IDC13a]

Operating System	**3Q13 Shipment Volumes**	**3Q13 Market Share**	**3Q12 Shipment Volumes**	**3Q12 Market Share**	**Year-Over-Year Change**
Android	211.6	81.0%	139.9	74.9%	51.3%
iOS	33.8	12.9%	26.9	14.4%	25.6%
Windows Phone	9.5	3.6%	3.7	2.0%	156.0%
BlackBerry	4.5	1.7%	7.7	4.1%	-41.6%
Others	1.7	0.6%	8.4	4.5%	-80.1%
Total	**261.1**	**100.0%**	**186.7**	**100.0%**	**39.9%**

Obviously the Android system has the most users and Apple's iOS follows next. Both systems have captured more than 90% market share. The year-over-year change for Android devices increased significantly, iOS on the other hand had lost some market. Windows Phone is winner in the year-over-year change percentage wise despite its low market share. The other operating systems like BlackBerry experienced difficulty in the last year and their market share shrank obviously.

Based on the market trend we selected Android system as the research platform for our work in this thesis.

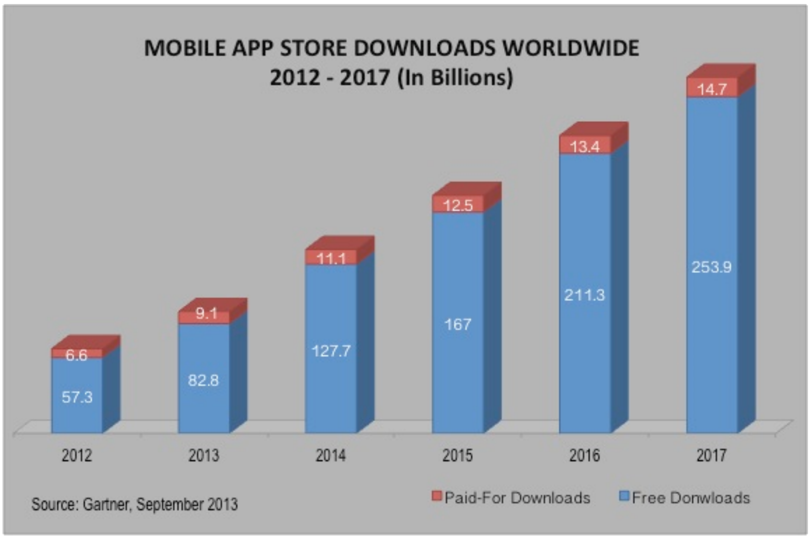


Figure 2-2 Worldwide mobile app store downloads [Gart13]

The growth of mobile device market leads to a corresponding increase in mobile app market. As shown in the chart above according to [Gart13] in 2013 mobile app downloads have increased by 28 billion. Furthermore, in 2014 an increase of 47 billion is expected. The growth rate will keep increasing till 2017. We can also observe that the number of free downloads will increase rapidly, indicating a higher penetration by mobile devices in the emerging markets. The number of paid app downloads will also show an increase, although the rate of increase will relatively be slower, indicates that more users will become aware of apps they want to use and demand for quality apps will increase.

2.2 Sensors

Nowadays the electronic sensors special the physiological and wearable sensors are used more and more in our daily life rather than only in the lab environment. Almost every mobile device has been integrated with several sensors such as accelerometer, gyroscope and GPS [HBPW08]. Meanwhile a lot of wearable physiological sensors have existed on the market as they can be easily connected to mobile devices through wireless communication protocol like Bluetooth. These sensors enable mobile devices to collect various personal data, such as heartbeat rate, body temperature and skin conductance. In this section we give a brief introduction to today's sensor technologies.

2.2.1 Sensor

According to [Wils04], **sensor** is defined as a device that converts a physical phenomenon into an electrical signal. It represents the part of the interface between the physical world and the world of electrical devices. Sensors are widely used in everyday objects such as touch-sensitive elevator buttons (tactile sensor) and lamps which dim or brighten by touching the base.

Sensors can be divided into different types based on their usage. The type of the sensors, which measure raw physiological signal data from human body, is called **physiological sensor** [BaLG04] or **biosensor** [TKWW87, Lowe89, and ChMa02]. Physiological sensors provide real-time, on-board derivations of basic parameters such as heart rate and respiration rate, which enables health monitoring for people, especially for patients.

Taking advantage of rapid innovation in wireless technology and miniaturization technology, sensors are becoming smaller and lighter. Also the advent of smart fabrics [ElYM00, Matt06] in recent years has allowed people to stay attached to sensors without the issues of discomfort, large bulky technology or skin break down associated with sticky patches. Therefore in recent years the wave of **wearable sensors** [LeMa02] hit the market. Currently the most wearable sensors on the market are physiological sensors, such as heart rate sensor, respiration sensor and EEG sensor.

2.2.2 Mobile sensing

Sensors are also widely used in mobile devices and have been regarded as a critical component that opens up mobile devices to new advances across a wide spectrum of applications domains. Sensor enabled mobile devices are set to become even more central to people's lives as they become intertwined with existing applications such as social networks and new emerging domains such as green applications, recreational sports, global environmental monitoring, personal and community healthcare, sensor augmented gaming, virtual reality, and smart transportation systems.

Recent technology advances and miniaturization have accelerated the convergence between mobile devices and powerful computers. The computation performance and storage capabilities of mobile devices are ever growing while integrating a suite of sensors including accelerometer, microphone, GPS, digital compass, gyroscope, and, in the future, air quality and chemical sensors [HBPW08]. By taking advantage of mobile devices' computational power and sensing capabilities, and their tight coupling with users' daily lives, mobile devices can become very compelling platforms to recognize users' activities and personal information [EOLL+08].

2.2.3 Market analysis

During the last several years improving daily life quality has gained lots of attention. Sensors have been widely used to monitor the life quality such as health status, which leads to an expansion of sensors, especially mobile wearable physiological sensors.

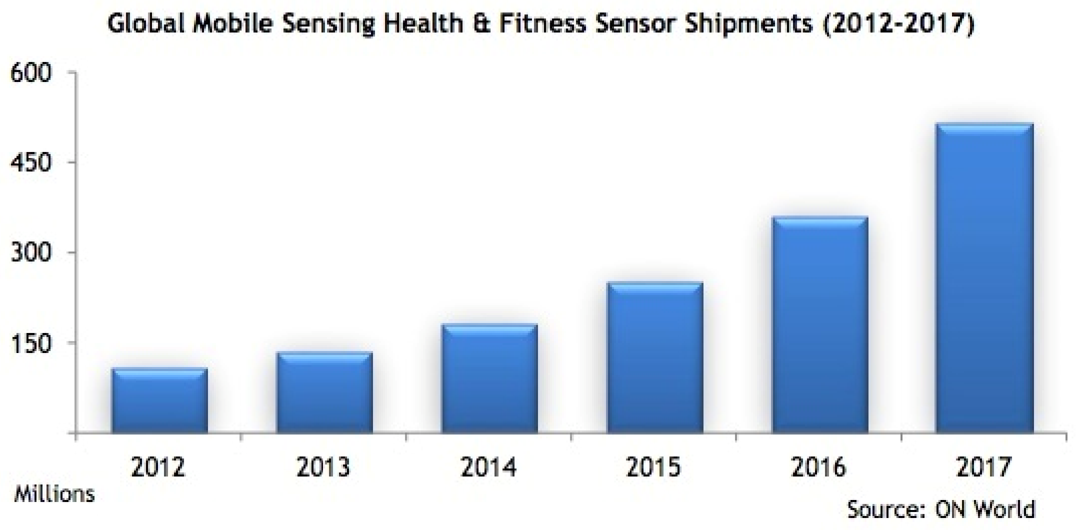


Figure 2-3 Global mobile sensing health and fitness sensor shipments (2012-2017)

As shown in Figure 2-3, according to the market reports [HaGu13a, HaGu13b], 515 million sensors for wearable, implantable or mobile health and fitness devices are predicted to be shipped globally in 2017, up from 107 million in 2012.

2.2.4 Sensor examples

In this subsection we introduce some sensors that are used in this thesis for experiments. Since the target user group of apps using the proposed software architecture in this thesis is ordinary people, we select the sensors, which are the mainstream sensors on the market and are affordable by most users.

The Zephyr BT HxM sensor [Zeph10] is a heart rate sensor, which also contains an internal accelerometer and provides algorithm-derived speed, distance and stride count. The sensor carries a bluetooth module, which enables the sensor to connect with mobile devices through Bluetooth connection.

Zephyr BT HxM sensor

Figure 2-4 Zypher BT HxM sensor (image source: http://www.zephyranywhere.com/)

The Zephyr BioHarness 3 sensor [Zeph12] is an ECG and breathing sensor. It is attached to a light-weight smart fabric strap or garment. It detects various physiological information, including heart rate, ECG, breathing rate, breathing wave amplitude, skin temperature and posture. It provides both Bluetooth connectivity and IEEE 802.15.4 connectivity, which enables the data transmission to mobile devices.

Zephyr BioHarness 3 sensor

Figure 2-5 Zypher BioHarness 3 Sensor (image source: http://www.zephyranywhere.com/)

3 Introduction to Complex Event Processing

Complex Event Processing (CEP) is a computer science discipline that has developed a set of techniques and tools to enable real-time computing. In this chapter we describe the foundation of the CEP technologies including fundamental definitions and terminologies. We also introduce the CEP architecture and explain the main building blocks of the CEP architecture.

3.1 Event

In order to better understand the complex event processing technologies, we start this section by explaining what an ***event*** is and what is meant by an ***event.***

Luckham [Luck01] defines an Event as an object that is a record of an activity in a system with three aspects:

- ***Form***: The form of an event is an object. It may have particular attributes or data components. A form can be something as simple as a string or more often a tuple of data components. The data components can also include a description of its significance and relativities.
- ***Significance***: An event signifies an activity. This activity is the significance of the event. An event's form usually contains data describing the activity it signifies.
- ***Relativity***: An activity is related to other activities by time, causality, and aggregation. Events have the same relations to one another as the activities they signify. The relationships between an event and other events are its relativity.

In [EtNi10], the authors define an event as an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. An event can also be used to describe a programming entity that represents such an occurrence in a computing system. Chandy et. al. [ChCC07] define an event as a significant change in the state of universe. In this sense every piece of new data can be considered as an event since it changes the state of universe. Events that require a reaction are forming a situation [AdEt02]. Not only the occurrence of something denotes an event but also the absence of expected events conveys information [ChSc09].

In sum, an event is an object that presents an occurrence or non-occurrence of an activity that has happened in a particular system or domain. It can happen not only in the real world, but also be in artificial domains such as training simulators, virtual worlds, and similar virtual environments.

Example: ***Heart rate events from heart rate sensor***

An event signifies the activity of the heart rate measurement of a user, which is sensed by a heart rate sensor. The measurement contains the following data fields: sensor id, measurement value, measurement time and so on. The event contains all the data carried by measurement and extra data fields such as event id, user id and relation to other measurements. So the heart rate event can be modelled in the following forms.

JAVA class

```
Class HeartRateEvent { String          event_id;
                       String          user_id;
                       String          sensor_id;
                       Measurement     value;
                       Time            measurement_time
                       Relation        (id1, id2,...)
                       }
```

XML/XSD

```
<?xml version = "1.0" encoding = "utf-8"?>
<xs:schema xmlns="www.mcep.fzi.de" targetNamespace=www.mcep.fzi.de
xmlns:xs=http://www.w3.org/2001/XMLSchema elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:complexType name="Event">
      <xs:sequence>
        <xs:element name="event_id" type="xs:string"/>
        <xs:element name="user_id" type="xs:string"/>
        <xs:element name="sensor_id" type="xs:string"/>
        <xs:element name="measurementValue" type="xs:integer"/>
        <xs:element name="measurementTime" type="xs:date"/>
        <xs:element name="relation" type="xs:String" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

The event described above is modelled both in Java and XML (defined in xml schema) forms. One heart rate event can be presented either as Java class instance or XML document. Each event belongs to an event type, which is a specification for a set of event objects that have the same semantic intent and same structure; every event object is considered to be an instance of an event type [EtNi10].

Furthermore, an event can be either simple or complex [ChEA11]. A simple event is atomic and does not contain further events. In comparison to a simple event, a complex event is composed of other simple or complex events. In [ChEA11] complex events are also described as summary-level facts. In [Luck01] Luckham defines that a complex event signifies an activity that happens over a time interval where a set of other events happened before. Moreover, there are relationships between these events. For instance, time, causality and aggregation are described as three most common and important relationships between events. Time is a relationship that orders events. This relationship depends on a clock. Causality is a dependence relationship between activities in a system. An event depends on other events if it happened only because the other events happened. Aggregation is an abstraction relationship. If event A signifies an activity that consists of the activities of a set of events B_i, then event A is an aggregation of events B_i.

3.2 Pattern

Pattern or called ***Event Pattern*** is another important definition in event processing, which describes the event processing logic. The detection of patterns over events is the core in the event processing [EtNi10], [Luck01], and [ChEA11]. An event pattern is a template that matches certain sets of events, which you want to find. Furthermore it describes precisely not only the events but also their causal dependencies, timing, data parameters and context [Luck01]. In [EtNi10] the authors define an event pattern as a template specifying one or more combinations of events. Given any collection of events, one or more subsets of those events may be able to be found that match a particular pattern. We say that such a subset satisfies this pattern.

Event patterns are used to create complex events that signify a set of events. The same event pattern can continuously trigger complex events of the same type. A pattern describes the concrete occurrence context of such event set. This context contains the event parameter of each event and the relationship between the events in the set using event operators. Event patterns are described in languages called event pattern language (EPL). Different event processing agents (explained in section 3.3.2) can have different EPLs.

Example: ***Pattern in Esper EPL***

every HeartRateEvent(user_id="Peter", value>80) where timer:within(60 seconds)

Esper[11] is an event processing agent (engine) that has its own EPL. The pattern in the above example processes the heart rate events to alert on each heart rate of Peter with a heartbeat value greater than 80 bpm within the next 60 seconds.

Based on the pattern matching operations, according to [EtNi10], the event patterns can be divided into two types: Basic patterns and Dimensional patterns.

- Basic patterns: These are simple patterns that relate to basic operations, including logical operations, threshold operations, subset selection operations, on event types or on collections of event types and don't depend on the timing or ordering of the incoming events. Basic patterns are frequently found in event processing applications.
- Dimensional patterns: These are patterns that relate to time, space, or a combination of time and space.

Next we introduce some matching operations of event patterns, which are often used in CEP applications.

Logical operations are the most frequently used pattern operations in CEP. Traditional logical operations have three operators: conjunction operator (AND), disjunction operator (OR) and negation operator (NOT).

Example: ***logical operation patterns***

Pattern1: **EventA AND EventB**

Pattern2: **EventA OR EventB**

Pattern3: **NOT EventA**

Pattern 1 uses a conjunction operator. It is fulfilled when both EventA and EventB happens. Pattern2 uses a disjunction operator and is fulfilled when either EventA or eventB happens. Pattern 3 uses a negation cooperator and is fulfilled when EventA doesn't happen.

[11] http://esper.codehaus.org/

Threshold operations are based on aggregation operations, which are performed against the set of participant events. The result of the aggregation is then compared against a threshold value. The assertion of the comparison can be one of the relations =, ≤, ≥, <, >, ≠. The threshold operations include following types:

- *Count:* a count operation counts the number of participant event instances and compares the value against threshold value.
- *Average:* an average operation calculates the average value of a special attribute over all participant events, and compares the value against threshold value.
- *Maximum and minimum:* a maximum/minimum operation calculates the maximal or minimal value of a special attribute over all participant events, and compares the value against threshold value.

Temporal order operations are operations in which time plays a major role. The most common temporal order operation is sequence operation.

Sequence operation is satisfied if the participant event set contains at least one instance of each event type mentioned in the relevant event types list, and if the order of these event instances matches the order of their types in the list.

Example: *sequence pattern*

EventA SEQ EventB

In the example above the pattern uses a sequence operator and is fulfilled whenever EventB happens after EventA has happened.

3.3 Event processing architecture

In the previous sections we have described the basic definition of events and patterns in order to bring readers into the world of event processing. In this section we finally come to the definition of event processing, and further present the architecture of event processing.

According to [EtNi10], Event Processing is computing that performs operations on events. Common event processing operations include reading, creating, transforming, and deleting events. Event processing aims to detect or to report the situations of interest based on the analysis of events from different event sources in real-time in order to achieve immediate reactions to the detected situations. Complex Event Processing (CEP) indicates the same operations as event processing, but performs on complex events [LSAB+11]. In this thesis event processing and CEP are used synonymously. Since 1990s CEP has already become the foundation of modern real-time information systems in the research field. The CEP area has roots in discrete event simulation and active database area. CEP provides a set of techniques and methods to analyze and understand event-driven systems [Luck01].

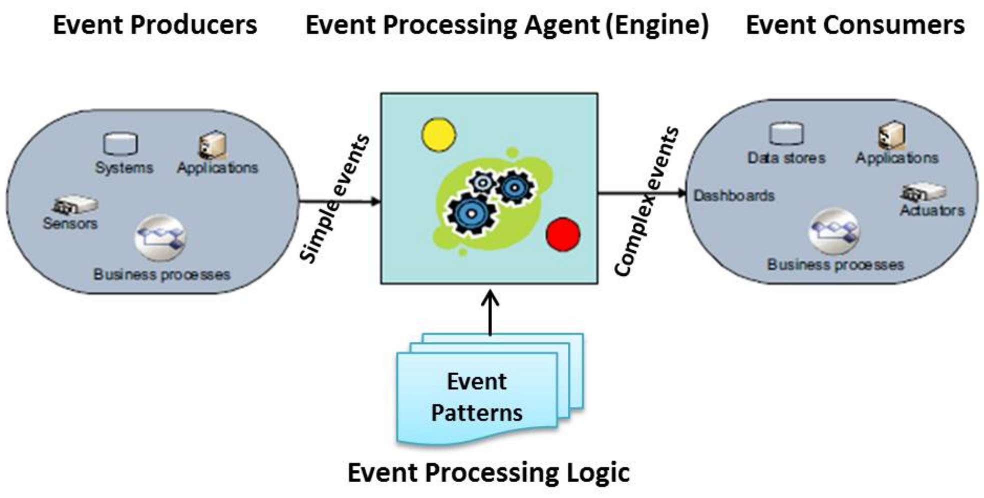


Figure 3-1Event Processing architecture based on [EtNi10]

Event processing architecture is a software architecture, which enables event processing applications using patterns to promote the reading, creating, transforming, and deleting events. As event processing applications are applied to different domains they have different requirements. Hence not all event processing applications are the same, of course, but by and large most of them have a structure that consists of a set of common components [EtNi10].

Figure 3-1 shows a basic event processing architecture with the common components including event producers, event processing agent (normally called event processing engine) and event consumers. The event processing logic is described as event patterns and is deployed in event processing agent.

However, some event processing applications have more complex structure and require more event processing agents. In such cases we need an **Event Processing Network (EPN)**, which is a collection of event processing agents, producers, consumers, and global state elements connected by a collection of channels.

Figure 3-2 displays an example of EPN with all of its components. The main components: *event producers, event processing agent* and *event consumers* are the same as those in event processing architecture. A *state* component refers to data that is available for use both by event processing agents and by contexts [EtNi10]. This data may be system-wide global variables, reference data used to enrich events and event stores that hold past events. An *event channel* receives events from one or more source processing elements, makes routing decisions, and sends the input events unchanged to one or more target processing elements in accordance with these routing decisions [EtNi10].

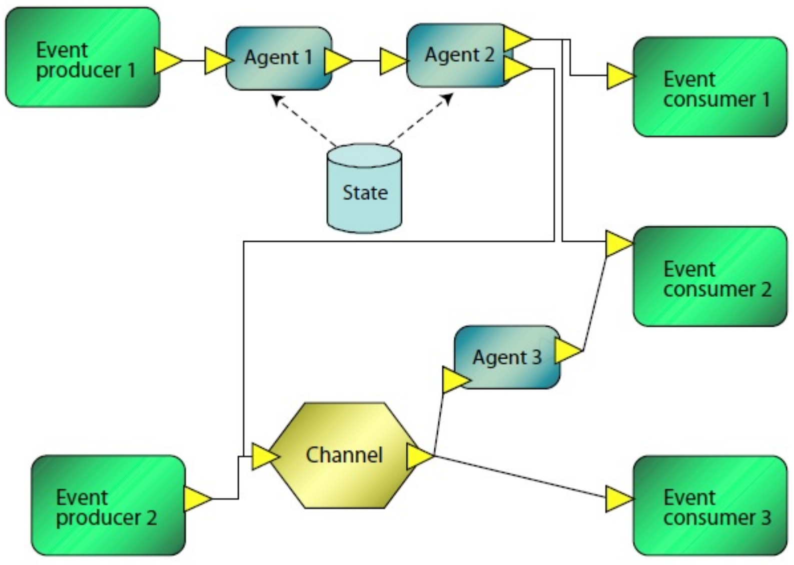


Figure 3-2 An Event Processing Network ([EtNi10])

In the following sections we describe these three main components of the event processing architecture including *event producers, event processing agent* and *event consumers.*

3.3.1 Event Producer

An *event producer* is an entity at the edge of an event processing system that introduces events into the system [EtNi10]. An event producer is also known as an event source, which creates events according to certain event reporting logic which is embedded in the event producer. According to [Luck01] event producers should have two functions: Observation and Adaptation.

- **Observation**: Event producers should be able to access and observe the activities at any level of a hierarchical system without changing the system's behavior.
- **Adaptation**: Event producers must transform observations into event objects that can be processed by CEP.

Regarding the various types of events, event producers also have many types that can be summarized into the following three categories [EtNi10]:

- **Hardware event producer**: Event producers in this category detect physical signals (e.g., temperature, light and etc.) and generate events that indicate activities in the physical environment, where it is integrated. Sensors like temperature sensors or humidity sensors are typical hardware event producers. Hardware event producers are widely used in medical applications, device management, security applications and etc.
- **Software event producer:** An event producer can also be part of a software application, which means it is a piece of application logic and generates event objects. Software events can also be produced by instrumentation technique. Here the events are not generated by application code itself; instead they are produced by software that is monitoring the application, looking for noteworthy activity.
- **Human interaction event producer:** Some events are generated directly by human interaction, albeit with a bit of software and hardware assistance. For example, the delivery confirmation produced by the driver's handheld device.

3.3.2 Event processing agent

The concrete processing of events tasks place in *Event Processing Agent* component. As defined in [LSAB+11, EtNi10], an event processing agent is a software module that processes events and plays a major part in event processing architecture.

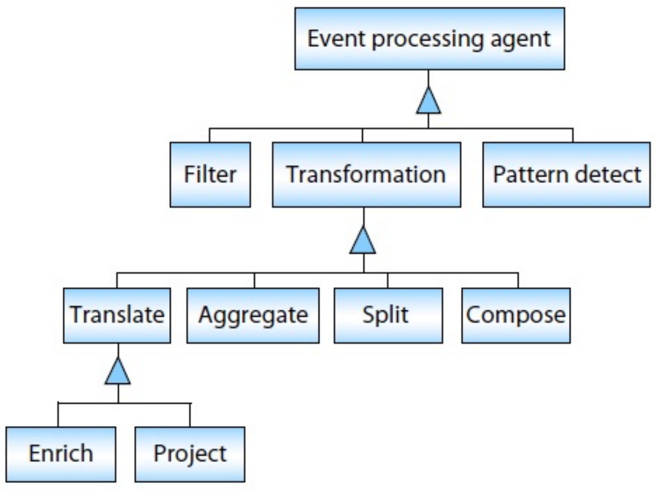


Figure 3-3 Type hierarchy of event processing agents according to [EtNi10]

An event processing agent has three logic functions: Filtering, Matching and Derivation [EtNi10].

- **Filtering:** It selects the input events that participate in the processing. The filtering step takes each incoming event as an input, and applies the filter conditions. In general it eliminates any event instance that does not meet these conditions.
- **Matching:** It finds patterns among events and creates sets of events that satisfy the same pattern. The matching step takes all events that have been left by the filtering step, and looks for matches between them, using an event processing pattern or some other kind of matching criterion.
- **Derivation:** It uses the output from the matching step to derive new events and set their content. The derivation step takes the matching set as an input and derives new events, applying derivation formulae to the events in the matching sets.

Event processing agents can also be classified into several types according to functionalities. Figure 3-3 shows the type hierarchy of different event processing agents.

- **Filter agent:** A filter agent has only the filter function and filters out irrelevant events with respect to a set of special filtering conditions.
- **Transformation agent:** A transformation event processing agent includes a derivation step and also optionally a filtering step. It transforms input events according to derivation formulae.
- **Pattern detect agent:** A pattern detect agent performs a pattern matching function on one or more input streams. It emits one or more derived events if it detects an occurrence of the specified pattern in the input events. For example it may detect the temperature increase when two certain temperature events are input, while the latter has a higher temperature value than the value in the previous event.

Transformation agents can further be divided into four sub-types according to transformation operators:

- **Translate agent**: A translate agent takes a single event as its input and generates a single derived event, which is a function of input event, using a derivation formula.
- **Aggregate agent**: An aggregate agent takes a set of events as input and creates a single output event applying aggregation function.
- **Spilt agent**: A split agent uses single event as input and creates a set of events, which can be a clone of the original event or a projection of the input event containing a subset of its attributes.
- **Compose agent**: A compose agent tasks groups of input events from two input streams, looks for matches using a matching criterion and creates output events. Each output event is a function of a collection of events taken from both input streams.

Translate agents contain further two sub-types:

- **Enrich agent**: An enrich agent takes a single event as input and extends the event with additional information.
- **Project agent**: A project agent takes a single event as input and removes information from this input event.

3.3.3 Event Consumer

An *event consumer* is an entity at the edge of an event processing system that receives events from the system [EtNi10]. An event consumer is also known as events sink and triggers certain type of actions, which indicates the reactive behaviors of the event processing system. Similar to event producers, event consumers can also be classified into three categories: hardware event consumer, software event consumer and human interaction event consumer [EtNi10].

- **Hardware event consumer**: Hardware event consumers are normally called actuators. An actuator takes an incoming event and reacts to it by performing a physical action, often in order to control something in the physical world, such as changing a magnetic field or producing an electrical signal.
- **Software event consumer**: The most common software event consumer is an event logger, which keeps a record of the received event, either in a flat file or a database. Further software event consumers can trigger certain workflows or create a new business process.
- **Human interaction event consumer**: People are also a kind of event consumer. Human interaction event consumers provide interfaces to humans, such as a visual display or alarm ring tons.

3.3.4 Event Interaction

One of the most important characters, which distinguish event-driven systems from traditional information systems, is the way of the information interaction between information producers and information consumers. In event-driven system the interaction of information (events) is based on Publish/Subscribe mechanism, which means information is pushed. While in traditional information systems using request-response mechanism, information is pulled.

According to [EtNi10], the differences between these two interaction models for event-driven system are:

- An event producer does not, in general, expect consumers to take any specific actions when they receive events that it has sent them.

- Events are often sent as one-way messages. In other words, after a producer has sent an event message, it can get on with other things without having to wait for responses**.**

Using Publish/Subscribe event interaction can result in reduced processing latency because the producer can send an event as soon as it has one to distribute. Many Internet applications are now based on the Publish/Subscribe model too. In general there is a trend of shifting web applications towards real-time web applications, which enable users to receive information instantly as soon as it is published by its producers, rather than requiring periodic updates.

3.4 Conclusion

Complex Event Processing (CEP) is a relative new technology, which allows users to perform traditional database and data-mining tasks like data validation, cleaning, enrichment, and analysis in real-time without persisting the data [Ober11]. With sensor and mobile devices proliferating in the modern world, we now have the ability to consolidate large volumes of sensor data about ourselves and the environment around us. In order to better comprehend and analyze these sensor data in real-time, CEP is the most appropriate technology to process the sensor data. By using CEP, query, filter, and transform data from multiple sensors for event detection can be done in real-time. Therefore, in this thesis we propose to use CEP to process real-time personal data on mobile devices in order to achieve an efficient solution.

4 State Of the Art

The research topic of this thesis covers several technologies, including Complex Event Processing, Mobile Computing and Semantic Web. In this chapter we will conduct a literature survey on the state-of-the-art technologies and theories that are the main focus of our work. The research work that is related to our approach mainly fit into the following research fields: mobile and distributed CEP system, efficient mobile processing, semantic-based pattern and sensor model, and mobile sensing system and e-fitness/e-Health mobile apps.

4.1 Mobile and distributed complex event processing system

In order to achieve efficient processing of real-time personal data detected by sensors on mobile devices, we propose to use Complex Event Processing (CEP) technology to process real-time sensor data. Considering that most real-time data are sensed by wearable sensors on mobile devices, processing the sensor data directly on mobile devices can be an efficient solution, which can reduce the superfluous data transmission to server. In this section we give a survey of existing work about event processing systems, and mobile/distributed CEP systems.

We start the survey from existing event processing systems. Within the last few years event processing has been developed rapidly with a number of academic and commercial projects. The most notable work is Esper and Etalis.

Esper [BeVa07] is an open source CEP engine. Esper uses Event Processing Language (EPL) to define patterns, which provides rich event conditions, correlation, possibly spanning time windows. The performance reports on Esper [Espe07] show very good performance. It is implemented in both Java and .NET, and can be integrated into other applications using Java or .NET.

ETALIS [AFRS$^+$10] is a framework for complex event processing and stream reasoning including a CEP engine. ETALIS uses ETALIS Language for pattern modelling supporting rule-based syntaxes and a formal declarative semantics. Besides event processing, ETALIS also provides functionality of stream reasoning. ETALIS is open source and is implemented in Prolog, where it provides a Java interface and can be integrated in Java programs.

Other academic research projects like Aurora [ACCC+03], Borealis [AABC+05], STREAM [ABBC+05], Cayuga [DGHR+05, DGPR+07], TelegraphCQ [KCCD+03, CCDF+03] use different methods to achieve complex event processing. The Aurora and Borealis systems are closely related systems. The goal of the both systems is to support various real-time monitoring applications. The both systems are based on “boxes-and-arrows” process and work-flow systems. STREAM processes both structured data streams and stored data together through queries, which are translated into flexible physical query plans. Cayuga is a general-purpose CEP system with a new implemented processing engine, which utilizes a variation of nondeterministic finite automata [DGPR+07]. TelegraphCQ is designed to provide event processing capabilities alongside relational database management capabilities by utilizing the PostgreSQL open-source database.

There are also many commercial CEP products on market including SAP SyBase Event Stream Processor (ESP)[12], Oracle Event processing[13] and IBM WebSphere Business Events[14]. However most of these engines are only capable of being deployed on powerful server systems and process events from business processes.

CEP technologies have been also used in distributed systems and mobile systems. In [BiER08], the authors have created a distributed correlation event processing network by integrating several centralized CEP systems. The main issue of this work is the configuration of connecting the correlation nodes, especially input and output nodes, and deciding which correlation tasks should be deployed at which node.

In [SKPR12] DHEP project (Distributed Heterogeneous Event Processing) is described. The project has been used to develop a distributed CEP system for industry. It developed a framework with DHEP Meta language and configuration tool to realize interoperability between heterogeneous CEP engines that are deployed in different system nodes.

DiCEPE (Distributed Complex Event Processing) [PHRM+12] is a platform for distributed CEP system. It provides the functionality to integrate different CEP engines. Meanwhile, it also provides a native support for various communication protocols in order to federate CEP engines and ease the deployment of complex systems-of-systems.

Afore mentioned three approaches [BiER08, SKPR12, PHRM+12] are capable of being used in large scale distributed systems. But they don’t consider the characters of mobile systems and are difficult to be used in mobile environments.

The most relevant work to our research is [DuBS13, StBD13], which describes an approach for using CEP to process sensor data on smartphone for AAL (Ambient Assisted Living) monitoring. It uses Esper on smartphones and processes the data from smartphone sensors correlatively to achieve situation awareness and context awareness. In comparison to our approach, however, it uses only the sensor data from smartphone without using events from external event sources or

[12] www.sybase.com/products/financialservicessolutions/complex-event-processing
[13] www.oracle.com/technetwork/middleware/complex-event-processing/overview/index. html
[14] www.ibm.com/software/integration/wbe/

additional context, such as domain knowledge base. It provides no adaptivity for availability changes of sensors and workload changes on smartphones.

Another interesting work is [MiLI11]. It developed an energy efficient continuous event processing system on smartphone. Instead of receiving all sensor data, the system uses a pull mechanism to only receive the data required by event processing, regarding the communication cost and properties of sensor streams. This work focuses only on energy efficiency without using external information or providing context aware monitoring.

Another related work is described in [KLCC+09]. The authors have introduced an embedded CEP system in mobile device, combining DDS (data distribution service) to provide the mobile patient healthcare service to users. The work uses one CEP engine on mobile device and provides data processing for multiple applications. It provides a function to dynamically switch the processing rules for different applications. However comparing to our approach, it doesn't consider the workload of mobile devices and the additional context of users or environment is not used in processing.

There are also other prototypes of using CEP in mobile environment, such as HARMONI [MEJM06], which describes a middleware for remote monitoring on mobile phones using CEP, and personal health monitoring system as described in [MPES09], which uses smartphones to monitor patients, while it uses CEP to process the sensor data in backend server.

To sum up, although there are some existing approaches for mobile CEP systems and distributed CEP system, however, most of them provide no context-aware data processing and the processing capacity limitation of mobile devices is also not considered, while the both are key features of our approach.

4.2 Offloading in mobile computing

In spite of the fast innovation in such key components as processor, memory and wireless technologies, it remains widely agreed that mobile applications are still confined by the limited computation capability of mobile devices. These limitations [FoZa94] are major impediments to providing reliable and sophisticated mobile applications to meet the real-time data processing requirements. Therefore efficient mobile processing should be achieved regarding the existing resource restrictions on mobile devices. In this section we provide a survey of existing approaches for achieving efficient mobile processing.

The original idea was to use an additional remote server to solve the problem of resource restrictions. [MGBM+02] describes an approach of using distributed platform to transparently offload portions of a service from a resource-constrained device to a nearby server. The authors developed a prototype and an emulator to evaluate this approach. The results revealed that using additional server can solve the problem of memory and processing constraints of mobile devices. Another similar approach is [WoPM99]. The authors use remote server to offload the computation for Java-based mobile applications.

After the rise of Cloud Computing, the idea of using clouds instead of traditional servers to support mobile applications has become more attractive. Mobile Cloud Computing (MCC) [QuAR11, Cox11] has been regarded as the best solution to solving the problem of resource limitations on mobile devices. Basically MCC combines the strength of clouds and the convenience of mobile terminals, while clouds provide high performance computation ability and large storage and mobile devices provide their mobility and usability. Therefore the processing tasks of mobile devices can be offloaded to cloud infrastructure.

In [MZZW+13] the authors introduced two methods of using cloud service to support the mobile applications to achieve the distributed mobile processing. The first method uses cloud as computation proxies for offloading, which means mobile applications move all their computation in cloud and use mobile devices only as display. Another is service partitioning for computation offloading, which divides computation into two parts and offloads only one part to clouds.

In [KuLu10] the possibility and methods of using mobile cloud computing to realize energy saving for mobile devices are discussed. The analysis of authors shows that using mobile cloud computing can potentially save energy for mobile applications, but not for all of them.

There is already some existing work using mobile cloud architecture to improve the computation performance or saving energy. MAUI enables fine-grained energy- aware offloading of mobile codes to a cloud based on a history of energy consumption [CBCW+10]. CloneCloud [CIMN+11] uses function inputs and an offline model of runtime costs to dynamically partition applications between a weak device and the cloud. In [FaMH13] the authors provided API for mobile device clouds and evaluated the computation offloading of mobile application focusing on energy and time. [SLAZ12] has introduced another extreme of mobile cloud computing, which the remote computation resources are also mobile devices instead of traditional high performance server.

Cloudlets are a new technology, which provides another possibility to achieve the computation offloading for mobile computing. The traditional cloud computing mobile apps require WAN latency, which makes apps insufficient for real-time applications. The cloudlets technologies [Saty10, FGNW12] enable the cloud computing technologies helping mobile devices to overcome the resource limitation [VSDD12]. To cope with high computation requirements of mobile computing cloudlets use trusted, resource rich computers in the nearby vicinity of the mobile user (e.g. near or co-located with the wireless access point in LAN). Mobile users can then rapidly instantiate custom virtual machines (VMs) on the cloudlet running the required software in a thin client fashion [SBCD09]. However the cloudlets technology depends heavily on additional hardware infrastructure in each LAN, which needs time and investment to upgrade the current mobile infrastructure.

In summary, similar to our approach, in order to solve the problem of resource limitations on mobile devices, additional infrastructure is required, which can be remote server, cloud infrastructure or cloudlets servers in local networks. However, the existing approaches are not applicable to event-driven systems and most of them are not dynamically adapted to current workload of the mobile device.

4.3 Semantic pattern and sensor model

As introduced in the chapter 1, we propose to use semantic technologies to manage patterns and event sources (i.e. sensors), in order to achieve dynamic pattern management. In this section we give an overview of existing ontologies for event patterns and for sensors.

There are almost no existing ontologies regarding both event/pattern aspect and sensor aspect.

In terms of event pattern ontology, the most related work is [KhSt09] and [SeSt10]. In [KhSt09] the authors present an ontology for modelling events for policies and rules for the compliance management. An event can be either used for input or output. Each Event can be implemented either as complex event or event stream. An event can be an event expression which combines events using event operators. Complex events can be a combination of event occurrences or event patterns. [SeSt10] presents a meta model for events and patterns regarding event types and event sources. Each complex event presents a pattern, which can be a combination of events, complex events and event operators. Both models share the concepts event, event operator, complex event and event pattern. However both models define event sources on an abstract level and provide no event source management. In contrast to these ontologies, one of our goals with our pattern ontology is to manage event sources for events used in the patterns.

The SARI event model presented in [Rozs08] focuses on event types. Event types are hierarchical structures containing an arbitrary number of event attributes. Each event attribute has an attribute type. Possible attribute types are collection types, dictionary types and single-value types. Single-value types are the basic types such as string or integer. The event types in SARI are managed as libraries which should be valid in the given event processing system. SARI aims to share events and not event patterns among a set of CEP related applications. However SARI provides no patterns definition in model and doesn't describe the relationship between event sources and events.

In terms of sensor ontology, the most famous model is SSN (Semantic Sensor network) ontology [CBBG+12], which is the most widely used ontology in semantic sensing domain. It defines an ontology using OWL2 [MPPB+09] to describe the capabilities and properties of sensors, the act of sensing and the resulting observations. The SSN ontology supports the description of the physical and processing structure of sensors. The sensors modeled in SSN are not only physical sensing devices, but also anything that can estimate or calculate the value of a phenomenon, such as a device, a computational process or a combination of devices and processes could play the role of a sensor. Taking advantage of this existing sensor ontology, we use SSN ontology also in this thesis, combing with our own development for event source description.

Beside SSN ontology there are also many other sensor ontologies, which are developed for different purposes. We introduce some of them afterwards.

SensorML [RoBo06] is Sensor Model Language. It specifies models and XML encoding that provide a framework within which the geometric, dynamic, and observational characteristics of sensors and sensor systems can be defined. The SWAMO Ontology [WSSM+08] is designed to enable dynamic, composable interoperability of sensor web products and services. It describes autonomous agents for system-wide resource sharing, distributed decision making, autonomic operations. The ontology is compatible with SensorML. ISTAR [GPJD+08, PGDV+08] is a set of ontologies, which model sensors, mission tasks and deployment platforms. The collections of types of sensors can be recommended for a particular task by using semantic reasoning. Ontonym [SKDN09] has defined a set of ontologies including time, location, people, sensor, provenance, events and minor ontology. Ontonym's sensor ontology is concerned with the description of sensors and the data they generate. It provides a high level description of a sensor and its capabilities (Frequency, Coverage, Accuracy and precision pairs). SENSEI [BaMP09] is based on OGC SWE Observations & Measurements (O&M) model and NASA's Sweet ontology[15]. It combines the information model for Observations and Sensing from OGC O&M and the units of measurement model from Sweet ontology. In additional it also contains domain knowledge and model of resource and entity.

In conclusion, there are already many existing approaches for sensors and patterns, however, in the existing approaches the pattern models and sensor models are separated and there are no existing approach connecting both sensor and concepts. While our approach combines both models through event resource concept to achieve dynamic pattern management considering the sensors, which are the event sources of most events used in patterns.

[15] http://sweet.jpl.nasa.gov/

4.4 Mobile sensing system and applications

There is much work in the domain of remote personal monitoring that can be related to different types of the psychophysiological monitoring (e.g. heart rate, hear rate variability, skin conductance) in order to determine some parameters of the user's healthy status (e.g. fitness level, stress level, cardio arrhythmias, …). In this section we present only the most important systems related to using smartphones and wearable (nonintrusive) sensors.

Recently, much research effort is put into more sophisticated approaches of using smartphone sensor data for personal health or physical activity monitoring. Furthermore, a large number of mobile applications are available in popular app stores.

From a scientific perspective, as mentioned in the previous section, [DuBS13] introduces a prototype of combining smartphones and sensors in AAL domain. The authors use CEP on smartphone to achieve real-time sensor data processing. The UbiFit Garden [CMTC+08], a joint project between Intel and the University of Washington, captures levels of physical activity and relates this information to personal health goals when presenting feedback to the user. Another smartphone application is presented by [LMLY+12], which is used to monitor wellbeing. The authors propose an application which consumes real-time data from different smartphone sensors in order to compute an overall score of wellbeing in several dimensions such as social activity and sleep. Compared to our work, the main difference is that these approaches [DuBS13, CMTC+08, LaMo12] do not make use of external sensors and do not allow the manual definition of interesting situations by taking into account temporal and spatial operators provided by complex event processing. Another approach, the BALANCE system proposed by [DACH+12], aims at encouraging healthier lifestyles to its users. This approach is especially interesting as it combines data entered manually (about food eaten) and sensor data in order to compute the caloric expenditure of a user. However, the authors only investigate one aspect of personal health monitoring, while today's smartphones offer the opportunity to monitor a much wider range of habits and we argue that especially the combination and aggregation of heterogeneous data is promising.

From a practical perspective, applications like the Nike+ ecosystem[16] are becoming more popular. Nike+ combines several available hardware sensors with a smartphone application and a community, allowing users to track sports activities in real-time and to compare themselves with other people through an online community. Even if this is already a very interesting application, the potential of such combined applications is much bigger, as Nike+ currently relies on simple metrics (such as NikeFuel, an overall fitness score or burned calories during running).

[16] http://www.nike.com/de/de_de/c/nikeplus-fuelband

Other very interesting apps are: SuperBetter[17] helps to achieve predefined health goals (or recover from an illness or injury) by increasing the personal resilience of the user. It is a gamification-based app that helps people to be more motivated to do physical exercises.

ShapeUp[18] is an app that helps co-workers to compete, support, and challenge each other with a shared goal of better health. It supports three steps: 1) tracking progress: it enables to set goals, track progress, compare results with peers, and share success stories; 2) finding supporters: it searches the network and browses profiles to find colleagues to support the user and 3) joining fun challenges: it finds the challenges the user will like, invite the colleagues, and spread good health.

Daily challenge[19] is another gamification-based app that helps in improving physical activity. As an example of the traditional systems we mention CardioNet[20], the leading provider of ambulatory, continuous, real-time outpatient management solutions to monitoring relevant and timely clinical information regarding an individual's health. CardioNet's initial efforts are focused on the diagnosis and monitoring of cardiac arrhythmias, or heart rhythm disorders, with a solution that it markets as Mobile Cardiac Outpatient Telemetry™ (MCOT™). Through its Cardio core division, CardioNet offers core lab services to pharmaceutical and devise sponsors as they provide new products through the development process. However, while wearable are better than 24h medical holters[21], these sensors are quite intrusive comparing to sensors that have been recently promoted on the market.

From the sensing point of view, the most advance model is Zypher BioHarness™ 3 sensor[22] and its applications that enables the capture and transmission of comprehensive physiological data on the wearer via mobile and fixed data networks – enabling remote monitoring of human performance and condition in the real world. BioHarness™ 3 has applications in any fields that require high-level wireless and remote physiological monitoring, including research, training and tele-health situations. The most important features are very complex measurements, including Heart Rate, R-R Interval, Breathing Rate, ECG, Posture, Activity Level and Peak Acceleration.

In summary, the most existing mobile sensing applications provide only simple processing functions using static rules and have no context-awareness. Some research approaches also use CEP technologies to process real-time sensor data. However they don't provide adaptation to current situation of users and don't consider the resource limitation of mobile devices as well.

[17] https://www.superbetter.com/
[18] https://mywellvolution.shapeup.com/
[19] http://meyouhealth.com/daily-challenge/
[20] http://www.cardionet.com
[21] http://en.wikipedia.org/wiki/Holter_monitor
[22] http://www.zephyr-technology.com/products/bioharness-3/

PART II

Efficient Context-aware Real-time personal data Processin

5 The Need for Context-aware Real-time Personal Data Processing

In this chapter we describe why having innovative software architecture for real-time personal data processing is necessary for mobile applications. We deduce the necessity of the context awareness, resource awareness and data collaboration by analyzing the issues in the current mobile technologies and existing approaches. We conclude by defining the requirements for developing the proposed approach.

5.1 Limitations of mobile devices and wireless sensors

As introduced in Chapter 2, the technology of mobile devices has been advancing rapidly since last decade. The modern mobile devices have been increasingly penetrating into people's lives, as they bring many advantages that raise people living standards, such as

(1) high mobility, where users are able to carry and use mobile device wherever they go and whenever they want;
(2) high usability, where the small size of the modern mobile devices enables most devices to be held in one hand and the touch-screen provides simpler and easier operations instead of using keyboard and mouse,
(3) more entertainment, where the integrated camera, high resolution display and touch-screen bring more entertainment and social usage than the traditional PC system, and
(4) the numberless mobile applications: this brings unlimited possibility to mobile devices.

While it seems that mobile devices can do anything what human beings need, there are still issues with mobile devices, which restrict the possible exploitation of the potential of mobile devices.

The first issue is the limited computation resources on mobile devices. Although the hardware of mobile devices has been rapidly innovated and their performance been exponentially improved. The first generation iPhone used only a 412MHz single core ARM processor and 128MB memory, while the latest Samsung Galaxy S5 has a 2.5GHz Quad-Core processor and 2GB memory, a 10-fold improvement. However, the processors found in mobile devices are still a lot slower and far behind the processors used in the desktop or server systems. According to the benchmark study in [Rahu13] a standard-voltage Intel Core i5 laptop processor is about 8 times faster than the best mobile device processors. Hence, in comparison to the desktop and server systems, mobile devices can only provide limited computation capability.

Of course, the computation capability (mainly the processor performance) of mobile device has historically increased at a rapid pace and it is promising to have huge improvement in the near future. But another key issue of mobile devices restricts the computation capability potential of mobile devices: battery capacity. In comparison with the development of other hardware, such as processor, memory and communication bandwidth, as the main power source of most mobile devices, battery has seen relatively slow improvement in the past decade. Battery capacity is growing only 5 percent annually [Robi09]. Because the high usage of processor and memory lead to quick battery drain, battery capacity has become a major impediment for mobile devices to provide high computation capability for reliable and sophisticated real-time data processing.

Hence, there is a gap between the limited (although ever increasing) computation capability of mobile devices and unlimited application potential of mobile devices. On one hand the huge amount of real-time mobile data processed by data-intensive mobile applications require more computation capabilities. Additionally, the development of mobile devices is going very strongly in the direction of having more and more "features", like new sensors, new specialized cameras, new software for gaze-driven control, etc., which require more computation capability and meanwhile generate more data, which increases the need for an extended processing. On the other hand although the fast innovation in key components of mobile devices such as processor, memory and wireless technologies, it remains widely agreed that mobile applications are still confined by the limited computation capability of mobile devices.

Since the purpose of this thesis is to process the personal data and environment data of users in real-time, considering the flow of huge real-time data going through the system at any given points in time, the proposed system is also required to have high computation capability available to undertake the processing tasks for huge amount of real-time data. This may not always be feasible in most mobile devices. Therefore, it is necessary for the proposed system to extend the computation capability of mobile devices to meet the requirements of processing huge amount real-time data by off-loading the processing tasks from mobile devices to remote infrastructure such as backend server or cloud infrastructure.

In addition, the modern mobile device operating systems such as Android are typical multi-task operating systems, which execute multiple applications simultaneously. Besides the foreground applications, it also executes many other applications in background such as telephone, messaging and etc. The high usage of computation resources may affect the execution of other mobile applications, which should be usually avoided. Hence, the computation capability of mobile devices should be extended for personal monitoring application, for example by using remote infrastructure. The number of the processing tasks assigned to user's mobile device should be limited based on the current available resources on user's mobile device.

The battery capacity is also a key issue for wireless sensors, since the battery is also the main power source of most wireless sensors. The limited battery capacity restricts the life time of wireless sensors. Furthermore, the remaining battery volume of the most wireless sensors, which are not equipped with a display panel, is normally not visible. Hence, in practice situations often arise where the battery of a wireless sensor is exhausted during the application execution and the wireless sensor will be disconnected.

Another important issue for the wireless sensors is the quality of the wireless communication. Currently the most wireless sensors are connected with mobile devices through personal area network (PAN) [Zimm96], commonly Bluetooth [BSIG07], IrDA [IrDA08] and ZigBee [ZiAl06]. The communication using these techniques can be affected and has limited distance range. If the wireless communication is interfered or the distance between the mobile device and the sensor is out of the communication range, the wireless sensor will disconnect from the mobile device.

In the case that the sensor events provided by such wireless sensors are used in the processing tasks, the disconnection of the wireless sensor leads to failure of the processing tasks. The proposed system should adapt to such situations in order to avoid the failure of the processing tasks.

5.2 Real-time personal data processing

The innovation of sensor technology enables ordinary people to obtain their own personal data and environment much more easily. The new generation mobile devices combining the integrated sensors and new wearable sensors provide a new possibility of using personal data to monitor people's health status. Due to the rise of the attention on the life quality especially the health status, more and more mobile applications, which provide health monitoring by processing personal data, appear on the market.

Most of the mobile applications in the m-Health domain or m-Fitness are oriented toward the detection of the particular real-time situations, which are relevant for users. As for the current technologies for processing real-time stream data, Complex Event Processing (CEP) is the most suitable technology. The advantages of using CEP to process real-time personal data can be summarized as following:

- **Real-time data processing with low latency**: Sensors of mobile devices produce enormous amount of mobile personal data and environment data continuously. CEP analyzes event data in real-time to generate immediate insight and enable instant response. It is designed to efficiently handle streaming input data directly in memory, which enables real-time operations with low latency.

- **Correlating heterogeneous data**: Most personal data are detected by different sensors. Normally each piece of information is fine-grained containing only little information and is difficult to interpret. Thus, single personal data streams have to be correlated with other data streams to gain meaningful insight. CEP enables the continuous correlation of streaming data from heterogeneous sources [BrDu10].

Although there are already some existing mobile applications for personal monitoring on the market, the mainstream mobile applications have the same issue: the real-time personal data are not integrated with other data, especially the user's data, and the processing method is almost fixed and provides no changes for the different user's situations.

In fact the personal data is difficult to interpret without additional information or in the context of specific situations. People are different and even there were two completely identical people in the world, the same personal data may have different meanings to each individual. For example, the heart rate data, which is over 120 BPM for more than ten minutes, may be normal for a professional sportsman, but it may indicate a hazardous situation for a regular individual as it may be a symptom of heart diseases. In order to interpret the real-time personal data correctly, additional context, especially the user's data should be integrated with the real-time data.

Furthermore, even for the same person, the same personal data should also be interpreted differently under different circumstances. For example, the heart rate data of 120 BPM can be normal for a person running, but for this person being sleeping in the bed; such a high heart rate may indicate some symptoms of some diseases. Obviously the current situation of a user plays a critical role in the real-time personal data processing. Different processing methods should be used to dynamically adapt to the different situations.

Therefore, the context awareness should be achieved for the real-time personal data processing with integration of user's context and additional domain knowledge, such as medical domain knowledge or sport domain knowledge, and the processing method should be dynamically adapted to the current situations of user.

The integration of user's context and using additional domain knowledge ensure the real-time personal data to be processed correctly and results are insightful and meaningful. So far the personal data processing has focused on one individual only, i.e., the personal data of each user is processed separately. However, for some special cases, the result cannot be readily inferred from the personal data; rather the collaborative data processing is required. Fortunately, the ever increasing number of users of mobile applications provides just such opportunities for collecting data among most alike users.

5.3 Requirements

Based on the analyses described in the previous sections and the research questions defined in Chapter 1, we conclude that the following requirements are necessary for developing innovative mobile applications for real-time personal data processing in various domains, especially for m-Health and m-Fitness domains.

- **R1**: An event-driven software architecture, which enables to use Complex Event Processing (CEP) on mobile devices to process real-time personal data. (Q1)
- **R2**: An innovative software architecture for mobile applications, which enables extension of the computation capability of mobile devices for real-time personal processing. (Q2)
- **R3**: Resource-aware processing tasks distribution, which ensures stable and good user experiences of mobile devices without affecting the performance of other applications and mobile device system. (Q2)
- **R4**: Context-aware personal data processing, which enables the integration of user's context and additional domain knowledge in personal data processing and achieves dynamically adaptation to current situation of user. (Q3)
- **R5**: Real-Time data collaboration, which enables the data collaboration between different users and provides methods to find the most suitable users for the collaborative data processing. (Q4)
- **R6**: Dynamic adaptation to the real-time changes in the event streams availability. (Q5)

In the subsequent chapters of this part we describe the details of our approach based on the defined requirements.

6 Event-driven Hybrid System Architecture

In this chapter we introduce the software architecture for realizing the mobile application described in our approach. Considering the limitation of mobile devices, the proposed event-driven hybrid architecture combines mobile devices and backend server infrastructure to extend the performance of real-time data processing. The publish/subscribe middleware is used for real-time event exchange between mobile devices and backend server. We also develop an event and stream model for the proposed system to implement event-driven software architecture.

6.1 Architecture overview

Taking advantage of popularity and quickly developed mobile and sensor technology, mobile devices are enabled to collect much personal information and environment information of user in real-time. Hence there is a significant increase in the mobile applications, which deal with huge amount of real-time mobile data, like in the case of m-Health and m-Fitness. Many of these applications are oriented toward the detection of the particular real-time situations of users which brings them to the domain of Complex Event Processing (CEP).

However, there are several issues for the development of mobile application for real-time personal data processing. Firstly, the limitation of resources on mobile devices [FoZa94], especially the computation resources and disappointing battery capacity [Robi09] of mobile devices are a major impediment to providing reliable and sophisticated mobile applications to meet the real-time data processing requirements. Secondly the existing processing methods of mainstream mobile applications pay less attention to the integration of real-time data and other data, especially the user's context, which lead to isolate and static processing of real-time personal data.

According to the requirements R1 and R2 defined in 5.3, in order to process the real-time personal data in an efficient and effective way, we propose an event-based hybrid software architecture for developing a new generation of real-time personal data processing mobile applications. The proposed software architecture uses CEP to process real-time personal data for efficient data processing. It combines mobile devices and backend server infrastructure, using the idea from mobile cloud computing [QuAR11, Cox11], to extend the processing capability of mobile devices. By using semantic technologies it integrates user's context with personal data in real-time.

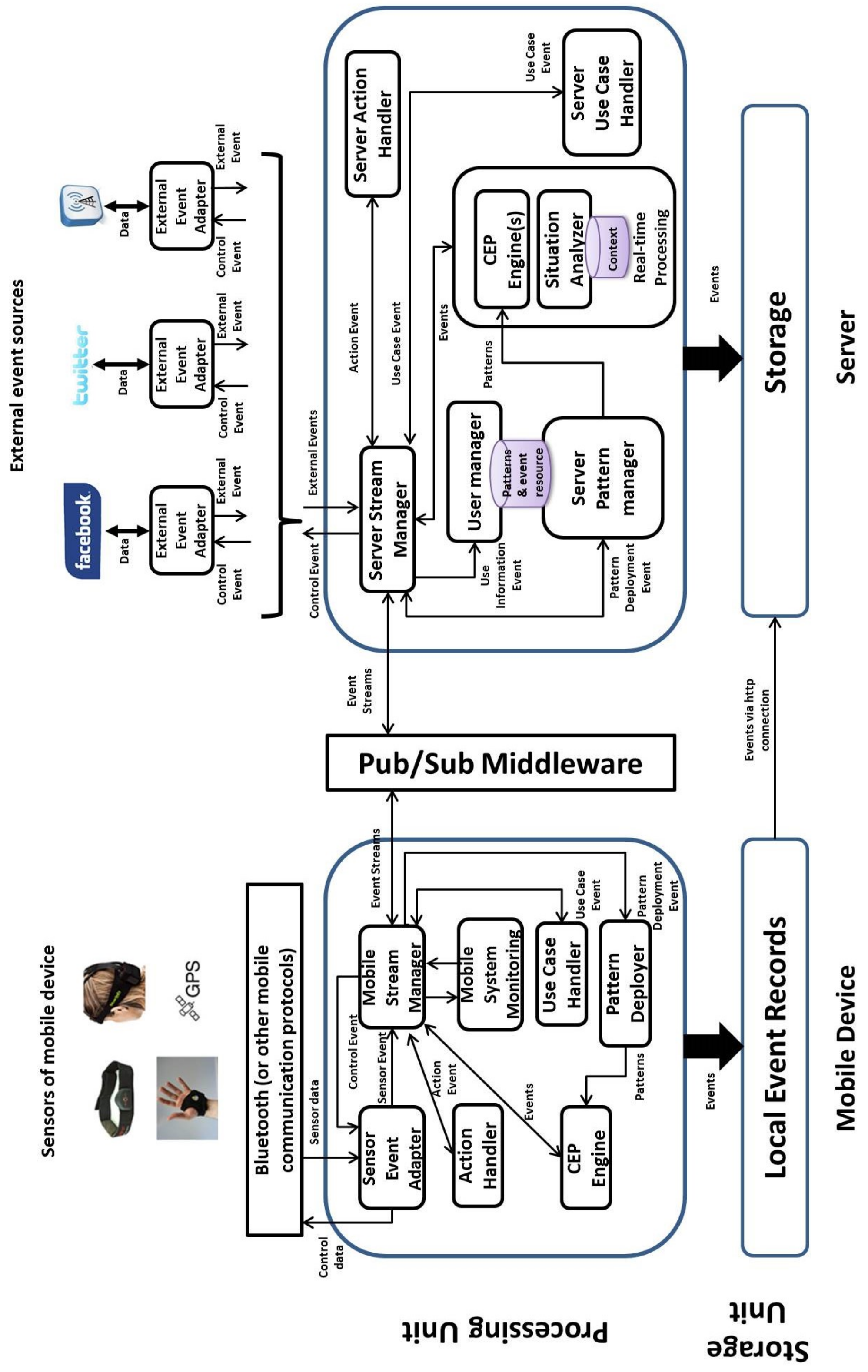


Figure 6-1 Conceptual architecture for event-based hybrid system

Because the key feature of the proposed system is Mobile-based Complex Event Processing (CEP), we name the proposed system **MCEP system**. Figure 6-1 shows the conceptual architecture of the proposed MCEP system. The whole system is event-driven and all components are loosely coupled through events.

The whole conceptual architecture consists of two parts: mobile device part and backend server part, corresponding to the mobile device and backend server infrastructure as mentioned early. Each part has two units: processing unit and storage unit, containing several components.

The processing units are in charge of processing the real-time data. They process the real-time personal data dynamically to detect the situations of interest for a user according to the current situation of user and user's context.

The storage units store the events including real-time events and complex events. The stored events can be used for off-line analysis or as historical information of users, which is used as user's context in the future data processing. In order to avoid the high communication workload caused by events transmission from local storage unit to server storage unit, the storage units upload events via http in high speed network environment, e.g., WIFI.

In the following, we introduce the functionality of each component in brief.

Mobile Device Part

On the left side of the figure is the mobile device part. The software of this part is realized as mobile application and will be installed and executed on user's mobile device. As shown in the figure, the processing unit of the mobile device part consists of seven components.

- **CEP Engine**: This is the core component of the processing unit on the mobile device part. It provides the key function of the processing unit: processing the personal events on mobile devices in real-time. Considering the characteristics of mobile devices and the operation system of mobile devices, it should be a light weight CEP engine, which doesn't cause high workload on mobile devices. Hence in our prototype, we use Android-based Esper engine [Bade10, Espe07] to realize the real-time personal data processing in this component.
- **Sensor event adapters**: The modern mobile devices can obtain information from many sensors, including the mobile devices integrated sensors and the external sensors, which are connected to mobile devices through different communication protocols, such as Bluetooth [BSIG07] or infrared [InDA99]. However, the information sensed by these sensors is represented in different data formats and cannot be directly used in the proposed MCEP system. Hence, sensor event adapters are required, in order to translate the information sensed by sensors into the event format, which is used by the proposed MCEP system. In

addition, sensor event adapters also provide the function of controlling sensors, in the case that sensors are controllable.

- **Mobile stream manager:** Considering the event-driven architecture used in the system, events are not only used to transmit personal information to CEP engine, but also used to couple different components and to exchange information among the components. Hence, besides sensor events, there are also many other events used in the proposed MCEP system, such as control events like pattern deployment events or user information events. Different events are required by different components and should be sent to different targets. In order to achieve this functionality, event stream manager connects all components in the processing unit, receives all events from different components and dispatches the events to their targets. Event stream manager sends the events according to the stream of the events for a flexible dispatching (events and streams in MCEP system is described in section 6.3). Event stream manager provides also the Publish/Subscribe function, which enables the mobile device part (mobile application) to exchange real-time events with backend server through publish/subscribe middleware.
- **Mobile system monitoring:** Considering the limited resource on mobile devices, we propose to use hybrid architecture to extend the computation capacity of the MCEP system and use resource-aware pattern distribution (described in Chapter 8) to achieve efficient processing tasks assignment. Therefore, the information about the real-time workload of mobile device is required. The mobile system monitoring component monitors the information of the resources on mobile devices, including CPU usage, memory usage and available battery volume, for the workload calculation.
- **Pattern deployer:** This component is tightly connected to CEP engine and provides the function of updating the current active patterns. Pattern deployer deploys and undeploys the patterns in CEP engine in real-time. It also integrates the pattern translator, which translates the pattern from standard format into the special pattern language used by CEP engine (e.g., in the prototype, Esper CEP engine is used and all received patterns are translated into the Esper EPL language). In addition, there is also a simple repository for the translated patterns to store all the received patterns. In case that the required pattern was used in the past, the repository can provide the translated pattern directly to CEP engine without downloading and translating the pattern.
- **Action handler**: This component performs the actions such as playing alarm sound and displaying recommendations on mobile device in order to interact with users. The actions are triggered by assigned events, which are normally complex events and present some situation of interest.
- **Use case handler:** A mobile use case contains a set of settings for the user, including event definition, stream definition, stream assignments, action assignments and local pattern def-

inition. **Use case handler** manages all pre-defined mobile use cases and provides the function to start and stop use cases according to the different situations of users.

Backend Server Part

On the right side of the figure is the backend server part. The software for this part can be implemented as a standalone application or web service and can be deployed either on a server or on cloud based infrastructure. The processing unit of backend server part consists of seven components and several knowledge bases.

- **Real-time processing:** This is the core component of the backend server. It consists of two subcomponents and several knowledge bases.
    - **CEP engine:** This is the CEP engine on the backend server. It provides the additional processing capacity to support the CEP engine on mobile devices.
    - **Situation analyzer:** This component provides the dynamic adaptation to real-time personal data considering the user's current situation and taking user's context and additional domain knowledge into account. According to the monitoring procedure defined in Monitoring Goal Network (MGN, described in Chapter 7), situation analyzer performs different actions, including deploying patterns to adapt the current situation of user based on user's context and related domain knowledge.
    - **Context knowledge bases:** They provide information, such as user's static context, user's historical records and special domain knowledge like health and fitness, for the situation analyzer to achieve dynamic adaptation.
- **Server Stream manager:** This component is in charge of the communication of the whole backend server. Externally, it communicates with the mobile applications of users through publish/subscribe middleware and receives the events from the external event sources through external event adapters. Internally, it receives the events from all components and dispatches the events to different targets. It also performs the function of **event transfer,** which enables to transfer the copy of an event to a user for data collaboration.
- **External event adapters:** This component is used to receive the information from external event sources, such as social networks like Twitter and Facebook, public information providers like weather station, and commercial information providers. External event adapters also translate the information to the MCEP event format and send the events to server stream manager. Different external event adapters can use different techniques depending on the external event sources, which provide information.
- **User manager:** The backend server is designed to provide service for multiple users simultaneously. User manager is in charge of managing the information of different users, who are connected to the backend server. It manages the event resources of each user and up-

dates the information of event resources in pattern and event resource ontology during run-time. It also stores the dynamic context of users for situation analysis. Each user will be registered in the user manager, as soon as the mobile device part (mobile application) is connected to the backend server (i.e., sending registration event to the backend server). It also provides the function of relevant user search, in order to find the most suitable users for data collaboration.

- **Server pattern manager**: This component is in charge of the management of patterns, including query, adaptation, distribution calculation and dispatching. It receives the id of a pattern, which is required to be deployed according to the processing result by **situation analyzer** and retrieves the pattern from the pattern and event resource ontology by querying the received pattern id. It provides pattern adaptation according to the currently available event resources of the user (the pattern and event resource ontology and pattern adaptation are described in Chapter 9). In order to achieve efficient pattern distribution, the server pattern manager uses resource-aware dynamic pattern distribution algorithm (described in Chapter 8) to calculate the optimal event resource assignment and deployment position selection. It is also responsible to deploy patterns to MCEP engine on the backend server.
- **Server use case handler:** This component manages the server use cases for different users. A server user case indicates the monitoring goal network (MGN) used by the user and defines also required events and streams. A server use case also provides a possibility of using some native codes to process the personal data for some special situations, which are difficult to be modelled as patterns and detected by CEP. Server use case handler manages all pre-defined server use cases and starts the use cases as soon as the user is registered by the user manager.
- **Server action handler:** Similar to the action handler on the mobile device part, the server action handler performs pre-defined actions on the backend server, when the assigned trigger event is received.
- **Pattern and event resource ontology:** This ontology stores all patterns that are required by monitoring procedure defined in MGN. It also stores the event resources of each user and updates the information of event resources in real-time. This ontology is used by the server pattern manager for pattern distribution. The details of this ontology are described in Chapter 9.

In the middle of the figure is the publish/subscribe middleware, which exchanges real-time events between mobile devices and backend server through publish/subscribe mechanism. The details of the publish/subscribe middleware are described in the section 6.2.

In order to validate the design of above described conceptual architecture, we have implemented a prototype of MCEP system including all components of the processing unit. The prototype was used to show demos at different conferences and used to evaluate our approach of MCEP system.

6.2 Communication middleware

As mentioned in section 3.3.4, one of the most important characteristics, which distinguish event-driven systems from traditional information systems, is the publish/subscribe mechanism. The publish/subscribe mechanism ensures the event communication between different components in real-time. In the proposed MCEP system, the communication middleware provides the publish/subscribe function to enable the real-time event exchange between mobile device part and backend server part.

As stated in the Chapter 1, the proposed MCEP can be used in various use cases. Considering the requirements of possible use cases, the communication middleware is required to realize the following functions:

- Basic topic-based publish and subscribe function.
- Single user communication: the communication middleware should be able to transmit events from the backend server to a certain user.
- Broadcast communication: the communication middleware should be able to transmit events to all users.
- Group communication: the communication middleware should be able to transmit events from the backend server to all users in a certain group.

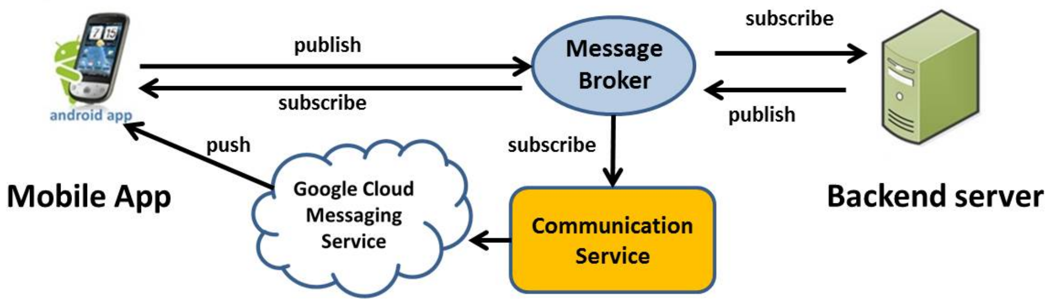


Figure 6-2 Publish/subscribe middleware

In order to achieve requirements of the communication defined above, we design the publish/subscribe middleware and communication process for the MCEP system. Figure 6-2 illustrates the concrete publish/subscribe data flow and components of the communication middleware. The whole communication middleware consists of three parts: message broker, communication service and Google Cloud Messaging service (**GCM**[23]).

Message broker: it is the core of the communication middleware providing publish/subscribe function. It receives the events from different connected clients and sends the events to clients, who subscribes to the events. There exist many different implementations of message brokers, including Apache ActiveMQ[24], Apache Apollo[25], Open AMQ[26], JBoss Messaging[27], IBM WebSphere Message Broker[28], SAP PI[29], Play DSB[30], etc.

In implementing the prototype, the ActiveMQ JMS server is selected to be used as message broker. ActiveMQ is an open source message broker written in Java together with a full Java Message Service (JMS) client. Comparing to other implementations mentioned above, ActiveMQ receives events from mobile devices using MQ Telemetry Transport (MQTT) protocol [Lock10], which is a publish/subscribe based light weight messaging protocol for use on top of the TCP/IP protocol. Because MQTT is designed for connections with network bandwidth limitation, it is used in the communication between mobile device part and publish/subscribe middleware of the proposed MCEP system.

Google Cloud Messaging service (GCM): Google Cloud Messaging for Android (GCM) is a service that allows the system to send data from server to users' Android-powered device. The GCM service handles all aspects of queuing of messages and delivery to the target Android application running on the target device. Since the GCM service is integrated in Google Service and can be easily used by Android mobile applications, it is used to push events to mobile device part of MCEP system.

Communication service: it connects the message broker and GCM service and realizes the required communication for MCEP system, including single user communication, broadcast communication and group communication. It subscribes the events published by the backend server and pushes the events to users, who subscribe to the events or are required to receive the events, using google GCM service.

[23] http://developer.android.com/google/gcm/index.html
[24] http://activemq.apache.org/
[25] http://activemq.apache.org/apollo/
[26] http://www.openamq.org/
[27] http://www.jboss.org/jbossmessaging
[28] http://www-03.ibm.com/software/products/en/ibm-integration-bus
[29] http://scn.sap.com/community/pi-and-soa-middleware
[30] http://www.play-project.eu/

6.3 Event and stream model

Since the MCEP is an event-driven system, events play an important role in the whole system. Events are not only used to transmit the information, but also used to control the system. An event stream denotes a sequence of events, which have the same characteristics. In this section we explain the event and stream model used in the proposed MCEP system.

6.3.1 MCEP Event

As defined in section 3.1, an event is an object that contains information of an activity or an occurrence [Luck01, EtNi10]. Events from different event sources can be described in different event formats, which is an obstacle for achieving efficient event processing. In order to overcome this obstacle, we define a standard MCEP event model for all events used in the MCEP system.

The MCEP event model consists of two objects: **Event Type** and **Event Message**. **Event Type** defines the metadata and the list of the information that can be carried by an event. An **Event Message** is a concrete event object containing real-time information.

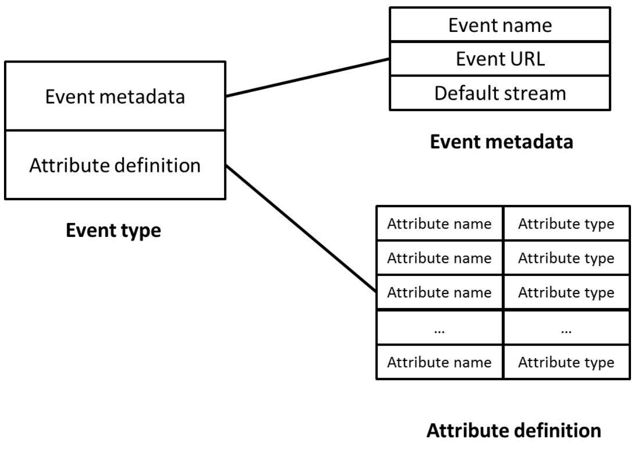


Figure 6-3 Event Type structure

Figure 6-3 shows the structure of **Event Type.** An **Event Type** consists of two parts: event metadata and attribute definition. Event metadata describes some metadata of such event type, including event name, event URL and the default stream. Attribute definition describes the structure of the information that is carried by the events of this type. Attribute definition defines the name and value type of each attribute. In the implemented prototype, the MCEP event model supports all JAVA primitive data types and more attribute types can be easily added in the future.

Considering that the event types are tightly associated with patterns, the event types are also stored in semantic form in pattern and event resource ontology (described in chapter 9). The metadata **Event URL** is the identity for the semantic object.

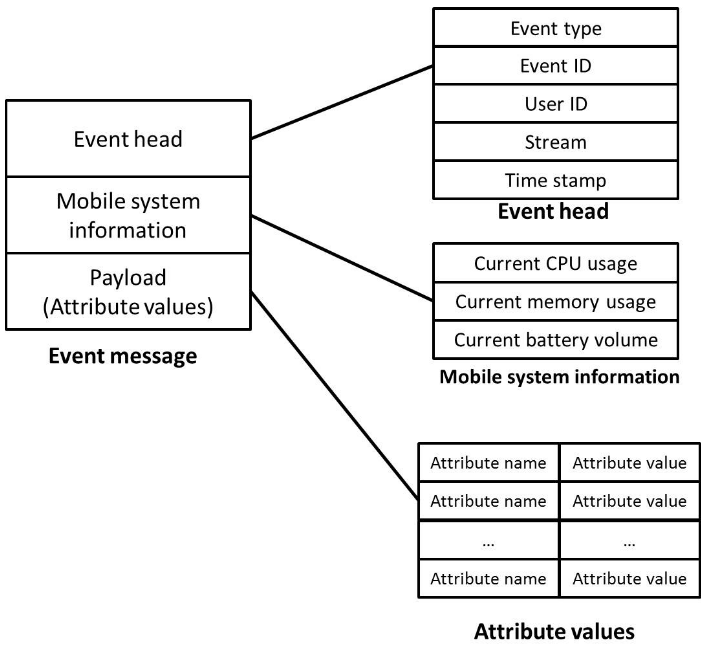


Figure 6-4 Structure of event message

Figure 6-4 illustrates the structure of Event Message. An Event Message can be divided into three parts: event head, mobile system information and payload. In the event head some metadata of the current event message are described:

- Event Type: This indicates the **Event Type** of the current event message.
- Event ID: This is the ID of the current event message. Each event message has a unique ID.
- User ID: It indicates the ID of the user, who sends this event message or should receive this event message.
- Stream: This is the stream of the current event message. By default it is same as the default stream defined in the event type of the current event message. A different stream can also be used to distinguish the events from different event sources.
- Time stamp: It indicates the time of the occurrence of the current event message.

Mobile system information stores the information of current workload of the mobile device, including current CPU usage, current memory usage and current battery volume. The information of workload is used in the dynamic pattern distribution algorithm (introduced in chapter 8) to achieve resource-aware pattern distribution. Only the event messages sent by mobile devices carry this information. The values of the mobile system information in the event messages sent by backend server stay zero.

The payload stores the information of activity or occurrence in the form of a list of attributes, which are defined by the **Event Type** of current event message.

According to the usage of events, the MCEP events can be divided into two categories: information events and system events. The information events carry the real-time information of personal data, environment data or the situation detected by CEP. The information events are normally used by CEP engine. The system events carry the system information or commands and are used to control the components during the run time.

6.3.2 MCEP Stream

An event stream is a set of associated events, which are temporally ordered [EtNi10]. An event stream can contain events, which have different event types. The events of the same event type can belong to different event streams as well. In the proposed MCEP system, event streams are not only used to indicate the associations of events, but also used to manage the dispatch of MCEP events.

As shown in the Figure 6-4, each MCEP event message belongs to an MCEP stream, which is defined in the event head. The stream manager (including **mobile stream manager** in mobile device part and **server stream manager** in server part) dispatches MCEP events based on their streams.

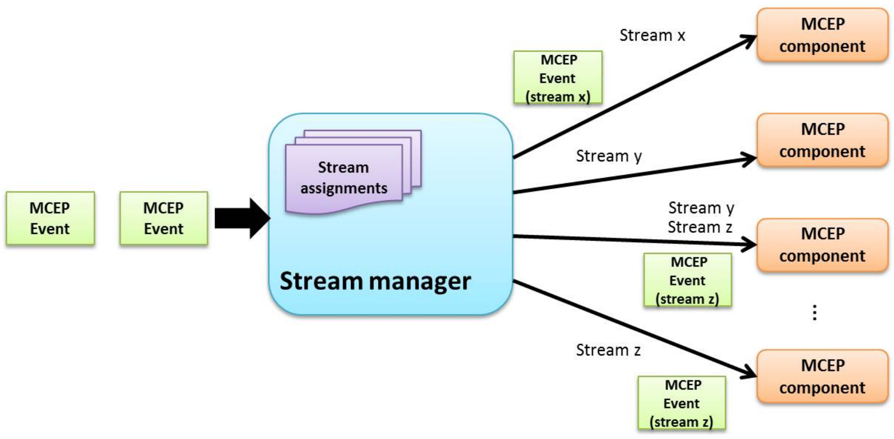


Figure 6-5 MCEP Event dispatch process

Figure 6-5 shows the process of MCEP event dispatch. Firstly all MCEP events will be sent to stream manager. Then stream manager sends the events to assigned targets according to the stream assignments, which are assigned by the system or use cases.

In addition, the MCEP streams are used to keep the privacy of the user. Each MCEP stream contains several metadata, and one of these metadata is the privacy level, which indicates the privacy of the stream data. Users can customize the privacy boundary for the MCEP mobile app. If the privacy level of a MCEP stream is higher than the customized privacy boundary, the events of this stream are not allowed to be sent to the backend server and can only be processed on the user's mobile device.

6.4 System variants

In the previous section we described the complete software architecture of proposed MCEP system, including mobile device part and backend server part with all features. Considering the different requirements of various mobile applications, some features are not required by some mobile applications. Therefore we also provide several variants of the proposed MCEP system focusing on different features.

Currently following two variants are provided:

- Full-version MCEP system
- Mobile-only MCEP system

The full-version variant consists of both mobile device part and backend server part. Taking advantage of the extended computation capacity by using backend server, the full-version MCEP system variant is able to undertake the complex monitoring tasks, which require high computation capacity. In addition, the backend server enables the context-aware personal data processing, which provides dynamic adaptation to the current situation of user based on user's context and additional domain knowledge. The feature of data collaboration is available in this variant. The full-version MCEP system variant is able to use external event resources using backend server. The feature of resource-aware dynamic pattern distribution ensures the optimal pattern distribution considering the available resource on mobile device.

The mobile-only variant is a light weight system using only the mobile device part of the MCEP system and focusing on the real-time event processing. Due to the resource limitation on mobile devices, this variant can only run limited number of processing tasks simultaneously. It can only provide limited context-aware adaptation based on the dynamic user context produced in run-time, without using the static user context and additional domain knowledge on the backend server. The mobile-only variant has no communication with backend server and can work without internet connection.

Table 6-1 Variants comparison

Features	Full-version variant	Mobile-only variant
Context awareness	Supported	Limited (no additional knowledge base)
Data collaboration	Supported	Not supported
Resource-aware dynamic pattern distribution	Supported	Not supported
Real-time pattern adaptation	Supported	Not supported
Sensor connection	Supported	Supported
External event sources	Supported	Not supported
Backend server	Required	Not required
Internet	Required	Not required
Number of simultaneous active patterns	No limitation (offloading to backend server)	Limited

Table 6-1 compares two variants regarding features. Based on the different usage, mobile applications can use different variants depending on their requirements.

6.5 Conclusion

In this chapter we have introduced event-driven hybrid system architecture for data intensive mobile applications, which provides a basis for efficient personal data processing on mobile devices. The proposed hybrid system architecture uses external backend server to extend processing capacity with the purpose of solving the issue of the resource limitation on mobile devices. The backend server part of the proposed architecture also provides functionality of context awareness using additional knowledge bases. The event and stream model enables personal data processing through CEP technologies on mobile devices. The communication middleware provides various communication types through publish/subscribe mechanism to enable efficient event transmission between mobile devices and the backend server. Two system variants are provided to meet the different requirements of various use cases.

7 Context-aware Data Processing and Data Collaboration

In this chapter we describe the data processing and data collaboration that are based on the context-aware monitoring goal. Context-aware data processing enables the dynamic adaptation to real-time personal data considering the user's current situation that takes user's context and additional domain knowledge into account. We introduce the so-called Monitoring Goal Network (MGN), which is used to model user's situation, to correlate the real-time personal data with user's context and domain knowledge, and to define the possible reactions to the detected user's situations. In the second section we introduce the data collaboration in the proposed system, focusing on the search for relevant users using the monitoring goal defined in MGN.

7.1 Context-aware data processing

By taking advantage of embedded sensors in mobile devices and various wearable sensors, mobile applications are able to capture a large amount of various personal data of a user in real-time. However, most real-time personal data is difficult to interpret without additional information about user's context and domain knowledge. For example, if the user has cardiac disease, the heart rate data of this user should be processed and interpreted in different ways than that of a healthy person.

In addition, there are many possible processing tasks that may be performed based on various real-time personal sensing data. For instance, the process of detecting whether a user's running speed is in the normal training range may not yield meaningful interpretation if the user doesn't do fitness training but is sitting in a chair. These redundant processing tasks can increase the computation capability requirements of the system and hinder performance. This should be avoided in the resource limited mobile application development.

Thus, according to the requirement R4 defined in 5.3, the real-time personal data must be correlated with the user's context regarding the user's current situation to achieve more precise and efficient personal data processing. For different situations mobile applications should use different patterns that are narrowed and relevant to the situation in which the user is involved.

7.1.1 Monitoring Goal Network

One of the most important issues in realizing context-aware dynamic adaptive personal data processing is the modelling of the situations that user may encounter. In the proposed system we propose to use monitoring goal based data processing to realize dynamic adaptive context-aware personal data processing in real-time. We model data processing procedure in a **Monitoring Goal Network (MGN)** for defining the processing tasks in each monitoring procedure and systems' reactions to situations. The idea of MGN is taken from Hierarchical Task Networks (HTNs) [Mitc97, HoLM05], Behavior Trees (BTs) [Drom06, BeTG07] and Situation Action Network (SAN) [PPVA+13].

MGN is used to define the processing tasks (i.e. patterns) of the monitoring procedure, possible situations that can happen during the monitoring procedure and reactions to them at design time, where it correlates the real-time personal data with user's context and domain knowledge that will be able to recommend at run-time solutions to the detected situations. Reactions to the detected situations include sending recommendation to the user, deploying additional new patterns, saving information of the situation as user's context, etc.

In reality, however, it is difficult to predefine every possible situation that might happen along with corresponding reactions at design time. Nevertheless we are able to define some general monitoring goals in advance for describing the guidelines for the system behaviors.

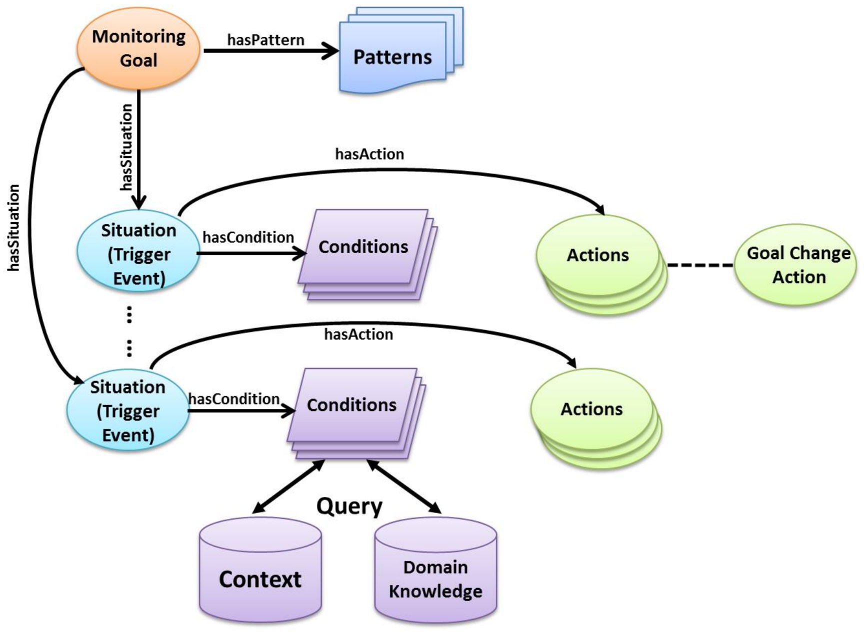


Figure 7-1 Monitoring Goal

Definition 7.1: A **Monitoring Goal** is a set of rules that define the situations, the condition of situations and the reactions to adapt to such situations. The situations in the same monitoring goal are possibly happening in the same monitoring procedure and related to the same monitoring purpose.

For example, a monitoring goal can be defined for jogging monitoring. The situations of interest of jogging monitoring could be “running too fast”, “running too slow” and etc.

Figure 7-1 shows the detailed structure of a monitoring goal, including ***Pattern***, ***Situation***, ***Condition*** and ***Action***. A monitoring goal is a tree structure and each node carries specialized functionality. We explain each of these nodes as follows:

- **Monitoring Goal Node**: this is the root node of the tree and specifies the purpose of the monitoring. A monitoring goal node contains the processing tasks, which enable to achieve the monitoring purpose defined in this goal and the possible situations that might happen. The monitoring goal has two types of child nodes: **patterns** and **situations,** corresponding to the processing tasks and predefined possible situations. One monitoring goal node can have several instances for each type of child nodes.
- **Pattern**: the patterns indicate the processing tasks, which correlate to the current monitoring goal. The patterns are used by the CEP engine to process the real-time personal data in order to detect the situations of interest belonging to the monitoring goal. The patterns of a monitoring goal will be deployed, once the monitoring goal is activated.
- **Situation**: The situations denote the situations of interest. Each situation is triggered by an event called trigger event. Most trigger events are detected by patterns of the current monitoring goal. However some situations also use real-time personal sensing data as trigger event. Situations have two types of child nodes: **condition** and **action**. Each situation can have more than one condition and action.
- **Condition:** The condition node specifies the requirements that the detected situation should fulfill. The requirements defined in the conditions are not static. They can also be the values from the domain knowledge or run-time user context. Hence the conditions combine the real-time personal data with additional information including user’s context or domain knowledge.
- **Action:** The action node indicates the adaptation to the detected situation, such as to deploy new patterns or to send a recommendation to the user. One important action is “Goal change action”, which changes the active monitoring goal of the MGN. The actions of a situation will be run, only when the situation is triggered and all belonging conditions are fulfilled.

A monitoring goal models the monitoring procedure for a special purpose that correlates the real-time personal data with user’s context and defines the adaptation to detected situation. The execution process of a situation (triggered by event), conditions and actions can be treated as E-C-A (Event Condition Action) rules [HKMS94, DiGG95, KaRR98].

Definition 7.2: A **Monitoring Goal Network** is a network consisting of several connected monitoring goals.

Each monitoring goal might have several predecessors and successors, which make the MGN a directed graph. Each MGN has one and only one staring monitoring goal, which indicates the starting point of this network. After the MGN has been started, the starting monitoring goal will be activated immediately, afterwards the active goal shifts among all goals depending on the real-time situation that the user is involved in.

7.1.2 MGN Execution process

In this subsection we explain how the MGN works. The MGN consists of multiple monitoring goals, where there is only one active monitoring goal in the whole MGN at any given time. The active monitoring goal processes the situation events that are detected by CEP. The first active monitoring goal is the starting monitoring goal.

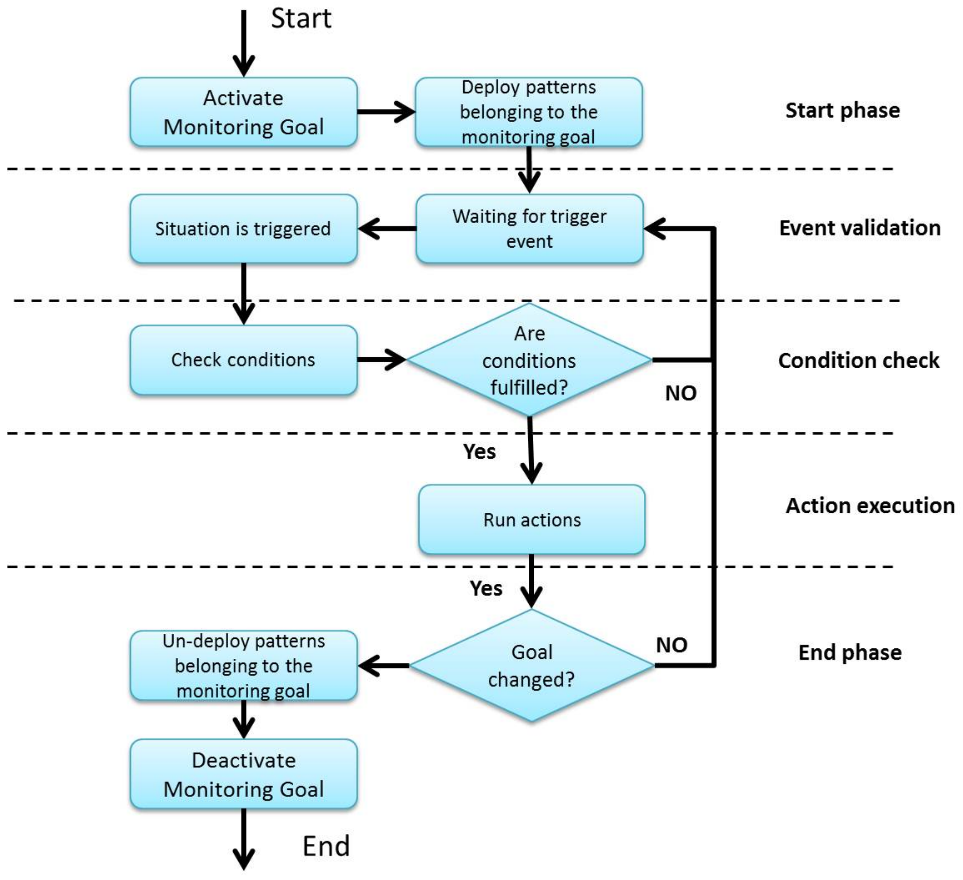

Start
Activate Monitoring Goal
Deploy patterns belonging to the monitoring goal
Start phase
Situation is triggered
Waiting for trigger event
Event validation
Check conditions
Are conditions fulfilled?
NO
Condition check
Yes
Run actions
Action execution
Yes
Un-deploy patterns belonging to the monitoring goal
Goal changed?
NO
End phase
Deactivate Monitoring Goal
End

Figure 7-2 Flow diagram of monitoring goal execution

Figure 7-2 illustrates the execution procedure of an active monitoring goal. The whole execution process consists of five phases: start phase, event validation, condition check, action execution and end phase.

- Start phase: Once a monitoring goal is activated, all patterns belonging to this monitoring goal will be deployed immediately, which process the real-time personal data for the detection of possible situations.
- Event validation: In case an event is sent to the monitoring goal, each situation should check whether the incoming event is a predefined trigger event. If the incoming event is a predefined trigger event, then the execution continues to the next condition check phase, otherwise all situations wait for next input event.
- Condition check: In this phase all conditions belonging to the triggered situation should be checked. Only when all conditions of such situation are met, the execution of this situation can continue. If one condition fails, the execution of this situation will be stopped and the situation goes on for waiting for future trigger event.
- Action execution: When all conditions are met, all the actions that are assigned to this situation should be executed.
- End phase: If a monitoring goal change action has been successfully executed in the action execution phase, the current monitoring goal should be deactivated. Right before the deactivation, all the patterns that belong to this goal and were deployed in the start phase will be un-deployed.

7.1.3 MGN Implementation

We developed an RDF-based ontology for formally expressing all the characteristics of MGN and a MGN API for implementing MGN in Java. The semantic-based RDF language [KlCa06] enables the formal description of the MGN model (see Appendix I), which can be used in different systems. In addition, the RDF-based model description can be easily extended in the future based on the new requirements. In this subsection we introduce the implementation details of MGN.

Figure 7-3 illustrates the ontology of MGN model. In order to elaborate the ontology structure in a more clear way, Figure 7-3 shows a simplified version of ontology with only the important classes and properties. The complete ontology is attached as appendix at the end of this thesis.

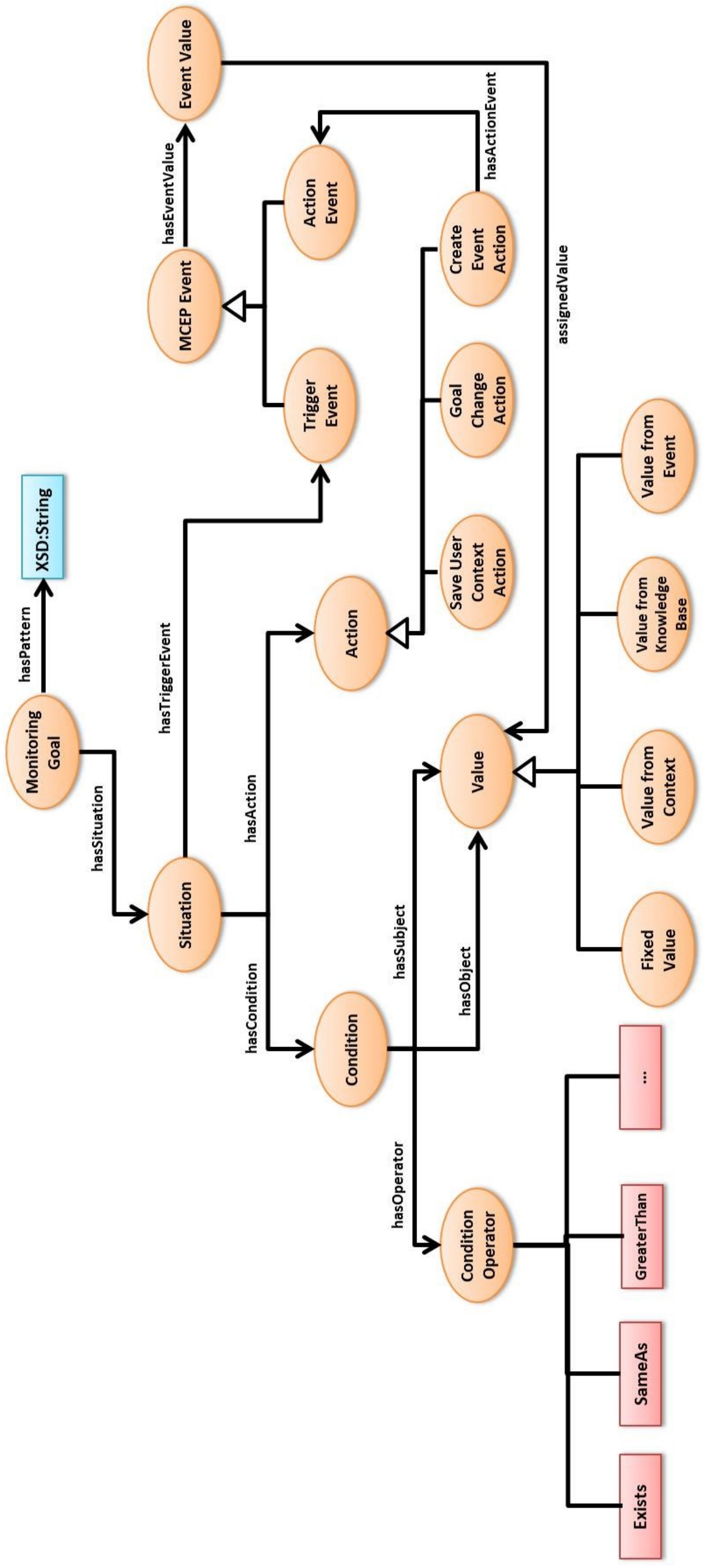


Figure 7-3 Monitoring goal network Ontology (simplified)

In the following we introduce the classes and their functionality briefly.

- **Monitoring goal:** this class models the monitoring goal concept introduced in the previous section. It has a data property ***hasPattern,*** which links a String data indicating the ID of the pattern that is used in this monitoring goal. It connects to **Situation** through object property ***hasSituation.***
- **Situation:** this class encompasses all the possible predefined situations of a monitoring goal. It has three important object properties. ***hasTriggerEvent*** indicates the event that triggers this situation. ***hasCondition*** indicates the condition that should be fulfilled. ***hasAction*** shows the connection between situations of actions.
- **Condition:** this class models the condition of situations. Each condition presents an expression, consisting of one subject, one object and one condition operator. The condition connects to these three parts through three object properties: ***hasSubject***, ***hasObject*** and ***hasOperator***. When the expression is asserted as true, the condition is fulfilled. The subject and object are instances of **Value** class.
- **Action:** this class indicates the action of situations. We predefine three concrete actions, which are most important for the proposed system.
    - **Goal Change Action:** as already introduced, this action changes the active monitoring goal of the MGN.
    - **Save User Context Action**: this action saves the information from the current trigger event of the detected situation as user context, which can be used later in the situation analysis.
    - **Create Event Action:** this action creates a new real-time event and sends the event to the system. The created event can be either a real-time personal information event, which is conducted according to user's run-time context or domain knowledge, or a predefined system event, such as pattern deployment (i.e., to deploy or un-deploy patterns) or recommendation event (i.e., to show a recommendation on user's mobile device). This action provides a general interface to interact with the system. It indicates the concrete **Action Event** through ***hasActionEvent*** property.
- **Value:** this class indicates the values, which can be used as subject/object of conditions or concrete value assignment of the event values. According to the sources of values, we define four sub-types of values.
    - **Fixed Value:** the value is predefined and fixed. It cannot be changed during the run-time.
    - **Value from Context:** this value is from the user's context, which was saved in run-time.

  - **Value from knowledge base:** values in this type are retrieved from the related domain knowledge.
  - **Value from event:** this type indicates that the value is from the event that triggers the current situation.
- **MCEP Event:** this class abstracts the event definition of event types used in the proposed system. Concretely it is used as **trigger event** and **action event**. Each MCEP event can have **Event Value,** which specifies the concrete value of the event.
  - **Trigger event:** this class denotes the event that triggers the predefined situations.
  - **Action event:** this class denotes the event that is created by the creating Event Action.

7.1.4 Example

We give a concrete example of MGN in this sub-section that shows how the MGN is defined and executed.

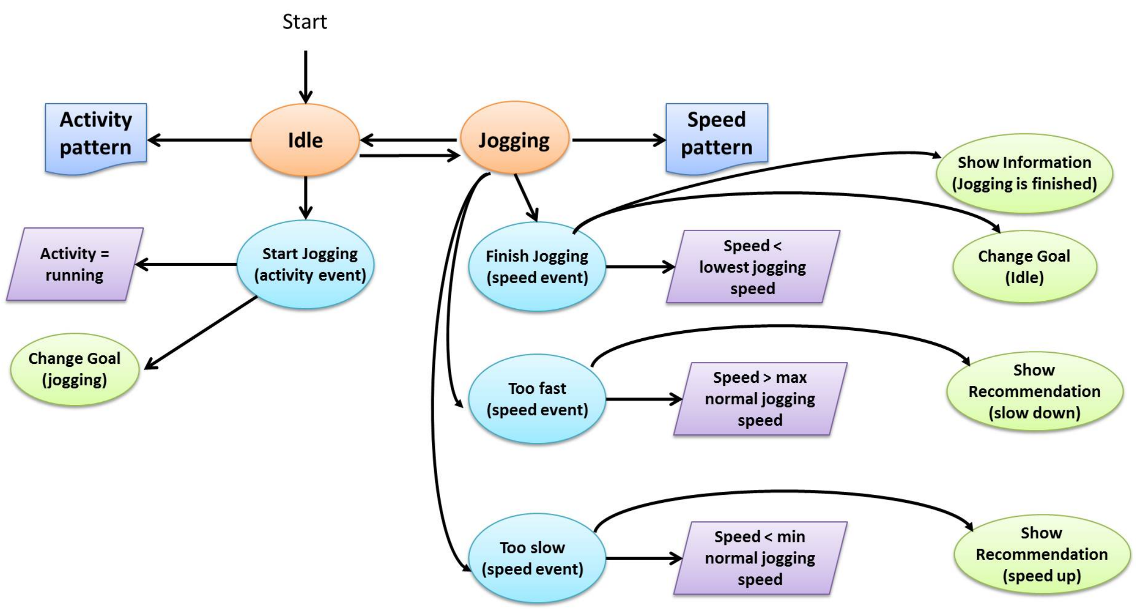

Start
Activity pattern
Idle
Jogging
Speed pattern
Show Information (Jogging is finished)
Activity = running
Start Jogging (activity event)
Finish Jogging (speed event)
Speed < lowest jogging speed
Change Goal (Idle)
Change Goal (jogging)
Too fast (speed event)
Speed > max normal jogging speed
Show Recommendation (slow down)
Too slow (speed event)
Speed < min normal jogging speed
Show Recommendation (speed up)

Figure 7-4 An example of monitoring goal network

As shown in Figure 7-4 a simple MGN is defined for jogging monitoring. The whole MGN contains only two monitoring goals: “Idle” and “Jogging”. On the left side of the figure is the “Idle” monitoring goal. It monitors user’s activity to recognize the start of the jogging. It is the start monitoring goal of the MGN. On the right side is the “Jogging” monitoring goal, which monitors the speed of a user to ensure that the user runs at a right speed.

The “Idle” monitoring goal has only one assigned pattern: activity pattern, which generates activity events indicating the current activity of users. In the “Idle” monitoring goal one situation is defined, called “Start Jogging”. This situation is triggered by the activity event. In the case the activity type indicated by the incoming activity event is running, a “Change Goal” action will be executed, which changes the current monitoring goal to “Jogging” monitoring goal.

The “Jogging” monitoring goal has also only one assigned pattern: speed pattern, which detects the current running speed of the user in real-time and generates the speed events. Three situations have been defined in the “Jogging” monitoring goal: “Finish jogging”, “Too fast” and “Too slow”. All the three situations are triggered by speed events.

In the “Finish jogging” situation, the speed of the user will be compared to “lowest jogging speed” that has been defined in a domain knowledge base about jogging. This value describes the boundary speed between walking and running for most people based on scientific studies. If the speed of user is lower than this value, indicating that the user is walking instead of running, two actions will be executed. Firstly a piece of information will be shown on user’s mobile device to inform the user that the jogging is finished due to the low speed. Secondly a “Change Goal” action will be executed, which changes the current monitoring goal to “Idle”.

In the “Too fast” situation, the speed value in the trigger event will be compared to the “max normal jogging speed”, which is stored in the user’s context reflecting the user’s historical training records. In the case where the current speed of the user is higher than this value, a “show recommendation” action will be executed to warn the user about the high speed and to recommend the user to slow down.

The “Too slow” situation is similar to the “Too fast” situation. It checks the current speed of the user and recommends the user to speed up in the case the user runs too slowly.

7.2 Data collaboration

The most mainstream mobile applications, which process the real-time data for monitoring, monitor users individually and provide no data collaboration between different users. However, the data collaboration is very important during the monitoring in some special cases. For example, a cardio problem that has been sensed from one patient can be better analyzed by having some values measured by another patient who is in a similar context. The data from the users who are in a similar context may improve the monitoring quality in some special cases.

One of the most important advantages of today’s mobile applications is the large number of users, which make it possible to achieve data collaboration among the users having the similar context.

According to the requirement R5 defined in 5.3 and to exploit the potential of mobile applications, we provide data collaboration in the proposed mobile software system.

7.2.1 Data collaboration process

In this subsection we introduce the data collaboration procedure in the proposed system in order to explain how data collaboration works.

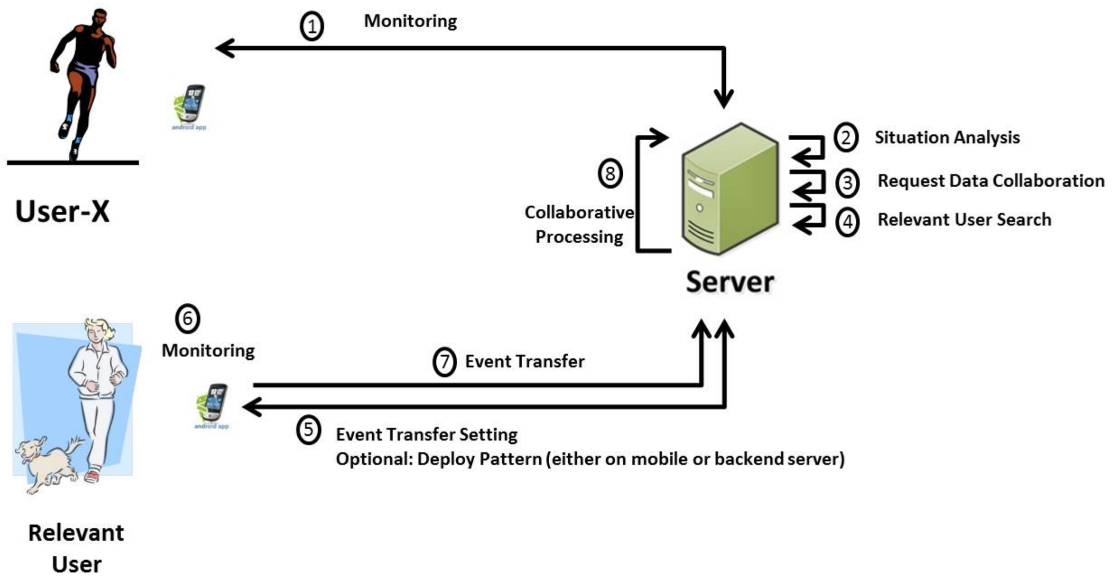


Figure 7-5 Data collaboration process

Figure 7-5 illustrates the data collaboration procedure implemented in the proposed system. User-X is a user of the proposed system. The predefined monitoring procedure in MGN for User-X requests data from other users, who are similar to User-X, in some special situations (i.e., data collaboration). The whole data collaboration procedure contains 8 phases:

1. Monitoring: in this phase the mobile application monitors user-X in a normal mode processing real-time data only from user-X.
2. Situation analysis: the system analyzes the detected situations of user-X using predefined MGN and reacts to the detected situations.
3. Request data collaboration: In the case the monitoring requests additional data from other user in some special situations (defined in MGN), MGN sends an action event to indicate such data collaboration request. The action handler on the backend server processes this request and initiates data collaboration.
4. Relevant user search: this is the most important phase in the data collaboration. The proposed system searches the relevant users among all registered users, in order to find the most suitable users to provide collaborative data. Combining the monitoring goal based da-

ta proceeding introduced in the previous section, we develop monitoring goal based search mechanism to achieve this goal. The details about the monitoring goal based search mechanism are introduced in the next subsection.

5. Event transfer setting: After finding the most suitable relevant users, who can provide the data for the data collaboration, the system sets the **event transfer** to collect data from different users. Event transfer creates a copy of the original event and sends the copy to the target user (i.e., user-X), who requested data collaboration. In the case the requested event is not directly available and must be generated by a pattern, the pattern will be deployed in order to enable the data collaboration.
6. Monitoring: the requested data for collaboration is obtained by monitoring of relevant users.
7. Event transfer: once the requested data is available, it will be sent to the backend server for data collaboration.
8. Collaborative processing: the backend server processes the data from different users collaboratively for data processing of user-X.

The advantages of the proposed data collaboration procedure can be summarized as following:

- It enables collective data processing by combining the data from different users in real-time.
- It supports user search to find the most suitable relevant users for data collaboration.
- It deploys patterns for detecting requested real-time data on the fly.

7.2.2 Monitoring goal based relevant user search

In this subsection we introduce the monitoring goal based relevant user search aiming to find the most suitable users for data collaboration. As mentioned earlier, the large number of users is an important advantage of mobile applications. However, the large number of users also leads to the problem of finding relevant users for data collaboration. The suitability of the relevant users affects the quality of the collaborative data processing. The data collaboration can be worthless by using the data from the unsuitable or irrelevant users. For example, it doesn't make sense to compare the heart rate of a person, who is sprinting, with the heart rate of a person, who is sitting in Starbucks and drinking latte, although both people may have the similar health status. Therefore, the search of suitable relevant users is a key issue in data collaboration.

As introduced in 7.1.1 we model the monitoring procedure in Monitoring Goal network (MGN) and split the whole procedure into several connected monitoring goals. Normally, all predefined situations in a monitoring goal relate to the same activity and same purpose. Hence, the users, who are monitored using the same monitoring goal, have a high probability of being the suitable relevant users for data collaboration of each other. Using the model of monitoring goal, we develop monitoring goal based search mechanism to find the most suitable relevant users for data collaboration.

The monitoring goal based search mechanism has two search phases:

1. Monitoring goal search: in the first phase the proposed system searches across all the registered users to find out those who are using the same monitoring goal with the current user, who requests the data collaboration.
2. User-defined context-based search: considering the different requirements of concrete use cases, it is also possible to search for the users using custom search criteria based on different context of user, such as location, age group, health status and etc. or the combination of these criteria.

In order to make the search more flexible, both phases are optional, meaning that the user can use only monitoring goal search or user-defined context-based search alone.

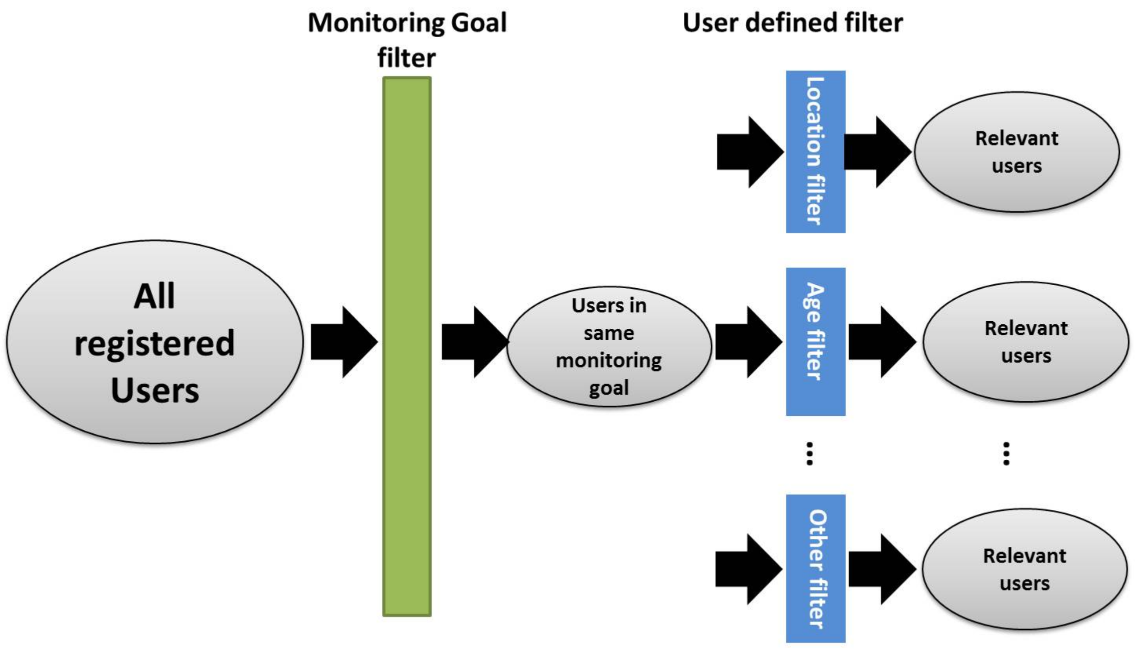


Figure 7-6 Monitoring goal-based search mechanism

The searches are based on various filters. Figure 7-6 illustrates the monitoring goal-based search mechanism. The monitoring goal based search is realized through monitoring goal filter, which is implemented and provided by the proposed system. The user-defined context-based search uses the filters, which should be implemented by the developer later according to individual cases. All registered users are filtered by monitoring goal filter and user defined context-based filter sequentially. The results of the search are the most suitable relevant users for the data collaboration.

7.3 Conclusion

In this chapter we have introduced context-aware data processing and data collaboration. The Monitoring Goal Network (MGN) is described, which enables the modelling of monitoring procedure and the integration of the real-time data and context data including user's historical data and domain knowledge bases. Using MGN the proposed MCEP system is capable of achieving dynamic adaptive monitoring based on the user's current situation.

The feature of data collaboration enables the MCEP system to process the personal data collaboratively. Combining the monitoring goal based data processing the MCEP system is able to find the most suitable relevant users for data collaboration.

8 Dynamic Pattern Distribution

In the last chapter we solved the problem of modeling the monitoring procedure. By applying the monitoring goal network, the proposed system uses different patterns for different monitoring purposes. In this chapter we describe the dynamic pattern distribution for efficient pattern distribution. We develop a pattern distribution model regarding the proposed hybrid software architecture and an intelligent pattern distribution algorithm for the resource-aware pattern distribution considering the resource limitation of mobile devices. An example of distribution calculation is also provided.

8.1 Pattern distribution model

As stated in the previous chapters, one of the most important challenges for mobile applications is the resource limitation on mobile devices. Limited computing resources on mobile devices restrict the performance of real-time data processing and lead mobile devices to be incapable of executing some complex processing tasks, which are essential for real-time personal data processing. To make it even worse, battery, as the only power source of most mobile devices, has seen relatively slow improvement in the past decade and becomes a major impediment to providing reliable and sophisticated mobile applications to meet the real-time data processing requirements.

In the proposed system we use two methods to overcome these limitations: the extension of computation capacity and the reduction of simultaneous processing tasks on mobile devices. Firstly we combine the remote computing framework and local mobile CEP infrastructure to build our hybrid software architecture, which has been introduced in Chapter 6. The hybrid software architecture enables the proposed system to efficiently process personal data on mobile devices in real-time, and at the same time to extend the processing capacity by offloading processing tasks (patterns) to backend server.

Secondly we use Monitoring Goal network (MGN) to model the monitoring procedure to achieve adaptive data processing, which is introduced in Chapter 7. The whole monitoring procedure is split into several connected monitoring goals, and in each monitoring goal all required patterns for the monitoring are assigned. Only the assigned patterns are used in the execution of each monitoring goal. Hence the number of simultaneously running patterns can be substantially reduced.

Although the adaptivity of data processing reduces the redundant patterns, due to the resource limitation on mobile device, it is still unable to execute all patterns on mobile device for some complex monitoring tasks, which must use numerous patterns and may cause heavy computation load (see 10.1.1). In such cases some of these patterns have to be offloaded to the backend server, in order to reduce the workload on mobile devices.

To realize such pattern offloading in the proposed system, the patterns might be distributed to either mobile device or backend server. However, there are other challenging issues in distributing patterns: Which pattern can be executed on mobile device and which pattern must be deployed on backend server? When should patterns be distributed to mobile devices and when should they be distributed to backend server?

In addition, a pattern uses several events as input for detection of a situation of interest. In the proposed system one event can have multiple different **Event Resources**.

Definition 8.1: An **Event Resource** can be defined as a stream of events, which have identical event type and are from identical source.

For example, a temperature event can be gathered either from the sensor that is equipped by the user or from other open information sources, such as a weather station. Another example is, if a user wears two different sensors and both of the sensors can detect the heart rate, then the heart rate event have two sources. Both the temperature event and the heart rate event in the examples have two event resources. Different event resources can have different attributes, such as frequency or cost, although they present the same event type.

Due to the multiple event resources of an event used in a pattern, a pattern can have several **pattern bindings** with different event resource allocation.

Definition 8.2: A **Pattern Binding** is an instance of pattern with concrete event resource allocation.

In the case an event, which is required by a pattern, has more than one event resources, it means that the pattern also has more than one pattern bindings. By pattern distribution the proposed system should also decide which pattern binding will be used. Since different event resources have different attributes, different pattern bindings also lead to different execution requirements. For example, a pattern binding may require more communication capacity than all other pattern bindings if it uses an event resources having high event frequency; or another pattern binding may require additional cost, if the event resource it used is obtained from a paid (not free) web service.

In addition, different deployment positions can lead to different execution requirements as well. In the case a pattern binding uses the most event resources from mobile device, if such pattern binding is deployed on backend server, all required events should be transmitted to the backend server, which can cause a heavy communication workload.

In order to solve the problems mentioned above: we need (1) to find the best event resource allocation among multiple event resources and (2) to perform optimal distribution of patterns, a pattern distribution mechanism that is required to allocate the event resources and distributes patterns based on the current available resources of mobile devices.

According to the requirement R3 defined in 5.3 we develop a dynamic pattern distribution mechanism regarding the resource limitation on mobile devices. For the purpose of comparing between different pattern bindings and deployment positions, we use **Distribution Fitness**[31] to measure the suitability of each pattern binding and deployment position. The dynamic pattern distribution calculates the distribution fitness of all possible pattern bindings and deployment positions (mobile device or backend server) according to the current workload of mobile device and backend server and deploys the result, which has the highest fitness value.

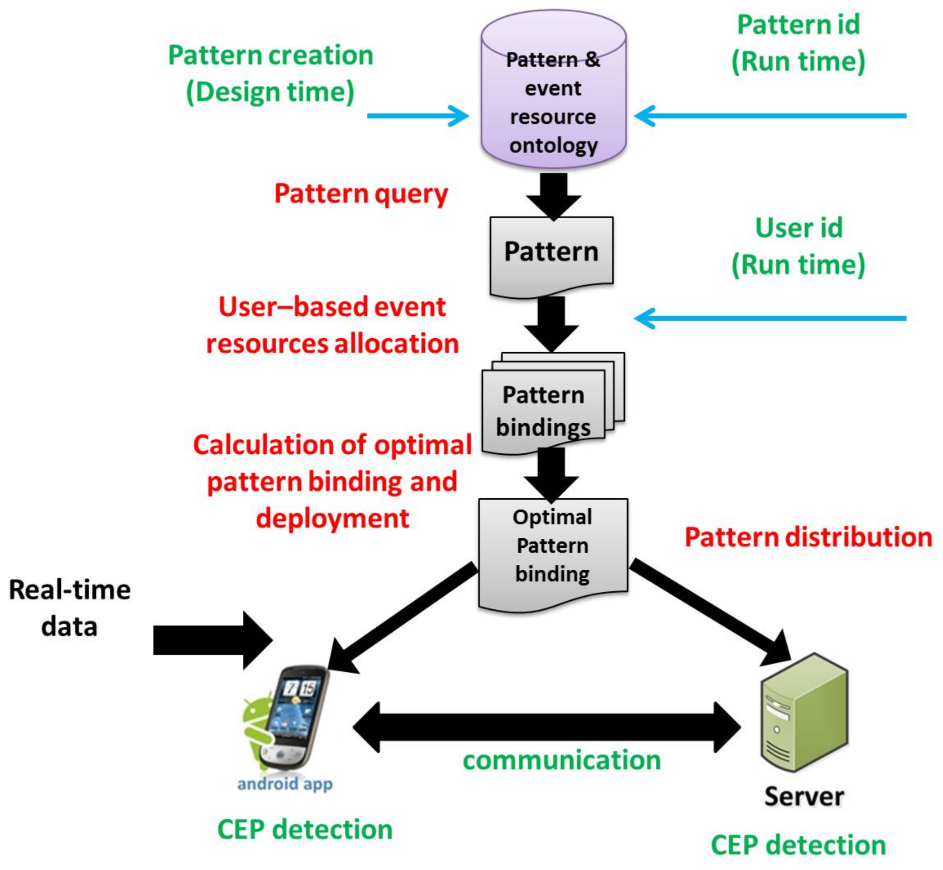


Figure 8-1 The process of dynamic pattern distribution

[31] Distribution Fitness is a value that indicates the propensity of the distribution. The word “fitness” here doesn’t refer to sport fitness.

Figure 8-1 shows the model of dynamic pattern distribution. The whole model consists of four steps: Pattern query, user-based event resources allocation, calculation and distribution. We explain the functionality of each step as follows.

1. Pattern query: in the proposed system all patterns are predefined at design time and stored in pattern and event resource Ontology (this ontology is introduced in Chapter 9). As soon as a pattern is required for the data processing in run time, the system queries the pattern and event resource ontology to find the required pattern according to the pattern id, which is defined in MGN.
2. User-based event resources allocation: due to different equipment of users, different users also have different event resources. In this step the system searches all the available event resources of the user for the events required by the pattern and creates all possible pattern bindings, which make the pattern executable.
3. Fitness calculation: In this step, for each pattern binding we calculate the distribution fitness values for both possible deployment positions (i.e., mobile device and backend server).
4. Distribution: in the last step the proposed system finds the optimal result, which is the combination of pattern binding and deployment with the highest distribution fitness value. Then the pattern is deployed to the deployment position of the optimal result and sets the stream assignment for the event resources used by the optimal result.

8.2 Dynamic pattern distribution algorithm

In the previous section we introduced the general idea of dynamic pattern distribution and described the dynamic pattern distribution model. In this section we explain the implementation details of the proposed dynamic pattern distribution focusing on the fitness calculation.

We implement a dynamic pattern distribution algorithm to realize the dynamic pattern distribution in the proposed system. For a better understanding we present the algorithm in the following pseudo-code.

Algorithm 8-1 Dynamic Pattern Distribution Algorithm

```
Dynamic Pattern Distribution Algorithm

Step 1
1: Pattern Patt = query (pattern_id);                    // search for a pattern with given id
Step 2
2: Event = Patt.getEvents ();                            // Event is a set of all events used in the pattern Patt
// Event_resource is a set of the event resources of each event in Event
3: Event_resource ← searchEventResource (Event, user_id );
4: PattB = allocate (event_resource); // PattB is a set of pattern bindings with a different event resource allocation
Step 3
5: Deployment = {server, mobile};                        // Set of possible deployment positions
6: for each PattBi ∈PattB
7:     for each deploy ∈ Deployment
8:         fitness = calculate (PattBi, deploy);
9:         Result.put ((pattBi, deploy, fitness));       // all fitness results are saved in the Set Result
10:    end for
11: end for
Step 4
12: optimal = getHighestFitness (Result);                // search for the highest fitness value
13: patternBinding = optimal.getPatternBinding ();
14: location = optimal.getDeployment ();
15: deploy (patternBinding, location);
```

The line 1 realizes the first step of pattern distribution. The **query** function searches the pattern and event resource ontology and returns the pattern with given pattern id. The codes from line 2 to line 4 implement the second step. The **getEvent** method of pattern returns all required events of such pattern. The **searchEventReource** function searches all available events resources for given user and events. The **allocate** function creates all possible pattern bindings. In line 5 **Deployment** lists both deployment positions. The codes from line 6 to line 11 use two loops to calculate the fitness values for all combinations of pattern bindings and deployment positions. The **calculate** function realizes the concrete distribution fitness calculation. The last step distribution is implemented in line 12 to line 15. The **getHighestFitness** function returns the best result according to the calculated fitness values. The codes in line 13 and 14 get the pattern binding and deployment position from the optimal result. In line 15 the **deploy** function deploys the best result.

Obviously the core of the dynamic pattern distribution is the distribution fitness calculation of each combination of pattern binding and deployment position. Afterwards we explain the details of how to calculate the distribution fitness.

The calculation is based on a set of formulas, which rate the suitability of pattern distribution from different aspects, including computation workload, communication and cost of deploying a pattern with a set of event resources on a deployment position.

Before explaining the formulas we introduce some definitions in order to describe pattern in a formal way. According to [SeSt10] a pattern can be presented as a tuple object **P (E, EO)**, where **E** is the collection of the events used in pattern and **EO** is the collection of the pattern operators used in pattern. Each event $\mathbf{e} \in \mathbf{E}$ can have several different event resources: $\mathbf{S_e} = \{\mathbf{s_1}, \mathbf{s_2}, \dots \mathbf{s_n}\}$. For example, a temperature event can be obtained from either a wearable sensor equipped by the user or a web service of weather station. Due to the different allocations of event resources, each pattern $\mathbf{p} \in \mathbf{P}$ can have several pattern bindings: $\mathbf{PB}_{\boldsymbol{p}} = \{\boldsymbol{pb_1}, \boldsymbol{pb_2}, \dots \mathbf{pb_n}\}$. A certain pattern binding **pb** has a fixed allocation of event resources, which are different from all other pattern bindings.

Each event resource has a set of attributes, in which two attributes are important for the distribution calculation: frequency and cost.

Definition 8.3: The **frequency** of event resource indicates the number of events that are provided by such event resource within a time unit.

Different event resources of an identical event can have different frequencies. For example an integrated temperature of a smartphone generates temperature once per second, while web service of weather station updates the temperature information every minute. Different frequency leads to different communication workload, which should be considered in the distribution calculation. The frequency of event resources is described through frequency factor. The frequency factor of event **e** regarding the different event resources $\mathbf{S_e}$ can be presented as $\mathbf{F_e} = \{\mathbf{f_{s1}}, \mathbf{f_{s2}}, \dots \mathbf{f_{sn}}\}$.

Definition 8.4: The **cost** of event resource indicates the money, which is required to be paid for the information.

Taking the commercial information providers into account, we also consider the costs of the event resources in the distribution calculation. The cost of the event **e** regarding the different event resources $\mathbf{S_e}$ can be presented as $\mathbf{C_e = \{c_{s1}, c_{s2}, \ldots c_{sn}\}}$.

The distribution fitness of a pattern binding and a deployment position can be calculated by following formula:

$$\boldsymbol{Fitness_{(pb,d)} = w1 \times Cost_{pb} + w2 \times DF_{(pb,d)}} \tag{1}$$

where

- ***pb*** is the pattern binding,
- ***d*** is the deployment position (i.e., mobile device or backend server),
- $\boldsymbol{Cost_{pb}}$ denotes the cost factor of such pattern binding,
- ***DF*** denotes the deployment fitness of such pattern binding,
- ***w1*** and ***w2*** are the weight coefficients of the cost factor and deployment fitness,
- ***w1*** > 0, ***w2*** > 0, ***w1*** + ***w2*** = 1.

The value of distribution fitness is in the range [0, 1]. The whole formula consists of two parts: cost factor and deployment fitness. The cost factor rates the pattern binding on the financial aspect and deployment fitness rates the pattern binding and deployment position on the workload aspect. For the system using no commercial information providers, the cost factor can be ignored (set the **w1** as 0). For different users/systems the importance of both parts can vary using different weight coefficients **w1** and **w2**.

The cost factor indicates the expense of a certain pattern binding comparing to all other pattern bindings. The concrete value of cost factor can be calculated as following:

$$\boldsymbol{Cost_{pb} = \begin{cases} 1, & if \sum_{e \in E'} max \quad c_e = 0 \\ 1 - (\sum_{s \in S'} c_s \ / \sum_{e \in E'} max \quad c_e), & otherwise \end{cases}} \tag{2}$$

- ***E'*** is the set of all events used in the current pattern,
- **S'** is the set of event resources used in the current pattern binding,
- $\boldsymbol{\sum_{s \in S'} c_s}$ is sum of costs of the event resource used in pattern binding ***pb,***
- $\boldsymbol{\sum_{e \in E'} max \quad c_e}$ is the sum of the max cost of the events used in pattern ***p***.

The value of cost factor is between 0 and 1. The higher value of cost factor means that the pattern binding performs better on the financial aspect. If the cost factor of a pattern binding is 1, the pattern binding is totally free and uses no paid information provider. In contrast, if the cost factor of a pattern binding is 0, such pattern binding is the most expensive pattern binding among all pattern bindings. In the case the pattern uses no paid information provider in any pattern bindings, the values of all pattern bindings are constantly 1.

The **deployment fitness** indicates the workload caused by a certain pattern binding deployed on a certain position considering both computation workload and communication workload. The concrete value of deployment fitness is calculated using following formula:

$$\boldsymbol{DF_{(pb,d)} = w3 \times CF_{(pb,d)} + w4 \times LF_d} \tag{3}$$

where

- $\boldsymbol{pb}$ is the pattern binding,
- $\boldsymbol{d}$ is the deployment position (i.e., mobile device or backend server),
- $\boldsymbol{CF}$ denotes the communication fitness of the deployment,
- $\boldsymbol{LF}$ denotes the workload fitness of the deployment,
- $\boldsymbol{w3}$ and $\boldsymbol{w4}$ are the weight coefficients of the communication workload fitness and computation workload fitness,
- $\boldsymbol{w3 > 0, w4 > 0, w3 + w4 = 1}$.

The deployment fitness contains two parts: communication workload fitness and computation workload fitness. The influences of both parts can be varied by different weight coefficients **w3** and **w4** depending on different systems. Since a pattern can be deployed either on the user's mobile device or on the backend server, each pattern binding has two deployment fitness values: $\mathbf{DF_{server}}$ and $\mathbf{DF_{mobile}}$.

The communication workload is mainly caused by the events transmission between mobile device and backend server. The source of each event resource can be located either in the mobile device or in the backend server. Since a pattern normally uses more than one event, in most situations the events are sourced from different parts of the system. In order to execute the pattern, all the events that are required by the pattern should be transmitted to the part of the system, where the pattern is deployed. Such event transmission is called **Event Forwarding**.

Event Forwarding leads to additional workload of the whole system, especially the mobile device. We use communication workload fitness ***CF*** to indicate such additional workload. The communication workload fitness ***CF*** in (3) depends mainly on the amount of the events resources in a pattern binding that should use **Event Forwarding.** Obviously if most event resources of a pattern binding are sourced from the backend server, from the communication point of view deploying such pattern binding on the server is better than deploying it on a mobile device, since fewer events are required to be forwarded to the mobile device. The frequency of event resources that should be forwarded influences the communication workload as well. Using event forwarding on an event resource with high frequency can cause more communication workload than forwarding two event resources with low frequency. Hence the communication workload fitness **CF** can be calculated as:

$$\boldsymbol{CF}_{(\boldsymbol{pb},\boldsymbol{d})} = \frac{\sum_{s \in S'} f_s - \sum_{s \in S''} f_s}{\sum_{s \in S'} f_{s'}} \tag{4}$$

Where

- ***pb*** is the pattern binding,
- ***d*** is the deployment position (i.e., mobile device or backend server),
- ***f*** denotes the frequency factor of event resources,
- ***S'*** is the set of all event resources used in the current pattern binding ***pb*,**
- ***S"*** is the set of the event resources that need event forwarding,

The value of ***CF*** is located in$[0, 1]$.

Based on some experiments of event forwarding and studies of sensor event frequency, frequency factors are categorized into several groups and defined as following[32]:

$$f = \begin{cases} 1, & frequency \le 1\ event\ per\ hour \\ 2, & 1\ event\ per\ hour \le frequency \le 1\ event\ per\ minute \\ 3, & 1\ event\ per\ minute \le frequency \le 10\ event\ per\ minute \\ 4, & 10\ event\ per\ minute \le frequency \le 1\ event\ per\ second \\ 5, & frequency > 1\ event\ per\ second \end{cases}$$

The result of the communication workload fitness estimates the possible workload of communication caused by the current pattern binding comparing to the all other pattern bindings and deployment positions. In the case all event resources should be transmitted, the value of ***CF*** is 0. If no event resource requires event forwarding, the value of ***CF*** is 1.

The computation workload fitness ***LF*** considers the current computation workload of the deployment positions (i.e., mobile device or backend server). The value of computation workload depends only on the deployment position and is same for all pattern bindings. The computation workload fitness can be calculated as:

$$\boldsymbol{LF_d} = \mathbf{1} - \boldsymbol{L_d} \tag{5}$$

where

- ***d*** is the deployment position (i.e., mobile device or backend server),
- ***L*** denotes the current computation workload of mobile device (or backend server),
- The value of ***L*** is between 0 and 1 presented as percentage.

[32] The values of frequency factors are only for our prototype. For different hardware models of mobile devices, the frequency factors may also have different values.

The computation workload is calculated depending on CPU usage, memory usage and current battery volume (for the backend server the battery volume is always 100%) and can be presented by the following formula:

$$\boldsymbol{L_d = w_{cpu} \times CPU\ Usage + w_{memory} \times Memory\ Usage + w_{battery} \times (1 - Battery\ Volume)} \quad (6)$$

For different configuration of system hardware, different weight coefficients $\boldsymbol{w_{cpu}}$, $\boldsymbol{w_{memory}}$ and $\boldsymbol{w_{battery}}$ can be used to correspond with current hardware.

Therefore according to formulas presented in (1), (2), (3), (4), (5) and (6) for a pattern with **N** pattern bindings, there are maximal $\mathbf{2 \times N}$ possible combination of pattern bindings and deployments. We calculate the fitness values of all $\mathbf{2 \times N}$ combinations and find the pattern binding and deployment with the highest fitness value. This combination should be the optimal solution of the pattern distribution.

8.3 Example

In the previous section we introduced pattern distribution model and the details about distribution fitness calculation. In this section we provide a concrete example to explain the whole distribution process for a better understanding.

Let's continue the Peter's scenario described in Chapter 1: Peter uses the proposed system to monitoring his health status during his fitness training. Considering the historical data and Peter's health records, a special Monitoring Goal Network (MGN) has been defined to monitor Peter's health status, especially for arrhythmia risk. As predefined in MGN, two patterns **P1** and **P2** are required for jogging monitoring:

- **P1** uses speed, temperature and humidity events to detect the situation, which is a danger for a person, who has arrhythmia,
- **P2** detects the symptoms that can be an indicator of an arrhythmia using heart rate event.

In the first step of pattern distribution, **P1** and **P2** will be retrieved from the pattern & event resource ontology by a pattern query. In the second step the system finds all the available event resources for Peter to build the pattern bindings. A total of four different events are required by these two patterns, the following table shows the concrete event resources of each event with detailed attributes.

Table 8-1 Table of all available event resources for patterns

Event Resource	Event Type	Frequency factor	Cost	Source
ER-1	Speed event	4	free	Sensor on smartphone
ER-2	Heart Rate event	4	free	Sensor on smartphone
ER-3	Temperature event	3	free	Sensor on smartphone
ER-4	Humidity event	3	free	Sensor on smartphone
ER-5	Temperature event	2	3€	Web service from weather station (backend server)
ER-6	Humidity event	2	3€	Web service from weather station (backend server)

The sensors that are connecting to Peter's smartphone can provide all four events required by the patterns. In addition, the temperature event and humidity event can be provided by the backend server through a web service of the weather station. However, Peter should pay 3 euros for the service. The frequency of temperature event and humidity events from sensors on smartphone is higher than the frequency of the events from web service.

According to the available event resources the system allocates the event resources in the patterns to create all the possible pattern bindings. As results, **P1** has four pattern bindings. Table 8-2 lists all four pattern bindings for pattern **P1** with detailed event resource allocation. **P2** has only one pattern binding Pb-2, which uses ER-2 as its input.

Table 8-2 Pattern bindings for P1

Pattern binding	Speed event	Temperature event	Humidity event
Pb-11	ER-1	ER-3	ER-4
Pb-12	ER-1	ER-3	ER-6
Pb-13	ER-1	ER-5	ER-4
Pb-14	ER-1	ER-5	ER-6

In the third step the system calculates the **Distribution Fitness** for all pattern bindings and deployment positions. The current system information of Peter's smartphone and the backend server is measured as following:

- $\mathbf{CPU_{Mobile}}$ = 23%
- $\mathbf{Memory_{Mobile}}$ = 60%
- $\mathbf{Battery_{Mobile}}$ =93%
- $\mathbf{CPU_{Server}}$ = 10%
- $\mathbf{Memory_{Server}}$ = 20%

The weight coefficients used in the calculation are defined as following:

- **w1** = 0.5, **w2** = 0.5
- **w3** = 0.2, **w4** = 0.8
- $\mathbf{w_{CPU}}$ = 0.33, $\mathbf{w_{Memory}}$ = 0.33, $\mathbf{w_{Battery}}$ = 0.33

In total, eight distribution fitness values should be calculated. Firstly we calculate the ***Pb-11-server,*** which uses Pb-11 and is deployed on the backend server.

The cost factor of ***Pb-11-server*** is 1, since no paid event resources are used by ***Pb-11-server.***

The communication workload fitness ***CF*** of ***Pb-11-server*** is 0, because all used event resources are obtained from Peter's smartphone and all of them should be forwarded to the backend server.

The current computation workload **L** of the backend server is calculated as following:

$$\boldsymbol{L} = 0.33\ \times 10\% + 0.33\ \times 20\% + 0.33\ \times 0\% = 10\%$$

Hence the computation workload fitness ***LF*** is:

$$\boldsymbol{LF} = 100\% - 10\% = 90\%$$

Based on CF and LF, the deployment fitness can be calculated:

$$\boldsymbol{DF} = 0.2 \ \times 0\% + 0.8 \ \times 90\% = 72\%$$

Finally, the distribution fitness of ***Pb-11-server*** is:

$$\boldsymbol{Fitness} = 0.5 \ \times 1 + 0.5 \ \times 0.72 = 0.86$$

Similarly the distribution fitness for all pattern bindings and deployment positions can be calculated. Table 8-3 shows the calculation results of all pattern bindings and deployment positions with all detailed elements. ***Pb-11-mobile*** has the highest distribution fitness values among all the pattern bindings and deployment positions. Hence in the last step, pattern **P1** should use all the event resources from sensors connecting to Peter's smartphone and be deployed on Peter's smartphone.

Table 8-3 Distribution fitness results of Pattern P1

Pattern binding & deployment	**Cost factor**	**CF**	**L**	**LF**	**DF**	**Distribution Fitness**
Pb-11-server	1	0	0.1	0.9	0.72	0.86
Pb-11-mobile	1	1	0.3	0.7	0.76	0.88
Pb-12-server	0.5	0.22	0.1	0.9	0.764	0.632
Pb-12-mobile	0.5	0.78	0.3	0.7	0.716	0.608
Pb-13-server	0.5	0.22	0.1	0.9	0.764	0.632
Pb-13- mobile	0.5	0.78	0.3	0.7	0.716	0.608
Pb-14-server	0	0.5	0.1	0.9	0.82	0.41
Pb-14- mobile	0	0.5	0.3	0.7	0.66	0.33

After the deployment of pattern P1, the workload of Peter's smartphone has been changed. The new information is measured as following:

- **CPU$_{\textbf{Mobile}}$** = 28%
- **Memory$_{\textbf{Mobile}}$** = 72%
- **Battery$_{\textbf{Mobile}}$** =92%

Table 8-4 Distribution fitness results of Pattern P2

Pattern binding & deployment	**Cost factor**	**CF**	**L**	**LF**	**DF**	**Distribu-tion Fitness**
Pb-2-server	1	0	0.1	0.9	0.72	0.86
Pb-2-mobile	1	1	0.36	0.64	0.712	0.856

The results of the distribution fitness calculation are shown in Table 8-4. The value of distribution fitness for ***Pb-2-server*** is higher than the value for ***Pb-2-mobile.*** Therefore pattern ***P2*** will be deployed on the backend server.

8.4 Conclusion

In this chapter we described the dynamic pattern distribution, which enables resource-aware pattern distribution between user's mobile device and the backend server. Using a four steps distribution model the proposed MCEP finds the best event resources allocation and deployment position for the required pattern. The distribution model has been implemented and the examples have shown how the calculation works. The performance of the dynamic pattern distribution is evaluated and will be described in Chapter 10 (see 10.1.3.).

9 Semantic-based Dynamic Pattern Management

In this chapter we introduce the semantic-based dynamic pattern management, which uses semantic technologies to manage patterns and event resources, in order to achieve dynamic adaptation to different event resources in run time. We develop a pattern and event resource model to manage patterns and event resources of users. Three methods are used for the dynamic pattern adaptation. We also develop algorithms for pattern deployment and adaptation to changes of the availability of event resources in run time.

9.1 Problem statement and overview

Since Complex Event Processing (CEP) technology [Luck01] is used in the proposed MCEP system, patterns play an important role in the real-time personal data processing. As the core feature of CEP technology, patterns define the concrete processing procedure of real-time data through specifying one or more combinations of events [EtNi10].

One of the most important issues in CEP technology is how to model patterns. The patterns used by different CEP engines are defined in different pattern languages. Considering the proposed hybrid architecture of MCEP system, at least two CEP engines are used in the proposed system. In our prototype, Esper [Espe07] is used as CEP engine on mobile devices, whereas another different CEP engine such as ETALIS [AFRS+10] can be used on the backend server. Furthermore, by combining cloud technology, multiple different CEP engines can also be used on backend server infrastructure. Since the patterns used for personal data processing can be distributed to different CEP engines, a general pattern model is required, which is not limited to special pattern language.

On the other hand, a pattern requires several events as inputs. In the proposed MCEP system the events used in patterns are mostly sourced from sensors connecting to users' mobile devices or from external event sources (such as web service) on the backend server. Different event sources can provide the same events in different streams, and each of them is defined as a separate event resource (defined in definition 8.1). As introduced in Chapter 8, to achieve the resource-aware pattern distribution, the concrete event resources for events required by the pattern are allocated through pattern deployment.

However, one important challenge in the pattern management is that sometimes there is no available event resource for an event that is required by a pattern. To make it even worse, the availability of an event resource can change in run time after it has been allocated, for various reasons, ranging from the sensor running out of battery or a disconnection between sensor and mobile device. The former situation makes the pattern unable to be deployed, whereas the latter situation leads to failure of pattern execution.

Hence, in order to ensure the deployment and execution of patterns, the information of event resources of each user should be stored and dynamically updated in run time.

According to the requirement R6 defined in 5.3, we propose to use semantic technologies to achieve dynamic pattern management with the purpose of solving the problems mentioned above: no available event resources of required event and the availability of event resource changes in run-time. The semantic-based dynamic pattern management provides the following functionalities:

- Providing general pattern model for pattern definition, which enables the defined patterns to be used by different CEP engines.
- Managing and dynamically updating information of event resources for each user.
- Annotating event resources' metadata, which enables optimal event resource allocation in the pattern distribution process.
- Providing semantic-based pattern adaptation methods.
- Adapting to the situation of not-available event resource by pattern deployment.
- Adapting to changes of availability of event resource during pattern execution.

In the rest part of this chapter we introduce these functionalities in details.

9.2 Pattern and event resource model

As stated in the previous section a general pattern model is required in the proposed MCEP system for defining patterns in a common format, in order to reduce the costs of defining patterns for different CEP engines in different pattern languages. We develop a tree structured model for patterns based on the pattern model described in [SeSt10]. It supports the common functions for most CEP engines, such as logic operators (i.e., AND, OR, NOT and etc.), temporal operators (i.e., SEQ, time window and etc.) and condition customization.

In addition, the information of event resources of users is required to be modeled and stored for pattern execution. Considering that event resources are tightly connected to the patterns for the personal data processing, we model both patterns and event resources together in one semantic-based model.

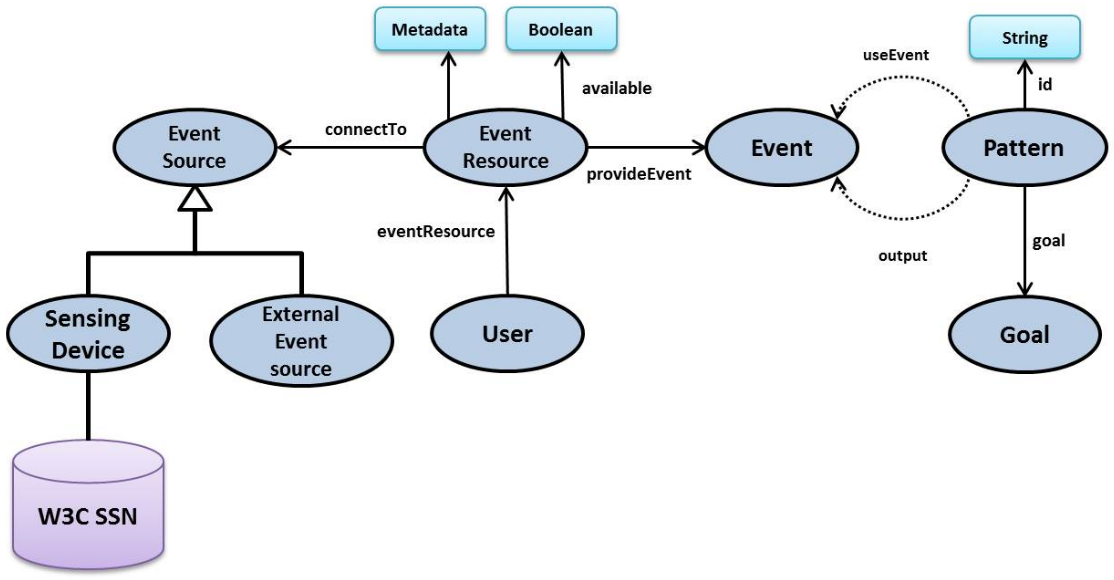


Figure 9-1 Pattern and event resource model

Figure 9-1 illustrates a simplified pattern and event resource model[33]. In the following we introduce the classes and their functionalities briefly.

On the right side is the pattern model including two classes: **Pattern** and **Goal**.

- **Pattern**: This class models the pattern defined for personal data processing. The complete pattern model includes the description of various pattern operators, operands and their relations and can be very complex. Since the pattern modeling is not the key feature of this thesis, it is not introduced here in details. Three important object properties are shown in the figure: "**goal"** indicates the purpose of the pattern. "**useEvent"** is an abstract object property[34] and indicates the input events of the current pattern. "**output**" is also an abstract object property, which is used to indicate the output event of the current pattern. Each pattern has a unique id of a string value and is indicated by data property "**id**".
- **Goal**: This class specifies the purpose of patterns. Different patterns can perform the same detection using different input events or expressions. For example, to detect the average heart rate of a user, the pattern can use either simple heart rate event or complex ECG (Electrocardiography) event as input. The patterns with the same goal perform the same detection and can be replaced by each other.

On the left side is the model for event resources.

[33] This is only the simplified patterns & event resource model. The complete model described in the RDF/XML format is attached in appendix.

[34] An abstract object property is a property that doesn't exist in the real ontology. It expresses a chain of classes and object properties between the domain and the range of the abstract property to simplify the description in the figure.

- **Event resource:** This class defines the concept of event resource. As shown in Figure 9-1 it has two object properties and several data properties. The object property "**provideEvent**" indicates the event type of such event resource. The object property "**connectTo**" indicates the source of current event resource. A set of data properties indicate the metadata of event resources. For example, the data property "**available**" indicates the availability of current event resource using a Boolean value. The other metadata of event resources mentioned in the chapter 8, such as cost and frequency are indicated by other data properties. We don't list all these properties here, since the metadata of event resources can be extended later.
- **Event:** This class denotes the type of events used in the proposed MCEP system. Each instance of this class indicates an MCEP event type, which has been defined in the system.
- **User:** This class indicates the registered users of the system. Each instance of this class is a registered user. The object property "**eventResource**" indicates the event resources of this user.
- **Event Source:** This class indicates the source of event resources. It has two sub classes: **Sensor** and **External Event Source**.
    - **Sensing Device:** This class denotes the sensors, which are connected to users' mobile device and sense the personal data of users. Based on the existing approach for the sensor model, we model the sensors in W3C Semantic Sensor Network (SSN) ontology [CBBG+12]. This class is equivalent to "Sensing Device" in SSN ontology.
    - **External Event Source:** This class indicates the information sources from internet, such as Web service from weather stations or social networks (e.g., Facebook, Twitter). Normally the external event sources are connected to the backend server.

The pattern used for a special use case should be defined in the design time of such use case and be stored in the pattern and event resource ontology with a unique id. In the run time of such use case, the pattern can then be found in the ontology by query with the id of the pattern. However, the patterns, which are the results of the queries in the ontology, cannot be directly deployed to the CEP engines, since the patterns are defined in the MCEP common form and should be translated to the special pattern language required by the CEP engine, to which the pattern will be deployed.

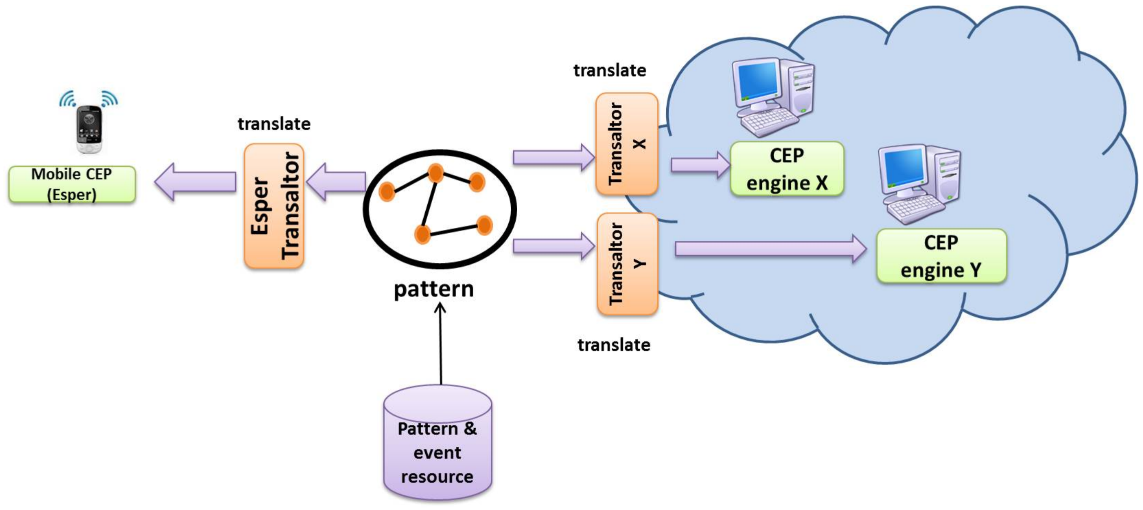


Figure 9-2 Pattern translation

Figure 9-2 shows the pattern translation procedure. The semantic modeled pattern is retrieved by a query with pattern id from the pattern and event resource ontology. According to the deployment of such pattern, it is translated into different pattern languages depending on the CEP engine. For example, in the prototype Esper CEP engine is used on mobile devices, in the case of being deployed on mobile device of user, the pattern is translated into Esper pattern language by an Esper translator. For each CEP engine that uses special pattern language, a pattern translator is required to translate the pattern from MCEP format into engine specified pattern language. In the prototype of MCEP system we provide two translators: one is Esper translator, another is BDPL translator, which translates patterns into BDPL language used by DCEP engine [SSOG12].

9.3 Adaptation methods

Considering the variety of sensors, mobile devices and other equipment, each user may have different events and different event resources. For example, one user may use a sensor, which measures only heart rate (i.e., it can only provide heart rate event), while another user uses a sensor measuring ECG (i.e. it generates an ECG event). The proposed MCEP system can perform the same monitoring for these two users but uses different patterns regarding the different events provided by the users. Hence the available event resources of each user play an important role for pattern deployment. Furthermore the availability of event resources can change during run time for various reasons. Since the change of the availability of event resources can affect the execution of patterns, the proposed MCEP system should be able to adapt to such changes to ensure the execution of patterns. Using proposed pattern and event resource ontology, the information of event resources of each user is stored and updated in run time.

Three methods are developed to achieve dynamic adaptation to such predicaments using querying and reasoning:

- Generating missing event by additional pattern,
- Using available replacement patterns, and
- Using alternative event resources.

In the following subsections we introduce all three methods in details.

9.3.1 Generating missing event by additional pattern

The events used for personal data processing are mostly sensed by sensors and contain the information about the user and the environment that is around the user. The nature of the information is that most of time information is related to each other, which enables some events to be deduced from other events using CEP technology. For example, the current running speed of a user can be deduced from the real-time GPS events. The relations between the events provide the possibility of using additional pattern to achieve adaptation to the situation of missing required events of a certain pattern.

In case where a required event of a certain pattern is missing, other patterns, which can produce this missing event through processing of other available events, can be used to generate this missing pattern in real-time.

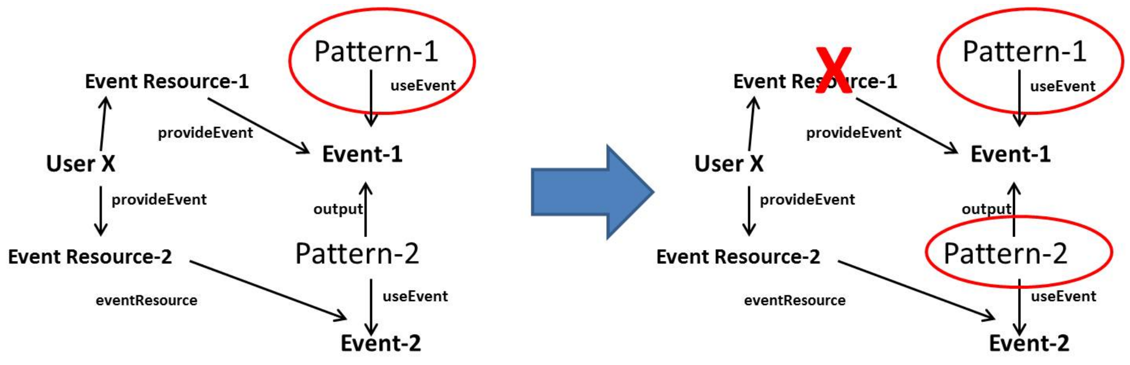


Figure 9-3 Example of generating missing event by additional pattern

Since patterns and event resource are modeled in semantic model, the search for an additional pattern can be easily done with queries and reasoning. Figure 9-3 shows an example of using additional patterns to solve the problem of missing event. According to pre-defined monitoring procedure, **Pattern-1**, which uses **Event-1** as the input event, should be deployed to perform some monitoring tasks for **User X**. However, **Event Resource-1**of **User X,** which provides **Event-1,** is not available at pattern deployment time. Fortunately **Pattern-2** can produce **Event-1** and the input event of **Pattern-2** is **Event-2**, which can be provided by **User X** through **Event Resource-2**. In such case **Pattern-2** can be used as additional pattern to generate **Event-1**.

9.3.2 Using available replacement patterns

The second adaptation method is to use available replacement pattern. Some monitoring tasks can be performed through different methods, which use different patterns and require also different input events. Hence, for such monitoring tasks the replacement patterns can be used, in the case that the pre-defined pattern cannot be deployed duo to missing events.

As introduced in the section 9.2, the "Goal" class indicates the purpose of patterns. The patterns perform the same monitoring task have the same "Goal" and can be used as replacement patterns for each other.

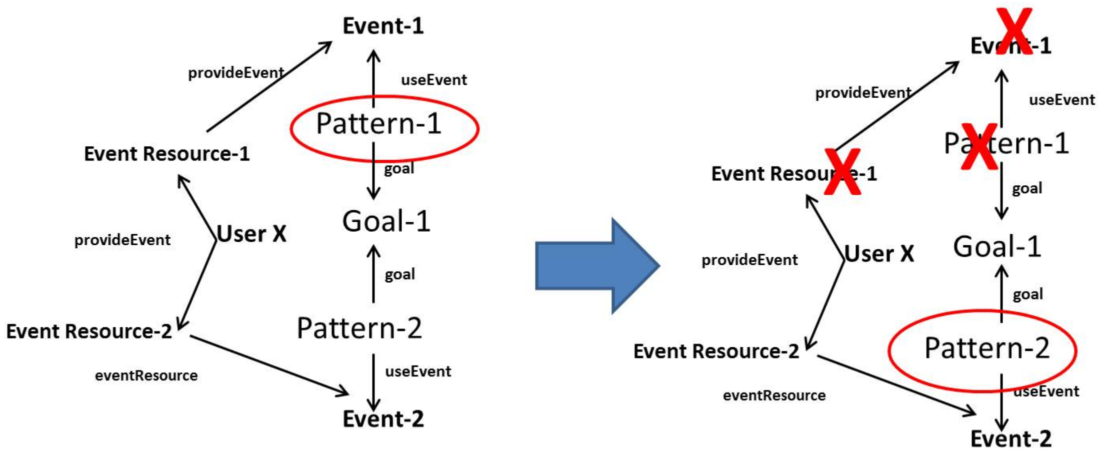


Figure 9-4 Example of using replacement pattern

Figure 9-4 shows an example of how the method works. **Pattern-1** and **pattern-2** have the same goal, which means they perform the same monitoring task. **Pattern-1** uses **Event-1** as input event while **Pattern-2** uses **Event-2** as input event. **Pattern-1** is the pre-defined pattern for achieving the monitoring task in Monitoring Goal Network (MGN). But due to the unavailability of **Event Resource-1** of **User X**, **Pattern-1** becomes unavailable for **User X**, while **Pattern-2** is available. In the case of such situation Pattern-2 can be used as replacement pattern of **Pattern-1**.

9.3.3 Using alternative event resources

The third method is to use alternative event resource. As mentioned in the previous sections, a user can have multiple event resources for one event, which are sourced from different sources. For example, if a user is equipped with an additional sensor with GPS function, considering that most smartphones integrate GPS as a standard sensor, this user has two event resources of GPS events, which are provided by his smartphone and sensor. The multiple event resources enable the proposed system to ensure the execution of the deployed pattern, in case that the assigned event resource becomes unavailable during the run time.

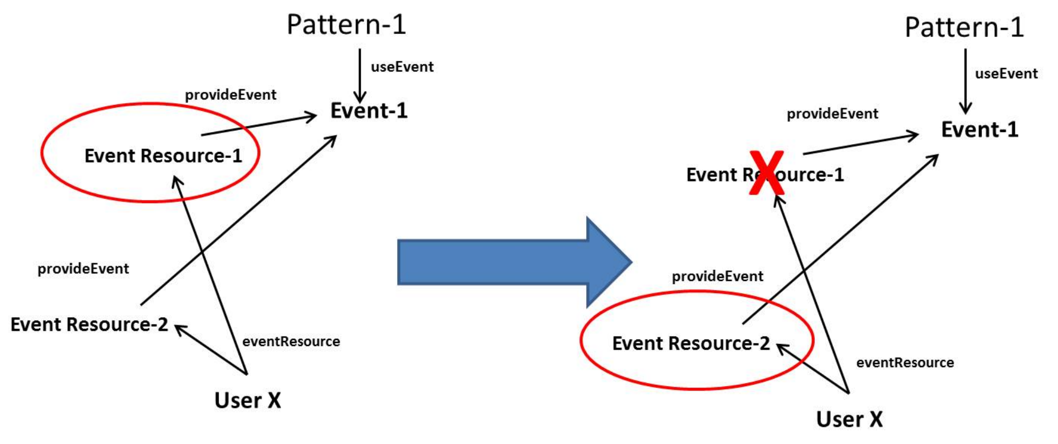


Figure 9-5 example of using alternative event resource

Figure 9-5 illustrates the example of using alternative event resource. **User X** has two event resources, which can produce **Event-1. Event-1** is used as input event by **Pattern-1**. According to the monitoring procedure, **Pattern-1** is required to be deployed. By deploying **Pattern-1**, **Event Resource-1** is selected to be used by **pattern-1**. During the monitoring **Event resource-1** becomes unavailable, due to the disconnection of sensor. In order to ensure the execution of **pattern-1**, **Event Resource-2** is used as an alternative event resource to provide **Event-1** in place of **Event-Resource-1.**

9.4 Algorithms of real-time pattern adaptation

As stated in 9.1, two issues can arise in practice: no available event resource for required event by pattern deployment and the change of the availability of assigned event resource in run time. In this section we introduce the algorithm that combines the proposed adaptation methods to deal with such predicaments.

9.4.1 Adaptation algorithm by pattern deployment

Combining the adaptation methods of using additional pattern and replacement pattern, the algorithm tries to achieve the dynamic adaptation of patterns by pattern deployment to solve the problem of no available event resource for required event.

Algorithm 9-1 Adaptation algorithm by pattern deployment

Adaptation algorithm by pattern deployment

```
Input:   pattern_ID: the id of the pattern that is required to be deployed
         user_ID: the id of the user
01:  boolean available = checkPattern (pattern_ID, user_ID);  // check the availability of pattern
02:  if (available)                       // if pattern is available, deploy pattern and finish algorithm
03:      deployPattern (pattern_ID);
04:      return;
05:  end if
     // method of using additional patterns for missing events
06:  missing_events = getMissingEvents (pattern_ID, user_ID);   // get missing events from pattern
07:  boolean hasAditionalPattern = true;
     // for each missing event, search additional patterns that can produce this event
08:  for each Event ∈missing_events
09:      event_additional_patterns = searchAdditionalPattern (event, user_ID);
10:      Pattern p = getAvailablePattern (event_additional_patterns, user_ID);
11:      if (p == null)                    // if no available additional pattern for one of the missing events
12:          hasAditionalPattern = false;
13:          break;                        // stop loop and jump to line 17
14:      end if
15:      additional_patterns.add (p);
16:  end for
17:  if (hasAditionalPattern)              // if additional patterns for all missing events are found
18:      deployPattern (additional_patterns);   // deploy additional pattern
19:      deployPattern (pattern_ID);            // deploy original pattern
20:      return;                                // end adaptation algorithm
21:  end if
     // method of using replacement pattern
```

```
22: replacement_patterns = searchReplacementPattern (pattern_ID, user_ID);
23: Pattern p = getAvailablePattern (replacement_patterns, user_ID);
24: if (p == null)        // if no replacement pattern is found, throw exception and finish adaptation algorithm
25:     throw PatternUnavailableException;
26: end if
27: deployPattern (p);                                         // deploy replacement pattern
```

Above is the adaptation algorithm for pattern deployment. The input to the algorithm is the id of the pattern that is required to be deployed and the id of the user. The code in line 1 checks the availability of the pattern. If the user can provide all the required events for the pattern, the pattern is directly deployed using the code in line 2 through line 5. The function **deployPattern** uses the algorithm 8-1 to find the best distribution of the pattern.

In case that the pattern is not available, the method of using additional patterns is used firstly. The code in line 6 gets all the missing events of the pattern. The loop from line 8 to line 16 searches additional patterns for each missing event. The function **searchAdditionalPattern** searches additional patterns for a special output event. If additional patterns for all missing events are found, the code in line 17 to 21 deploys the additional patterns and the required pattern. The code in line 10 through 14 checks the availability of additional pattern. If one missing event has no available additional pattern, the code in line 13 stops the loop of searching for additional patterns, since the method doesn't work.

The method of using replacement pattern is used in lines 22 through 27. The function **searchReplacementPattern** searches for the replacement patterns of the required pattern. The found replacement pattern is deployed in line 27. In the case that no replacement pattern is found, an exception will be thrown to indicate that the required pattern cannot be deployed and there is also no alternative solution.

9.4.2 Algorithm of adaptation to change of event resource

To deal with the situation that the availability of an event resource changes in run time, we combine all three adaptation methods and developed an adaptation algorithm to ensure the execution of patterns.

Algorithm 9-2 Algorithm of adaptation to change of event resource

Algorithm of adaptation to change of event resource

```
Input:  event_resource: the event resource that becomes unavailable in run time
        pattern_ID: the id of the pattern that uses the event resource
        user_ID: the id of the user
01: Event event = event_resource.getEvet ();          // get the event type of event resource
    // search the alternative event resource
02: Event_Resource resource = searchAvailableEventResource (event, user_ID);
03: if (resource != null)                             //alternative event resource is found
04:     assignResource (resource, user_ID);           //assign alternative event resource
05:     return;                                       //quit the algorithm
06: end if
    // method of using additional patterns for missing events
07: event_additional_patterns = searchAdditionalPattern (event, user_ID);
08: Pattern p = getAvailablePattern (event_additional_patterns, user_ID);
09: if (p != null)                                    // additional pattern is found
10:     deployPattern (p);                            // deploy additional pattern
11:     return;                                       //quit the algorithm
12: end if
    // method of using replacement pattern
13: replacement_patterns = searchReplacementPattern (pattern_ID, user_ID);
14: Pattern p = getAvailablePattern (replacement_patterns, user_ID);
15: if (p == null)        // if no replacement pattern is found, throw exception and finish adaptation algorithm
16:     throw PatternExecutionException;
17: end if
18: undeployPattern (pattern_ID);                     // undeploy current pattern
19: deployPattern (p);                                // deploy replacement pattern
```

Algorithm 9-2 shows the concrete code of algorithm to adapt to the change of availability of assigned event resource in run time. The inputs include the event resource that becomes unavailable, the id of the pattern using such event resource and the id of the user. The code in line 1 gets the event type that is provided by the event resource. The method of alternative event resource is used firstly in line 2 to 6. The function **searchAvailableEventResource** searches for the current available event resources for a certain event type. If there is available alternative event resource, it is assigned to the CEP engine to ensure the execution of the pattern. In the case that no alternative event resource is found, the algorithm searches for additional patterns to produce required event, which is described in lines 7 through 12. If there is still no available additional patterns for such an event, the algorithm then searches for the replacement pattern for the required pattern in lines 13 through 19. If the replacement pattern is found, the original pattern will be taken out and the replacement will be deployed. Otherwise, an exception of pattern execution will be thrown to inform the system that the deployed pattern doesn't work any longer.

9.5 Conclusion

In this chapter we described the dynamic pattern management based on semantic technologies. We developed a pattern and event resource model to manage the patterns and real-time information of event resources of users. Using query and reasoning function of semantic model, we developed three adaptation methods to achieve dynamic adaptation of patterns. We have also developed two algorithms to solve the issues of no available event resource for required event by pattern deployment and the change of the availability of assigned event resource in run time. The performance of the dynamic adaptation methods and algorithm are evaluated and will be described in Chapter 10 (see 10.1.2).

PART III

Finale

10 Evaluation

In this chapter we present some experimental results obtained with the proposed MCEP system. We have performed various evaluation tests including performance evaluations and use case evaluations in order to assess our approach from different points of view. In performance evaluations we performed several experiments from various aspects regarding the key features of MCEP system described in this thesis. In the use case evaluations, we evaluated two mobile applications, which use the proposed MCEP system, using questionnaire to analyze the real user experience.

10.1 Performance Evaluation

The approach described in this thesis provide a foundation for developing innovative mobile applications for efficient context-aware processing of real-time personal data streams with novel software architecture and several useful features. It is, however, quite a challenge to evaluate the performance of the proposed approach, as the performance of the whole personal data processing system relies not only upon the approach itself, but also on the patterns for the personal data processing and the monitoring procedure modeling (i.e., MGN). The quality of pattern modeling and MGN modeling can heavily affect the performance of the whole personal data processing system and at the same time the quality of pattern modeling and MGN modeling is difficult to measure and control as well.

In order to evaluate the performance of our approach, we propose to evaluate the key features separately. We have designed several experiments for the key features, including the performance of Complex Event Processing on mobile devices, the performance of adaptation methods and algorithms, and the performance of pattern distribution algorithm.

All the experiments presented in this section were carried out on an ASUS Eee Pad Transformer TF101 as the mobile device and a Lenovo ThinkPad T420 laptop as the backend server. ASUS Eee Pad Transformer TF101 has a 1 GHz dual-core Nvidia Tegra 2 processor and 1GB RAM running on Android 4.0.3 system. The Lenovo ThinkPad T420 has one Intel Core i5-2520M processor 2.5GHz, 8GB of RAM running on Windows 7 professional 64-Bit. In the following subsections we present the detailed experiment settings and the evaluation results.

10.1.1 Mobile CEP evaluation

The most important feature of our approach is to use Complex Event Processing on mobile devices to process real-time personal data. We carry out the experiment to evaluate the performance of mobile CEP engine used in the proposed MCEP system. As introduced in Chapter 6, the mobile CEP engine is an android-based Esper engine [Bade10, Espe07]. The experiment shows the processing capability of the mobile CEP engine and investigates the effect from a number of event streams and a number of patterns. The results of the experiment are shown in the form of throughput. As presented in [EtRS11], there exist many different definitions for throughput. Detailed throughput can be measured in following ways:

- Input throughput: it measures the number of input events that the system can process within a given time interval.
- Processing throughput: it measures the total processing time divided by the number of events processed within a given time interval.
- Output throughput: it measures the number of events that are emitted to event consumers within a given time interval.

In the experiment we adopt the input throughput. The whole outputs from all tests have been validated and it ensures that the system produces the consistent and correct results.

In the experiment we have implemented an experimental mobile application by using the proposed MCEP system along with an event stream generator and a pattern generator. The event stream generator produces random events belonging to several streams, where the generated numbers can be configured in the experiment application. The pattern generator generates random patterns using the events produced by the event stream generator.

We evaluate the performance of mobile CEP by measuring the input throughput with the variation of stream numbers and pattern numbers. The concrete experiment settings are as follow:

- 1000 events are produced and pushed to the mobile CEP engine continuously;
- Ten event groups are used, where events are from 1 to 10 different streams;
- The number of deployed patterns varies from 1 to 20.

The results of the performance evaluation are shown in Figure 10-1 and Figure 10-2. Figure 10-1 shows the event throughputs being plotted against different numbers of deployed patterns. Figure 10-2 shows the event throughputs grouped by stream numbers. The Y-axis measures the event throughput achieved by the mobile CEP. The X-axis in Figure 10-1 indicates the number of the deployed patterns. The X-axis in Figure 10-2 shows the number of the event streams. For each configuration in the experiment we have done the evaluation tests three times and the event throughput shown in the results are the average values of all the tests.

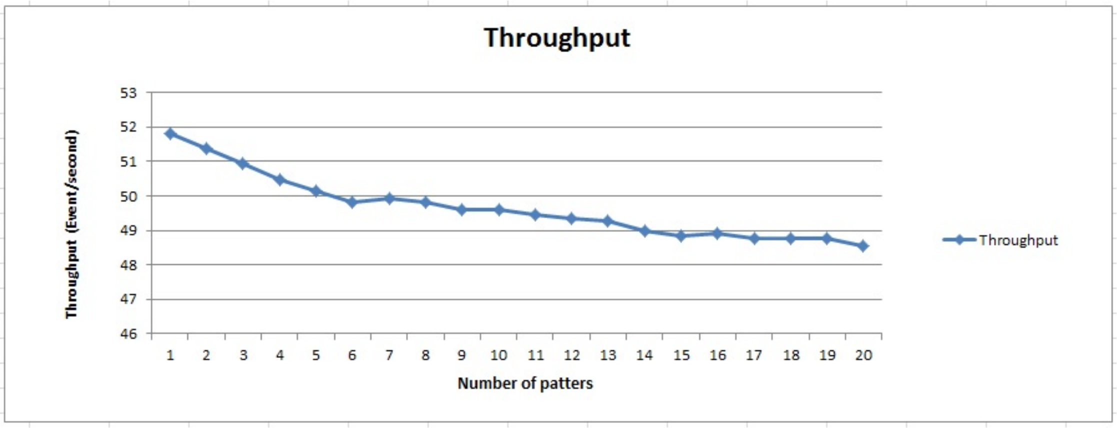


Figure 10-1 Experiment result for mobile CEP performance - pattern based throughput

As shown in Figure 10-1 the event throughputs of mobile CEP are around 50 events per second. The highest event throughput is near 52 events per second, which appears at 1 deployed pattern. The lowest event throughput is about 48.5 events per second by 20 deployed patterns. It is obvious that as the number of deployed patterns increases, the event throughputs reduce correspondingly. However, the reduction is not proportional to the number of deployed patterns. The average reduction rate of event throughputs in the experiment is about 5%, which is reasonable.

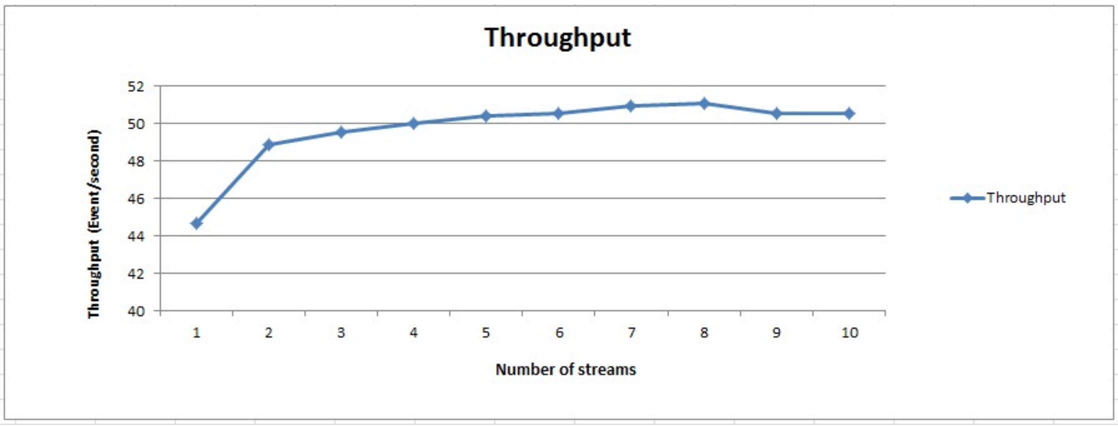


Figure 10-2 Experiment result for mobile CEP performance - stream based throughput

Figure 10-2 investigates the relation between the number of event streams and the event throughputs. As shown in the figure the average throughput by the different numbers of event streams is around 50 events per second. The lowest throughput is located in 1 event stream and the highest throughput is at 8 event streams. The throughputs by most settings with different number of streams show no obvious difference. When there is only one event stream, the events in this stream are used by all patterns, hence each input event will be calculated for all patterns, which can cause the lowest event throughput in the experiments. As the number of streams increases above 4, the measured event throughputs don't show further noticeable variation of the stream numbers. In general, the number of event streams doesn't seem to affect the performance of mobile CEP engine.

However, due to the resource limitation of mobile devices the event throughput of mobile CEP engine is relatively low comparing to the CEP engines that are running on desktop computers or servers. As shown in the figure the average event throughput of mobile CEP engine is only about 50 events per second. The mobile CEP engine with such throughput can't afford the complex processing tasks requiring a large number of various events with high frequency. In addition the increasing number of deployed patterns can also reduce the throughput of mobile CEP engine. Therefore, the number of patterns that are deployed on the mobile CEP engine should be limited in order to ensure the reliable event processing on mobile devices. This result provides further evidence in arguing for using hybrid software architecture and dynamic deployment for achieving scalable Complex Event Processing on mobile devices, as the approach in this thesis has proposed.

10.1.2 Dynamic adaptation evaluation

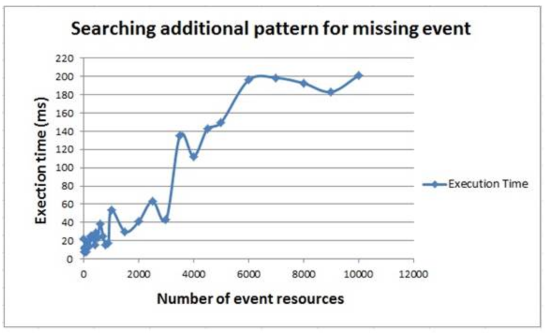
In this subsection we describe the performance evaluation of dynamic adaptation to real-time event resource changes described in Chapter 9, which is an important feature of this thesis. We evaluate three basic adaptation methods and two adaptation algorithms by measuring performance in terms of execution time.

To run the tests, we have generated several test data sets (i.e., pattern and event resource ontologies) with different numbers of event resources, users, event types and patterns, in order to evaluate the performance of adaptation methods and algorithms for different conditions. The detailed information of data sets is shown in the following table:

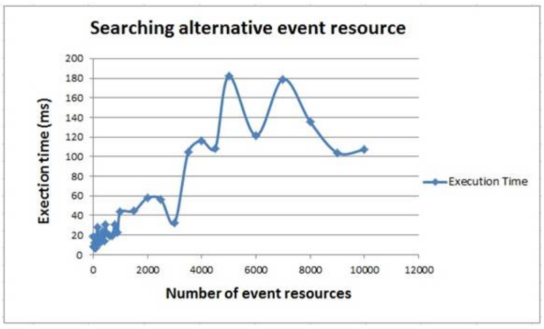
Table 10-1 Data sets for dynamic adaptation evaluation

Index	1	2	3	4	5	6	7	8	9	10	11	12
Number of users	1	2	3	4	5	6	7	8	9	10	10	20
Number of event resources	10	20	30	40	50	60	70	80	90	100	150	200
Number of individuals	632	643	654	665	676	687	698	709	720	731	786	846
Index	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Number of users	20	30	30	40	40	50	50	50	60	60	60	70
Number of event resources	250	300	350	400	450	500	600	700	800	900	1000	1500
Number of individuals	896	956	1006	1066	1116	1176	1281	1381	1481	1581	1795	2305
Index	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**
Number of users	80	100	130	160	190	220	250	300	400	500	600	700
Number of event resources	2000	2500	3000	3500	4000	4500	5000	6000	7000	8000	9000	10000
Number of individuals	2904	3424	4060	4590	5211	5741	6462	7506	8718	9919	11130	12339

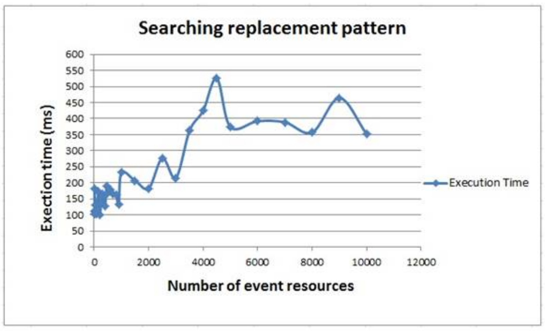
We start the evaluation of dynamic adaptation by presenting the experiments of three basic adaptation methods described in Chapter 9. We have executed each method for 10 times using all the data sets listed in the above table and measure the average of execution time. Figure 10-3 shows the experiments results of all three basic adaptation methods. The X-axis shows the number of event resources in the active data set, as listed in the Table 10-1. The Y-axis presents the execution time used by the adaptation method in millisecond.



(a)



(b)



(c)

Figure 10-3 Experiment results for basic adaptation methods: (a) searching additional pattern for missing event (b) searching alternative event resource (c) searching replacement pattern

Specifically, Figure 10-3 (a) shows the execution time measurement for searching additional pattern for missing event. As shown in the figure, the adaptation method of searching additional pattern for missing event executes very efficiently by small data sets. It uses only approximately 30 milliseconds for less than 1000 event resources. For the data sets that contain more event resources (1000 up to 3000 event resources), the adaptation method uses more time in around 40 to 60 milliseconds. For the large data sets that contain even more numbers of event resources (i.e., more than 6000) the execution time prolongs and varies at approximately 190 milliseconds.

Figure 10-3 (b) illustrates the results of experiments for searching alternative event resources. For the data sets with small number of event resources the execution time is approximate 20 milliseconds. In the case that the data sets contain large numbers of event resources, the execution time varies in the range between 100 milliseconds and 180 milliseconds.

Figure 10-3 (c) shows the results of experiments for searching replacement pattern. The minimal execution time of this method is approximate 100 milliseconds and the maximal execution is about 500 milliseconds. For small data sets the average execution time is about 150 milliseconds and for large data sets the average execution time is around 400 milliseconds.

While the execution time of all three adaptation method increases as the number of event resources increases, all adaptation methods can be executed within 500 milliseconds for the largest data set. The results of the experiments show that the performance of three basic event adaptation methods can meet the real-time requirement for real-time personal data processing. The variation of execution time may be caused by the different structures of data sets.

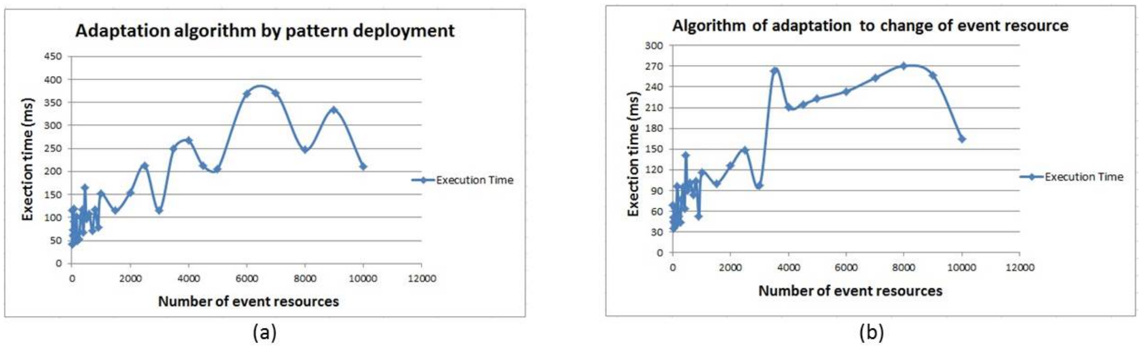


Figure 10-4 Experiment results for adaptation algorithms: (a) Adaptation algorithm by pattern deployment (b) Algorithm of adaptation to change of event resource

We also have performed the experiments for evaluating the performance of two adaptation algorithms. Each algorithm has been executed 10 times using all data sets and measured the average execution time. Figure 10-4 illustrates the experiment results of both adaptation algorithms. The X-axis shows the number of event resources in the active data set and the Y-axis presents the execution time required by the adaptation algorithms in millisecond.

Figure 10-4 (a) shows the results of experiments for adaptation algorithm by pattern deployment. The execution time varies between 50 milliseconds and 370 milliseconds. The average execution time for small data sets is approximately 100 milliseconds. As the number of the event resources increases, the execution time of the algorithm increases correspondingly but slowly. The average execution time for the large data sets is about 300 milliseconds.

Figure 10-4 (b) shows the results of experiments for the algorithm of adaptation to change of event resource. For the data sets containing less than 3000 event resources, the algorithm shows fairly good performance: it reacts to the changes of event resources within 150 milliseconds. In the case that the data sets contain more event resources, the algorithm needs more time to find the solution. The average execution time for the large data sets is then approaching approximately 240 milliseconds.

Overall, the performances of both algorithms are reasonably good and meet the real-time requirements for personal data processing. Large data sets require more execution time but at very slowly rates as the number of event resources increases. The variations of execution time may also be affected by the different structures of data sets.

10.1.3 Pattern distribution algorithm evaluation

In this subsection we describe the experiments of dynamic pattern distribution algorithm. We have performed two experiments in order to evaluate the algorithm from different aspects: the first experiment is to evaluate the performance of resource awareness achieved by the algorithm and the second experiment is to evaluate the execution performance regarding the execution time used by the algorithm.

In the first experiment, we compare the proposed dynamic pattern distribution algorithm with two other distribution algorithms in order to assess the advantages of the proposed resource awareness.

The comparisons of the distribution algorithms are as follows:

- Event source-based distribution algorithm: this algorithm selects the pattern-binding, which has the lowest number of forwarded events, and distributes the pattern to the part of the system that is the source of the most events.
- Random distribution algorithm: this algorithm randomly selects a pattern binding of the pattern, which is to be deployed, and distributes it to either mobile device of user or backend server without considering any other factors.

In the experiment we have created a use case with 40 events and 50 patterns. Each pattern uses at least two and at most eight different events. Events have different sources, 50% events can only be obtained from user's mobile device, 30% events can only be obtained from the backend server and 20% events can be obtained either from user's mobile device or from the backend server. Overall more than 50% events have more than one event resources. In addition, no cost is required for all events.

In order to test the performance of the algorithm regarding the different workload situations of mobile devices, two settings of mobile devices are selected in the experiments:

- Low workload settings: user's mobile device runs only the necessary apps and with almost full battery (CPU usage < 20%, Memory usage < 50%, 100% battery).
- High workload settings: user's mobile device has already run some apps and the battery has only ¼ remaining capacity (CPU usage is about 50%, Memory usage is about 60%, 25% battery).

In both settings the backend server always runs under the same conditions (CPU usage is about 20% and memory usage is about 30%).

For each setting, two experiments regarding two different pattern groups have been performed. One pattern group contains only five patterns and the other pattern group contains thirty patterns, which simulates both the simple (only a small amount patterns are required) and the complex (a large amount of patterns are required) use cases.

As described in Chapter 8, the dynamic pattern distribution algorithm can calculate pattern distributions according to different emphasis by varying the weight coefficients of the algorithm. In the two experiments, we focus on computation workload and use the following weight coefficients in dynamic pattern distribution.

- w1 = 0, w2 = 1. 0, since no cost is required for all events, the weight coefficient for cost factor is hence set to 0.
- w3 = 0.2, w4 = 0.8, the computation workload plays more important role
- w_{cpu} = w_{memory} = $w_{battery}$ = 1/3, where CPU usage, memory usage and battery volume all have the same importance. (Hence the initial workload for low workload setting is about 20% and for high workload setting is about 60%)

All experiments have been performed for 5 iterations and generated new pattern groups randomly for each iteration. As results we calculate the average values of the workload of user's mobile device by the three distribution algorithms.

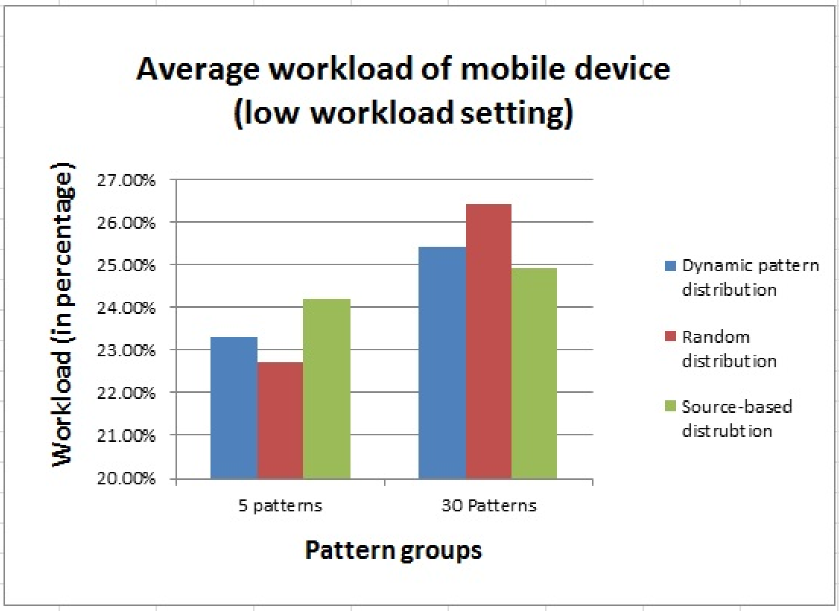


Figure 10-5 Experiment results for pattern distribution algorithm: low workload setting

Figure 10-5 shows the experiment results using the low workload setting. The X-axis shows two patterns groups. The Y-axis presents the computation workload of mobile device in percentage, which is calculated according to formula (6) in Chapter 8. In the experiment of 5-patterns group the random distribution algorithm shows the best performance among all three algorithms, which causes the lowest workload on the user's mobile device. The reason behind this result is that the random distribution algorithm distributes patterns to the mobile device and to the backend server almost equally, whereas the dynamic pattern distribution algorithm and event source based distribution algorithm distributed most patterns on the mobile device in order to reduce the communication workload, since the most events used in the patterns are sourced from the mobile device.

In the experiment of 30-patterns group, the source-based distribution algorithm has performed best among three algorithms. The proposed dynamic pattern distribution algorithm follows up with only very little difference and the random distribution algorithm has the highest workload. The source-based distribution algorithm distributes fewer patterns on the mobile device than our algorithm and has the least communication workload. Due to the low computation workload, the proposed dynamic pattern distribution algorithm distributes the most patterns to the mobile device and the correlated event communication causes additional workload. Although the random distribution algorithm distributes the fewest patterns on the mobile device, the random pattern binding and deployment position selection require much extra event communication, which causes high computation workload.

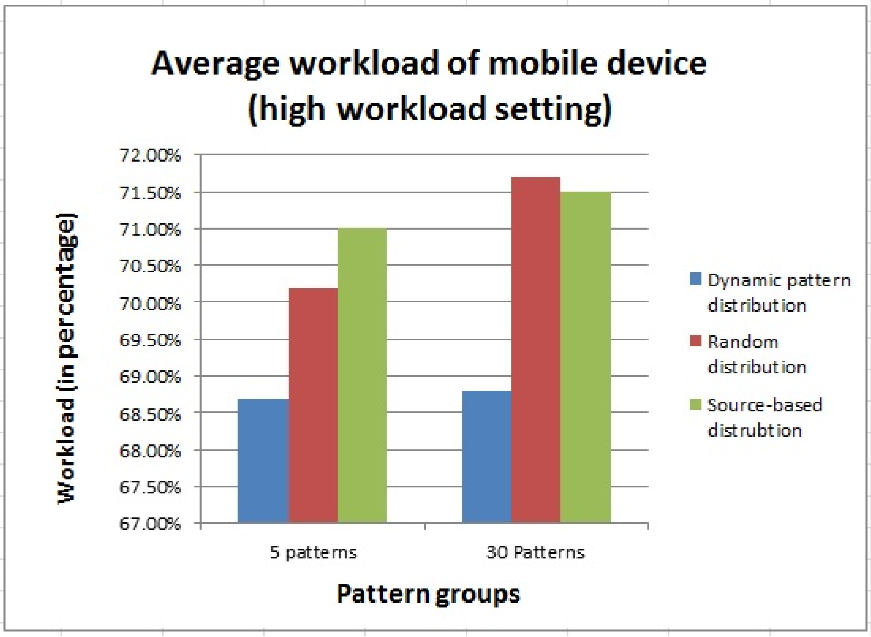


Figure 10-6 Experiment results for pattern distribution algorithm: high workload setting

Figure 10-6 shows the results from the experiment using the high workload setting. It is very obvious that the proposed dynamic pattern distribution algorithm has performed the best in both 5-patterns group and 30-patterns group. The random distribution algorithm follows up in the second position in the 5-patterns group and the source-based distribution algorithm comes in the third. Because the dynamic pattern distribution distributes the most patterns to the backend server considering the high computation workload of the mobile device, while the other two algorithms don't change their distributions. In the 30-patterns group the source-based distribution algorithm has performed better than the random distribution algorithm, since it requires the least event communication.

In summary, the proposed dynamic pattern distribution shows advantage of pattern distribution for mobile devices, especially in high workload situations. Through resource awareness it distributes most patterns to the backend server, which enables user's mobile device to avoid high workload. Hence it prevents the current application from blocking other applications due to use of too much computing resources and extends the battery lifetime.

In the second experiment we evaluate the performance of dynamic pattern distribution algorithm execution. Considering that the pattern distribution should be calculated in real-time, the performance of algorithm execution also plays an important role. In order to assess the execution performance of the algorithm we perform the experiment measuring the execution time of the algorithm regarding different complexities of patterns.

In the experiment we have implemented a pattern generator, which generates random patterns with different complexities indicated by the number of different events used in a pattern. In the experiment, following settings are used:

- 9 pattern groups are used, whose patterns use 2, 3, 4, 5, 7, 10, 12, 15 and 20 different events.
- Each pattern group contains 50 random patterns.
- Each event has at least one event resource and three event resources at most.

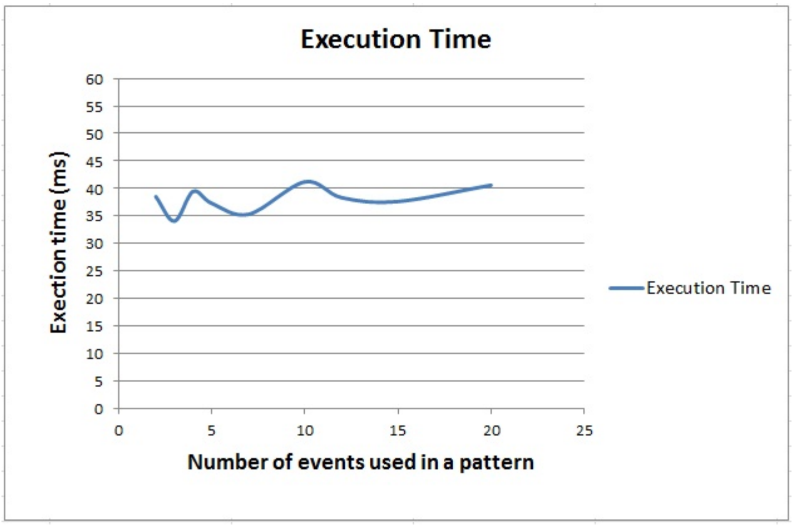


Figure 10-7 experiment results for execution time of dynamic pattern distribution algorithm

We record the execution time of all patterns for all pattern groups and calculate the average execution time for each pattern group. The results are presented in Figure 10-7. The X-axis shows the number of different events that are used in a pattern, which indicates the complexity of patterns. The Y-axis presents the execution time in millisecond.

As shown in the figure, the execution of the dynamic pattern distribution algorithm performs quite stably and efficiently. The execution time varies between 35 milliseconds and 40 milliseconds and shows almost no increase with the rise of the complexity of pattern.

The overall performance of algorithm execution is quite efficient and is sufficient to meet the real-time requirements of our approach.

10.2 Use Case Evaluation

In the last section we show the evaluation of some key features of our approach. In this section we evaluate our approach in real use cases, which mean that the approach is being evaluated in real mobile applications. We evaluate the mobile applications in order to assess the advantage and performance of our approach for developing real-time personal data processing mobile applications. Firstly we introduce the concrete use cases, in which our approach is used. Then we describe the design of use case evaluation. At last we show the results of the use case evaluation. Due to some technique reasons, the use case evaluation is a very preliminary evaluation.

10.2.1 Use cases

The approach described in this thesis has been used as a part of the two real mobile applications, namely My Cardio Advisor (MCA) and AlarMe, in m-Health and m-Fitness domain respectively. The two mobile applications have been developed by Nissatech innovation center[35]. There are also several deployments of these mobile applications in real-world settings.

10.2.1.1 My Cardio Advisor (MCA)

My Cardio Advisor (MCA)[36] is an intelligent interconnected mobile solution to the dynamic, real-time monitoring of complex health situations. It combines wearable and mobile sensing, linked both open data and social network data to enable the real-time monitoring for cardio patients. In the MCA system, the full version (complete) MCEP system has been used for real-time personal data processing.

[35] http://nissatech.com/
[36] http://nissatech.com/technologies/mycardioadvisor/

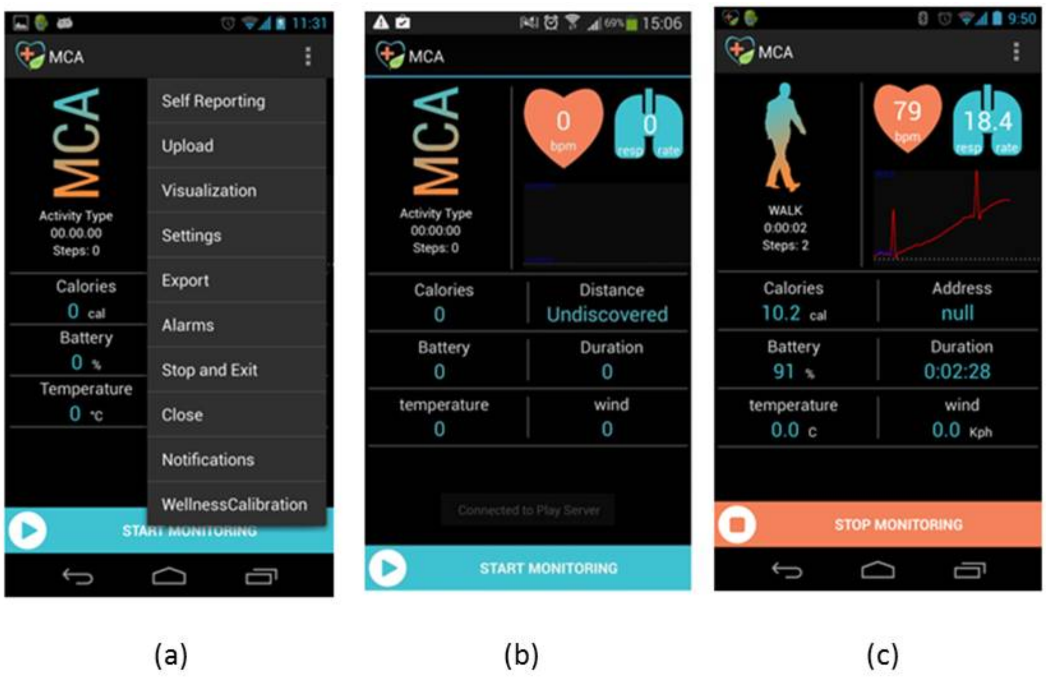


Figure 10-8 MCA mobile application: (a) setting view, (b) start view and (c) monitoring view

Figure 10-8 shows the screenshots of the MCA mobile application, including the setting view, the start view and the monitoring view. As shown in the monitoring view, the MCA app monitors the activity, heart rate, respiratory rate in order to prevent patients from suffering cardio attacks. The MCA system has been deployed as a remote patient monitoring system in a hospital in Hungary.

The MCA application can be used not only for patient monitoring, but also for fitness monitoring. One of the use cases in fitness monitoring domain is jogging monitoring. The MCA app monitors the health status (heart rate, respiratory rate, etc.) of joggers to avoid dangerous situations during training.

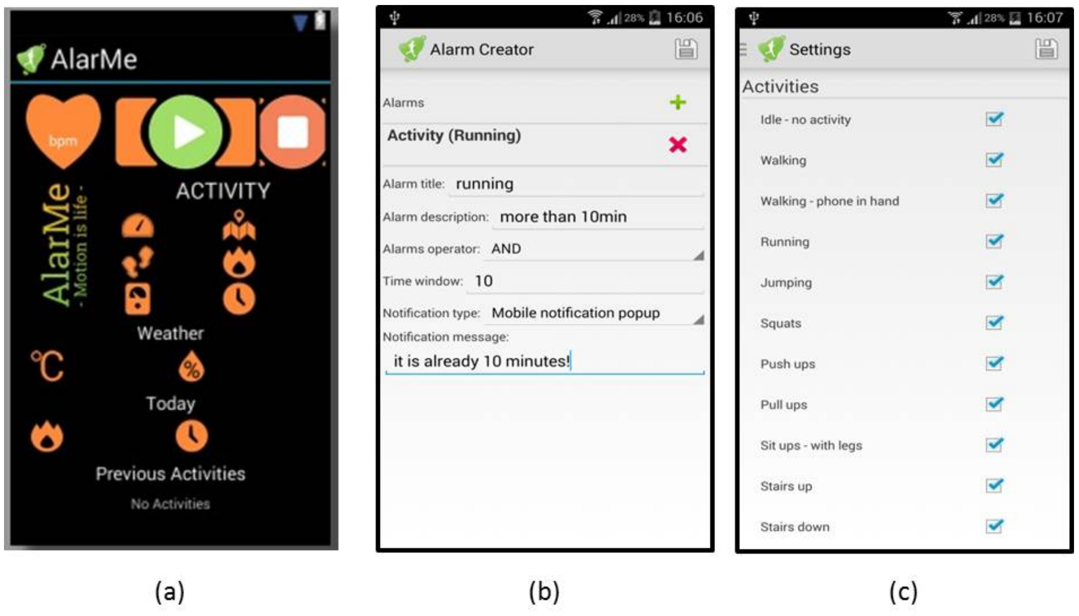
The most important features of MCA include:

- Real time processing and management of personal Big Data;
- Real time processing of patient's ECG and HR;
- Real time Doctor – Patient communication;
- Real time alarms and recommendations;
- Real time physical activity tracking;
- Anticipation of health problem;
- Self-reporting (medications and therapy, food, alcohol and cigarettes, physical activity);
- Full contextualization and situational awareness;
- Full data history, visualization and statistics;
- Full customization,

10.2.1.2 AlarMe

AlarMe[37] is a holistic mobile application that helps users improve and manage their fitness activity performance, well-being, safety or business efficiency. It monitors the fitness activities of users and fires alarms according to different patterns, which are defined by users based on their individual requirements. The mobile-only MCEP variant has been used in AlarMe application providing real-time alarm detection.

Figure 10-9 shows the screenshots of the AlarMe mobile application, including the start view, the alarm creation view and the setting view. As shown in the alarm creation view, users can create their own individual alarms using a combination of different conditions including fitness activity, environment information (e.g., temperature, humidity and location). The defined alarms will be translated into patterns and deployed in the proposed MCEP system.



Alarms

Activity (Running)

Alarm title: running

Alarm description: more than 10min

Alarms operator: AND

Time window: 10

Notification type: Mobile notification popup

Notification message:

it is already 10 minutes!

Activities

Idle - no activity	✓
Walking	✓
Walking - phone in hand	✓
Running	✓
Jumping	✓
Squats	✓
Push ups	✓
Pull ups	✓
Sit ups - with legs	✓
Stairs up	✓
Stairs down	✓

Figure 10-9 AlarMe mobile application: (a) start view, (b) alarm creation view and (c) setting view

The main features of AlarMe mobile application include:

- Fully personalized activity model;
- 24/7 Activity tracking;
- G-map position tracking;
- Detecting the quality and quantity of activity;
- Personalized calories burned;
- Number of steps;

[37] http://nissatech.com/technologies/wellnessnova/

- Speed & Distance;
- Idle alert and other customized alarms;
- Visualization and statistics, full data history;
- Smart alarms, Recommendations (notifications);
- Full customization.

10.2.2 Evaluation design

Since the proposed MCEP system has been successfully integrated in two mobile applications and the performance of the mobile applications depends on the integrated components including proposed MCEP system, we evaluate the whole mobile applications in order to assess the quality of our approach.

Since the both MCA and AlarMe applications are developed and tested by the partner outside Germany, it is difficult for us to set up complex use case evaluations. So we have to evaluate the applications very preliminary using questionnaire. We have designed a questionnaire (see Appendix III) for the users of the two mobile applications, in order to evaluate the mobile applications. The questionnaire contains 13 questions and each question has five optional answers, indicating excellent (5 points), good (4 points), average (3 points), below average (2 point) and poor (1 point). The questionnaire evaluates the mobile application in four aspects:

- Usability
- Reliability
- Robustness
- Efficiency

According to [Niel03] ***usability*** is a quality attribute that assesses how easy and pleasant the human-made objects are to use. The evaluation about the usability in the use cases aims to find out, whether the mobile applications are easy to install and configure, especially the setting of connection to sensors, which are important for our approach.

Software Reliability is an important attribute of software quality. According to [ANSI91] and [Lyu96] software ***reliability*** is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. However, software reliability is hard to achieve, because the complexity of software tends to be high and the high complexity may lead to conflicts between different components, which can reduce the reliability of software. In our evaluation we have used questionnaires to collect feedback for the correctness of alarm or recommendation regarding the timeliness and usefulness, with the purpose of measuring the reliability of the whole mobile application.

ANSI [ANSI91] and IEEE has defined ***robustness*** for software as the degree to which software can function correctly in the presence of invalid inputs or stressful environmental conditions. The robustness shows the ability of software to cope with errors during the execution. In our evaluation we focus on the stability of sensor connections and the fault-tolerance of mobile applications.

Software ***efficiency*** is defined as the degree to which a system or component performs its designated functions with minimum consumption of resources [ANSI91]. The evaluation of the efficiency for our use cases focuses mainly on the required resources of mobile devices, especially the battery and internet, and the effect on the performance of the other applications.

Although the proposed MCEP system has been successfully integrated into two mobile applications, the quality of the mobile applications depends not only on our approach, but also on many other factors, such as the mobile application implementation, the quality of pattern design, the quality of monitoring model definition and etc. The questionnaires described above evaluate the quality of the mobile applications, which may reflect the quality of our approach, but may also be affected by other factors that should be taken into account.

10.2.3 Evaluation results

In this subsection we show the results of the use case evaluation regarding the feedbacks of the questionnaire described in the last subsection.

For the evaluation of mobile application we recruited 20 participants, including 13 users of MCA mobile application and 7 user of AlarMe application. All participants had used mobile applications for more than one week.

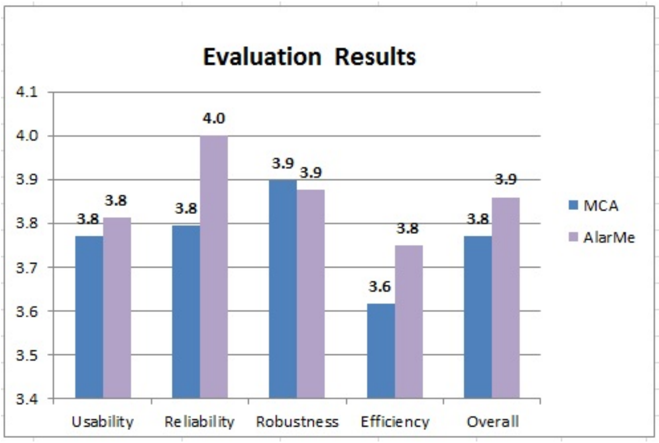


Figure 10-10 Use case evaluation results

As the optional answers of each question in the questionnaire ranges from 5 (excellent) to 1 (poor), the maximum score of the evaluation results is 5 and the minimum score is 1. Figure 10-10 shows the evaluation results of both use cases regarding four aspects. In general the MCA and AlarMe have got good evaluation results, the overall score is near 4 points, which is considered as good. In detail, on the whole the MCA application has the score of 3.8 and AlarMe application gets even a better score of 4.0. Both applications have been evaluated as above the average and almost reach the level of good.

Regarding the four detailed evaluation aspects, both applications get the lowest score in efficiency, while the scores in usability, reliability and robustness almost reach the level of Good. The reason of weakness on the efficiency aspect may be caused by high computation requirements of the mobile applications. Regarding the comparison of the two applications, AlarMe shows better performance in reliability and efficiency. Considering that the monitoring task of cardio risks of MCA is much more complex than AlarMe, and AlarMe uses mobile-only variant of the MCEP system and provides relative simple functions, the evaluation results are reasonable.

We analyze each aspect in detail as follows.

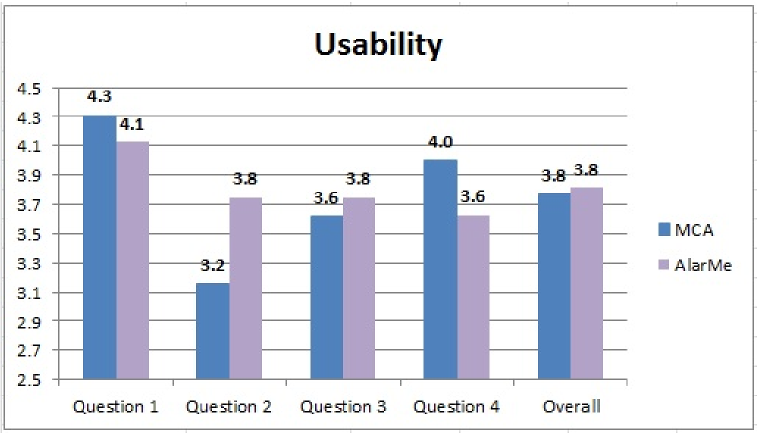


Figure 10-11 Use case evaluation results (Usability)

Figure 10-11 shows the detailed evaluation results about the usability of the mobile applications. As shown in the figure both applications get high score from question 1. AlarMe gets higher score than MCA from question 2 and question 3, which are about application configuration and sensor connection setting. The issue may be caused by additional sensor connection, which is required by the MCA application (heart rate sensor is required for cardio monitoring). The MCA scores better for question 4, indicating that MCA provides better feedback interface. As the AlarMe application has simple functionality and uses additional sensor only optionally, it gets better results in usability evaluation.

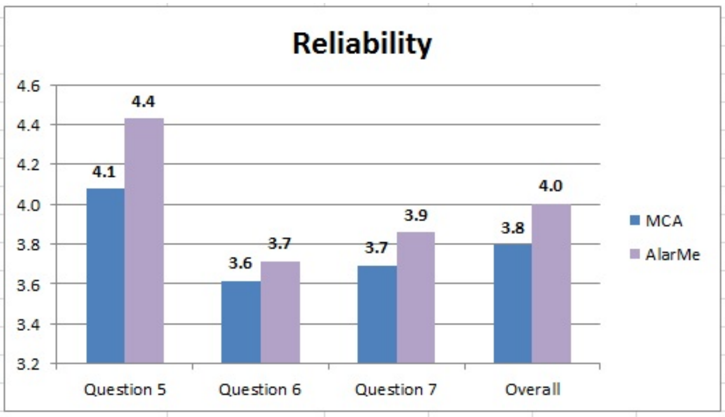


Figure 10-12 Use case evaluation results (Reliability)

Figure 10-12 displays the evaluation results of the reliability. Obviously AlarMe gets better results than MCA. The reason may be due to the fact that the patterns used in MCA to monitor the cardio risk are much more complex than the alarm patterns defined by users in the AlarMe application. In the evaluation results the timeliness of the data processing (question 5) shows good performance, which means that the personal data streams have been successfully processed in real-time. In comparison to the timeliness, the correctness and usefulness of the alarms/recommendations (question 6 and question 7) have been doubted by the participants. However the correctness and usefulness depend mainly on the quality of patterns, which are used to process the personal data and the monitoring procedure definition (in MGN), rather than depend on the proposed MCEP system. Hence the issue can be solved by improving pattern quality and monitoring procedure definition.

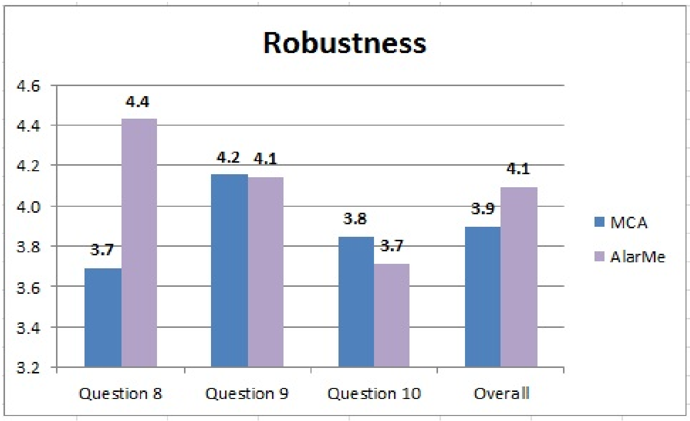


Figure 10-13 Use case evaluation results (Robustness)

Figure 10-13 shows the evaluation results of the robustness for both applications. Overall, AlarMe shows better robustness than MCA. The main different is located in question 8, which is related to the connection stability of sensors. Since MCA uses more sensors than AlarMe and requires an additional heart rate sensor, the users of the MCA application have encountered more problems in relate to sensors than the users of the AlarMe application.

Both applications get high scores from question 9, which indicate that the quality of application implementation is good. Question 10 concerns the run-time issue, which is not caused by errors. Such issue is probably caused by the incorrect definition of patterns. As the patterns (alarms) in AlarMe are defined by users themselves, it leads to more problems in the pattern definition.

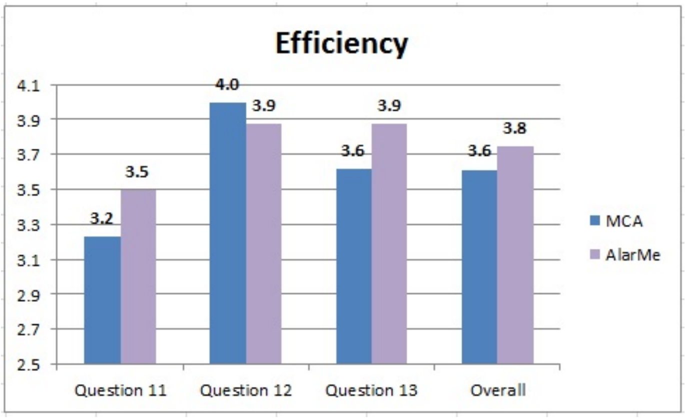


Figure 10-14 Use case evaluation results (Efficiency)

Figure 10-14 shows the evaluation results of the use cases on efficiency. The main issue is the quick battery consumption (question 11). Since the real-time data processing requires continuous execution of applications and leads to high energy requirement, the evaluation result seems to be reasonable. For question 13, since AlarMe uses mobile-only MCEP variant and requires no internet, it gets better results than MCA.

10.3 Conclusion

In this chapter we describe the evaluation of the proposed MCEP system in different aspects including performance and use case. The processing capability of mobile CEP has been evaluated for performance. The results show that the CEP engine on mobile devices is capable of processing real-time data, but not sufficient for some complex processing tasks, which requires a large number of patterns. The evaluation results of dynamic adaptation indicate that the performance of adaptation meets the real-time requirement of the MCEP system. The evaluation of dynamic pattern distribution algorithm has also been presented, and the results show that the dynamic pattern distribution prevents the MCEP system from using too much resource on user's mobile devices, especially in the high workload situation. In addition, the evaluation results of execution time of the distribution algorithm show the efficiency of the algorithm.

In the use case evaluation, the propose MCEP system has been used in two mobile applications. The results of the questionnaire evaluation validate the quality and usefulness of the approach as proposed in this thesis in real use cases.

11 Conclusion

The objective of this thesis is to develop a foundation for the development of innovative mobile applications for efficient context-aware processing of real-time personal data streams by taking into account the resource limitation on mobile devices. We conclude by summing up the results that have been accomplished towards that goal. Finally, we give an overview of future research topics that, in our view, represent the prospective direction of our work.

11.1 Summary

The main motivation behind this thesis came from two sources: firstly the expansion of mobile devices and the rise of sensors, especially the wearable sensors like Zephyr BioHarness 3 sensor [Zeph12], Nike+ Fuelband[38], Apple iWatch[39], and many similar products.[40] This development has profoundly impacted life as it enables ordinary people to collect their personal information much more easily than before. Secondly people have increasingly paid attention to life quality and life improvement, especially as it relates to health. This trend of demand has led to the rise of Quantified Self (QS) movement.

As a result, many mobile applications for personal data processing have come to the market, especially in the m-Health and m-Fitness domains. However many mainstream mobile applications have faced the same issue: the real-time personal data are not integrated with other data, especially the user's data, and the processing method is not dynamically adapted to the changes of the user's concurrent situation. Such mobile applications cannot be satisfactory as they don't incorporate users' most current data into their calculations. In order to overcome the shortcomings of the existing approaches, we combine the mobile computing, intelligent complex event processing and semantic technologies to develop a foundation of innovative mobile applications for much more efficient context-aware processing of real-time personal data streams by taking into account the resource limitation on mobile devices.

In order to achieve the research objective, we first determine the limitations of mobile devices and wireless sensors, analyze the issues of the existing solutions for real-time personal data processing, we then conduct a literature survey in the different research fields, including mobile and distribution CEP system, efficient mobile processing, semantic-based pattern and sensor model, and mobile sensing system and m-fitness/m-Health mobile apps (see Chapter 4) to analyze the existing tech-

[38] http://www.nike.com/us/en_us/c/nikeplus-fuelband
[39] http://www.smartwatchnews.org/apple-iwatch/
[40] http://www.smartwatchnews.org/activity-trackers-fitness-bands/

nologies and approaches. Based on these analyses and surveys we came up with a set of requirements for developing our approach (see Chapter 5).

According to the defined requirements we design an event-driven hybrid architecture for mobile applications, which is described in Chapter 6. The proposed software architecture combines mobile applications and remote server systems with the purpose of extending the computation capability of mobile devices. Real-time personal data are processed by the Complex Event processing (CEP) engines, which are executed on both mobile devices and backend servers. The communication between mobile applications and backend servers is achieved by the publish/subscribe middleware, which is able to provide various communication types. We also define a data model for the proposed system. Considering different use cases we also provide two different variant systems of our approach.

In order to achieve context awareness for real-time personal data processing, we develop the Monitoring Goal Network (MGN), which is described in Chapter 7, to model user's situations and the required patterns for processing the personal data in such situations. The MGN correlates the real-time personal data with user's context and domain knowledge, and defines the possible reactions of detected user's situations. Each monitoring goal models monitoring procedures for a specific purpose, assigns the patterns regarding the monitoring purpose, and defines the adaptation to the detected situations. The MGN was implemented in RDF/XML format and an executor of MGN has been implemented in our MCEP prototype.

In Chapter 7 we introduce the data collaboration mechanism that enables our approach to process user's personal data referencing the collaborative data from other users who have the similar context and are facing the same situation. Taking advantage of monitoring goal based monitoring realized through MGN, the approach provides a mechanism to find the most suitable and relevant users for data collaboration.

Considering the resource limitation of mobile devices, we design a resource-aware dynamic pattern distribution (see Chapter 8) to find the optimal pattern distribution between mobile device and backend server. The pattern distribution model we develop consists of four steps to achieve the dynamic pattern distribution. Employing a distribution fitness algorithm, for each pattern we calculate the fitness of all pattern bindings and deployment positions regarding the current workload of mobile device and backend server and find the optimal pattern binding and deployment position.

In order to manage patterns more efficiently and achieve dynamic adaptation to changes of availability of event resources, we develop a dynamic pattern management system (see Chapter 9) in our approach. Using semantic technologies we develop pattern and event resource ontology to store the patterns and event resources of users. The availabilities of event resources are dynamically updated during run-time. We define three basic adaptation methods using queries and reasoning functions of semantic technologies. Two algorithms are developed for real-time pattern adaptation. The first algorithm tries to achieve the dynamic adaptation of patterns by pattern deployment to solve the problem when an event resource for required event is not available. The second algorithm is used to deal with the situation that the availability of an event resource, which is used by a running pattern, changes in run-time.

In summary, the main achievements of this thesis are:

- We develop an event driven software architecture for CEP on mobile devices that provides a foundation for many similar applications to be developed.
- We devise a hybrid architecture by combining both mobile devices and backend servers such that the system based on this architecture can achieve maximum performance in detecting and processing real-time information and other relevant time-series and cross-sectional data.
- We employ the MGN for context-aware monitoring procedure modeling.
- The proposed system achieves to find the most suitable relevant user information for data collaboration.
- We employ the resource-aware dynamic pattern distribution for improving mobile application performance.
- We model patterns, event resources and sensors in an ontology.
- We develop pattern adaptation methods and algorithms.

11.2 Future research

This thesis describes a first step towards using CEP for real-time personal data processing on mobile devices. We developed a software prototype based on the described software architecture, methods and algorithms. However the system can be further improved in many areas:

Advanced hybrid architecture: In the current proposed hybrid architecture personal data are processed on both mobile device and backend server, while the situation analyzer, which executes MGN and realizes adaptation to real-time situations, is only run on the backend server. Therefore, the whole system depends heavily on backend servers. In the case of lost connection from backend server, mobile applications can only provide very limited functions. Although we also provide a mobile-only variant, the important features such as context awareness, data collaboration and dynamic pattern distribution are not supported in the mobile-only variant. In order to solve the issues mentioned above, the current hybrid architecture can be improved to advanced hybrid architecture, which implements all features on mobile devices. In such a system, the mobile device part would be able to download parts of MGN, patterns, context and domain knowledge from backend server and provide parts of functions of the backend server. In the case of disconnection from backend server, mobile applications can still provide full functions. However, the extended functions on the mobile device part in such a case can significantly increase the workload of mobile device, which causes the architecture to face the resource limitation issues of mobile devices as we very much address in the first place. Nevertheless, this situation should and needs to be considered for real-world applications.

Proactive data collaboration: The data collaboration is enabled and used in the proposed MCEP system for personal data processing. We provide only the basic data collaboration mechanism, while there are more potentials of data collaboration, such as proactive data collaboration. In the current approach the data collaboration is passive, which means the data collaboration is achieved only when the collaborative data are called or required. Such a mechanism can lead to a delay of personal data processing, due to the waiting time for the data collaboration. The proactive data collaboration processes the data from other users, who have the similar context and who are active during the run-time and produces collaborative data according to some special criteria. The required collaborative data can be immediately used without delay, since the data have already been produced in advance and stored in a backend server. The challenge of the proactive data collaborative is to decide what data should be used for collaboration and the data from which relevant users should be collaborated.

Pattern decomposition for pattern distribution: This is another important direction of further development of our approach. In our current pattern distribution model we focus only on the event resources allocation and deployment position selection, and regard each pattern as a unit. Pattern decomposition provides a new method for pattern distribution. It decomposes a pattern into several sub-patterns and deploys resources on different parts of the system. The advantages of pattern decomposition are: firstly the complex pattern, which needs intense computation capability, can be decomposed into simple patterns and be deployed flexibly; secondly the decomposition can reduce event transmission. As a pattern can use different events, which are from different sources, either from mobile device or backend server, in the current pattern execution model some events are required to be transferred to the position (mobile device or backend server), where the pattern is deployed. Pattern decomposition can decompose a pattern according to the event sources. Each sub-pattern uses all event resources that are from the same source (mobile device or backend server) and should be deployed in the corresponding position. The challenge of pattern decomposition is how to decompose patterns and how to merge the results of sub-patterns together without changing the function of the original pattern.

Graphical user interface for MGN and pattern model: In our approach we use semantic technologies to model monitoring goal network and pattern model for formal modeling and efficient querying. For different use cases, system administrators should model corresponding MGN and patterns. However, it is difficult to model MGN and patterns directly in the ontology. Hence graphical user interface (GUI) is required to simplify the modeling procedure. Duo to the limited research period, we are not able to provide such GUIs, but it should be done in the future development.

Appendix I

Monitoring Goal Network (MGN) ontology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [

  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >

  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >

  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

]>

<rdf:RDF xmlns="http://www.mcep.fzi.de/situations#"

   xml:base="http://www.mcep.fzi.de/situations"

   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

   xmlns:owl="http://www.w3.org/2002/07/owl#"

   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <owl:Ontology rdf:about="http://www.mcep.fzi.de/situations"/>

  <!--
///////////////////////////////////////////////////////////////////////////////////////

// Object Properties

///////////////////////////////////////////////////////////////////////////////////////

   -->

  <!-- http://www.mcep.fzi.de/situations#assignedValue -->

  <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#assignedValue">
```

```
      <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#EventValue"/>

      <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Value"/>

   </owl:ObjectProperty>



    <!-- http://www.mcep.fzi.de/situations#hasAction -->

   <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasAction">

      <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Action"/>

      <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Situation"/>

   </owl:ObjectProperty>



   <!-- http://www.mcep.fzi.de/situations#hasActionEvent -->

   <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasActionEvent">

      <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#ActionEvent"/>

      <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#CreateEventAction"/>

   </owl:ObjectProperty>



   <!-- http://www.mcep.fzi.de/situations#hasCondition -->

   <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasCondition">

      <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Condition"/>

      <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Situation"/>

   </owl:ObjectProperty>



 <!-- http://www.mcep.fzi.de/situations#hasEventValue -->

 <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasEventValue">
```

```
  <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#EventValue"/>

  <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#MCEPEvent"/>

</owl:ObjectProperty>



<!-- http://www.mcep.fzi.de/situations#hasObject -->

<owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasObject">

  <rdf:type rdf:resource="&owl;FunctionalProperty"/>

  <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Condition"/>

  <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Value"/>

</owl:ObjectProperty>



<!-- http://www.mcep.fzi.de/situations#hasOperator -->

<owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasOperator">

  <rdf:type rdf:resource="&owl;FunctionalProperty"/>

  <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Condition"/>

  <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

</owl:ObjectProperty>



<!-- http://www.mcep.fzi.de/situations#hasSituation -->

<owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasSituation">

  <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#MonitoringGoal"/>

  <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Situation"/>

</owl:ObjectProperty>
```

```
    <!-- http://www.mcep.fzi.de/situations#hasSubject -->

    <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasSubject">

        <rdf:type rdf:resource="&owl;FunctionalProperty"/>

        <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Condition"/>

        <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#Value"/>

    </owl:ObjectProperty>



    <!-- http://www.mcep.fzi.de/situations#hasTarget -->

    <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasTarget">

        <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#GoalChangeAction"/>

        <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#MonitoringGoal"/>

    </owl:ObjectProperty>



    <!-- http://www.mcep.fzi.de/situations#hasTriggerEvent -->

    <owl:ObjectProperty rdf:about="http://www.mcep.fzi.de/situations#hasTriggerEvent">

        <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#Situation"/>

        <rdfs:range rdf:resource="http://www.mcep.fzi.de/situations#TriggerEvent"/>

        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>

    </owl:ObjectProperty>

    <!--
///////////////////////////////////////////////////////////////////////////////////////////////

 // Data properties

///////////////////////////////////////////////////////////////////////////////////////////////
-->

    <!-- http://www.mcep.fzi.de/situations#fromEvent -->
```

```
<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#fromEvent">
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <rdf:Description
rdf:about="http://www.mcep.fzi.de/situations#SaveUserContextAction"/>
        <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#ValueFromEvent"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>


<!-- http://www.mcep.fzi.de/situations#fromKnowledgeBase -->
<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#fromKnowledgeBase">
  <rdfs:domain
rdf:resource="http://www.mcep.fzi.de/situations#ValueFromKnowledgeBase"/>
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>


<!-- http://www.mcep.fzi.de/situations#hasAttributeName -->
<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasAttributeName">
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:domain>
    <owl:Class>
```

```
            <owl:unionOf rdf:parseType="Collection">
               <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#EventValue"/>
               <rdf:Description
rdf:about="http://www.mcep.fzi.de/situations#SaveUserContextAction"/>
               <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#ValueFromEvent"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
   </owl:DatatypeProperty>


   <!-- http://www.mcep.fzi.de/situations#hasContextName -->
   <owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasContextName">
      <rdfs:range rdf:resource="&xsd;string"/>
      <rdfs:domain>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <rdf:Description
rdf:about="http://www.mcep.fzi.de/situations#SaveUserContextAction"/>
               <rdf:Description
rdf:about="http://www.mcep.fzi.de/situations#ValueFromUserContext"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
   </owl:DatatypeProperty>
```

```
<!-- http://www.mcep.fzi.de/situations#hasName -->

<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasName">

  <rdfs:range rdf:resource="&xsd;string"/>

  <rdfs:domain>

    <owl:Class>

      <owl:unionOf rdf:parseType="Collection">

        <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#Action"/>

        <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#MCEPEvent"/>

        <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#MonitoringGoal"/>

        <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#Situation"/>

      </owl:unionOf>

    </owl:Class>

  </rdfs:domain>

</owl:DatatypeProperty>


<!-- http://www.mcep.fzi.de/situations#hasPattern -->

<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasPattern">

  <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#MonitoringGoal"/>

  <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>


<!-- http://www.mcep.fzi.de/situations#hasQuery -->

<owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasQuery">
```

```
    <rdfs:domain
rdf:resource="http://www.mcep.fzi.de/situations#ValueFromKnowledgeBase"/>

    <rdfs:range rdf:resource="&xsd;string"/>

  </owl:DatatypeProperty>



  <!-- http://www.mcep.fzi.de/situations#hasType -->

  <owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasType">

    <rdfs:range rdf:resource="&xsd;string"/>

    <rdfs:domain>

      <owl:Class>

        <owl:unionOf rdf:parseType="Collection">

          <rdf:Description rdf:about="http://www.mcep.fzi.de/situations#FixedValue"/>

          <rdf:Description
rdf:about="http://www.mcep.fzi.de/situations#ValueFromKnowledgeBase"/>

        </owl:unionOf>

      </owl:Class>

    </rdfs:domain>

  </owl:DatatypeProperty>



  <!-- http://www.mcep.fzi.de/situations#hasValue -->

  <owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#hasValue">

    <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#FixedValue"/>

    <rdfs:range rdf:resource="&xsd;string"/>

  </owl:DatatypeProperty>
  <!-- http://www.mcep.fzi.de/situations#isStartGoal -->

  <owl:DatatypeProperty rdf:about="http://www.mcep.fzi.de/situations#isStartGoal">

    <rdfs:domain rdf:resource="http://www.mcep.fzi.de/situations#MonitoringGoal"/>

    <rdfs:range rdf:resource="&xsd;boolean"/>

  </owl:DatatypeProperty>

  <!--
///////////////////////////////////////////////////////////////////////////////////////
// Classes

///////////////////////////////////////////////////////////////////////////////////////

   -->

  <!-- http://www.mcep.fzi.de/situations#Action -->

  <owl:Class rdf:about="http://www.mcep.fzi.de/situations#Action"/>



  <!-- http://www.mcep.fzi.de/situations#ActionEvent -->

  <owl:Class rdf:about="http://www.mcep.fzi.de/situations#ActionEvent">

    <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#MCEPEvent"/>

  </owl:Class>



  <!-- http://www.mcep.fzi.de/situations#Condition -->

  <owl:Class rdf:about="http://www.mcep.fzi.de/situations#Condition"/>



  <!-- http://www.mcep.fzi.de/situations#ConditionOperator -->

  <owl:Class rdf:about="http://www.mcep.fzi.de/situations#ConditionOperator"/>



  <!-- http://www.mcep.fzi.de/situations#CreateEventAction -->
```

```
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#CreateEventAction">
    <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Action"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#EventValue -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#EventValue"/>


<!-- http://www.mcep.fzi.de/situations#FixedValue -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#FixedValue">
    <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Value"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#GoalChangeAction -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#GoalChangeAction">
    <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Action"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#MCEPEvent -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#MCEPEvent"/>


<!-- http://www.mcep.fzi.de/situations#MonitoringGoal -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#MonitoringGoal"/>


<!-- http://www.mcep.fzi.de/situations#SaveUserContextAction -->
```

```
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#SaveUserContextAction">
  <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Action"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#Situation -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#Situation"/>


<!-- http://www.mcep.fzi.de/situations#TriggerEvent -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#TriggerEvent">
  <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#MCEPEvent"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#Value -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#Value"/>


<!-- http://www.mcep.fzi.de/situations#ValueFromEvent -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#ValueFromEvent">
  <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Value"/>
</owl:Class>


<!-- http://www.mcep.fzi.de/situations#ValueFromKnowledgeBase -->
<owl:Class rdf:about="http://www.mcep.fzi.de/situations#ValueFromKnowledgeBase">
  <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Value"/>
</owl:Class>
```

```
  <!-- http://www.mcep.fzi.de/situations#ValueFromUserContext -->

  <owl:Class rdf:about="http://www.mcep.fzi.de/situations#ValueFromUserContext">

     <rdfs:subClassOf rdf:resource="http://www.mcep.fzi.de/situations#Value"/>

  </owl:Class>

  <!--
///////////////////////////////////////////////////////////////////////////////////////

  // Individuals

///////////////////////////////////////////////////////////////////////////////////////

   -->

  <!-- http://www.mcep.fzi.de/situations#exists -->

  <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#exists">

     <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

  </owl:NamedIndividual>



  <!-- http://www.mcep.fzi.de/situations#greaterThan -->

  <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#greaterThan">

     <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

  </owl:NamedIndividual>



  <!-- http://www.mcep.fzi.de/situations#notExists -->

  <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#notExists">

     <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

  </owl:NamedIndividual>
```

```
    <!-- http://www.mcep.fzi.de/situations#notSameAs -->

    <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#notSameAs">

        <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

    </owl:NamedIndividual>



    <!-- http://www.mcep.fzi.de/situations#sameAs -->

    <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#sameAs">

        <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

    </owl:NamedIndividual>



    <!-- http://www.mcep.fzi.de/situations#smallerThan -->

    <owl:NamedIndividual rdf:about="http://www.mcep.fzi.de/situations#smallerThan">

        <rdf:type rdf:resource="http://www.mcep.fzi.de/situations#ConditionOperator"/>

    </owl:NamedIndividual>

</rdf:RDF>
```

Appendix II

Pattern and event resources ontology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [

   <!ENTITY event "http://mcep.fzi.de/event/" >

   <!ENTITY stream "http://mcep.fzi.de/stream/" >

   <!ENTITY pattern "http://mcep.fzi.de/pattern/" >

   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >

   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >

   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

   <!ENTITY types "http://events.event-processing.org/types/" >

   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

]>

<rdf:RDF xmlns="http://mcep.fzi.de/"

    xml:base="http://mcep.fzi.de/"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

    xmlns:stream="http://mcep.fzi.de/stream/"

    xmlns:pattern="http://mcep.fzi.de/pattern/"

    xmlns:event="http://mcep.fzi.de/event/"

    xmlns:owl="http://www.w3.org/2002/07/owl#"

    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:types="http://events.event-processing.org/types/">
```

```
  <owl:Ontology rdf:about="http://mcep.fzi.de/">

  <owl:imports rdf:resource="http://purl.oclc.org/NET/ssnx/ssn"/>

  </owl:Ontology>

  <!--
///////////////////////////////////////////////////////////////////////////////////////
// Object Properties

////////////////////////////////////////////////////////////////////////////////////////

   -->

  <!-- http://mcep.fzi.de/connectTo -->

  <owl:ObjectProperty rdf:about="http://mcep.fzi.de/connectTo">

    <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

    <rdfs:range rdf:resource="http://mcep.fzi.de/EventSource"/>

  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/eventResource -->

  <owl:ObjectProperty rdf:about="http://mcep.fzi.de/eventResource">

    <rdfs:range rdf:resource="http://mcep.fzi.de/EventResource"/>

    <rdfs:domain rdf:resource="http://mcep.fzi.de/User"/>

  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/goal -->

  <owl:ObjectProperty rdf:about="http://mcep.fzi.de/goal">

    <rdfs:range rdf:resource="http://mcep.fzi.de/Goal"/>

    <rdfs:domain rdf:resource="&pattern;Pattern"/>

  </owl:ObjectProperty>
```

```
<!-- http://mcep.fzi.de/provideEvent -->

<owl:ObjectProperty rdf:about="http://mcep.fzi.de/provideEvent">

  <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

  <rdfs:range rdf:resource="&event;Event"/>

</owl:ObjectProperty>



<!-- http://mcep.fzi.de/event/attribute -->

<owl:ObjectProperty rdf:about="&event;attribute">

  <rdfs:range rdf:resource="&event;Attribute"/>

  <rdfs:domain rdf:resource="&event;Event"/>

</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/aggregationOperator -->

<owl:ObjectProperty rdf:about="&pattern;aggregationOperator">

  <rdfs:range rdf:resource="&pattern;AggregationOperator"/>

  <rdfs:domain rdf:resource="&pattern;AggregationValue"/>

</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/assignedAttribute -->

<owl:ObjectProperty rdf:about="&pattern;assignedAttribute">

  <rdfs:range rdf:resource="&event;Attribute"/>

  <rdfs:domain>

    <owl:Class>
```

```
            <owl:unionOf rdf:parseType="Collection">
               <rdf:Description rdf:about="&pattern;Condition"/>
               <rdf:Description rdf:about="&pattern;Value"/>
               <rdf:Description rdf:about="&pattern;Variable"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
   </owl:ObjectProperty>


   <!-- http://mcep.fzi.de/pattern/attributeSource -->
   <owl:ObjectProperty rdf:about="&pattern;attributeSource">
      <rdfs:range rdf:resource="&event;Attribute"/>
      <rdfs:domain rdf:resource="&pattern;AggregationValue"/>
   </owl:ObjectProperty>


   <!-- http://mcep.fzi.de/pattern/condition -->
   <owl:ObjectProperty rdf:about="&pattern;condition">
      <rdfs:range rdf:resource="&pattern;Condition"/>
      <rdfs:domain rdf:resource="&pattern;EventOperand"/>
   </owl:ObjectProperty>


   <!-- http://mcep.fzi.de/pattern/conditionOperator -->
   <owl:ObjectProperty rdf:about="&pattern;conditionOperator">
      <rdfs:domain rdf:resource="&pattern;Condition"/>
```

```
    <rdfs:range rdf:resource="&pattern;ConditionOperator"/>
</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/eventSource -->
<owl:ObjectProperty rdf:about="&pattern;eventSource">
    <rdfs:domain rdf:resource="&pattern;AggregationValue"/>
    <rdfs:range rdf:resource="&pattern;EventOperand"/>
</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/eventType -->
<owl:ObjectProperty rdf:about="&pattern;eventType">
    <rdfs:range rdf:resource="&event;Event"/>
    <rdfs:domain rdf:resource="&pattern;EventOperand"/>
</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/firstOperand -->
<owl:ObjectProperty rdf:about="&pattern;firstOperand">
    <rdfs:domain rdf:resource="&pattern;BinaryOperator"/>
    <rdfs:range rdf:resource="&pattern;Operand"/>
</owl:ObjectProperty>



<!-- http://mcep.fzi.de/pattern/operand -->
<owl:ObjectProperty rdf:about="&pattern;operand">
    <rdfs:range rdf:resource="&pattern;Operand"/>
    <rdfs:domain rdf:resource="&pattern;UnaryOperator"/>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/pattern/operator -->
  <owl:ObjectProperty rdf:about="&pattern;operator">
    <rdfs:range rdf:resource="&pattern;Operator"/>
    <rdfs:domain rdf:resource="&pattern;Pattern"/>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/pattern/output -->
  <owl:ObjectProperty rdf:about="&pattern;output">
    <rdfs:range rdf:resource="&pattern;EventOperand"/>
    <rdfs:domain rdf:resource="&pattern;Pattern"/>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/pattern/secondOperand -->
  <owl:ObjectProperty rdf:about="&pattern;secondOperand">
    <rdfs:domain rdf:resource="&pattern;BinaryOperator"/>
    <rdfs:range rdf:resource="&pattern;Operand"/>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/pattern/value -->
  <owl:ObjectProperty rdf:about="&pattern;value">
    <rdfs:domain rdf:resource="&pattern;Operand"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description rdf:about="&pattern;AggregationValue"/>
          <rdf:Description rdf:about="&pattern;Value"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/pattern/variable -->
  <owl:ObjectProperty rdf:about="&pattern;variable">
    <rdfs:domain rdf:resource="&pattern;EventOperand"/>
    <rdfs:range rdf:resource="&pattern;Variable"/>
  </owl:ObjectProperty>


  <!-- http://mcep.fzi.de/stream/stream -->
  <owl:ObjectProperty rdf:about="&stream;stream">
    <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>
    <rdfs:range rdf:resource="&stream;Stream"/>
  </owl:ObjectProperty>
  <!--
///////////////////////////////////////////////////////////////////////////////////////
 // Data properties
```

```
///////////////////////////////////////////////////////////////////////////////////////
 -->

   <!-- http://mcep.fzi.de/available -->

   <owl:DatatypeProperty rdf:about="http://mcep.fzi.de/available">

      <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

      <rdfs:range rdf:resource="&xsd;boolean"/>

   </owl:DatatypeProperty>


   <!-- http://mcep.fzi.de/cost -->

   <owl:DatatypeProperty rdf:about="http://mcep.fzi.de/cost">

      <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

      <rdfs:range rdf:resource="&xsd;integer"/>

   </owl:DatatypeProperty>


   <!-- http://mcep.fzi.de/eventFrequency -->

   <owl:DatatypeProperty rdf:about="http://mcep.fzi.de/eventFrequency">

      <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

      <rdfs:range rdf:resource="&xsd;int"/>

   </owl:DatatypeProperty>


   <!-- http://mcep.fzi.de/name -->

   <owl:DatatypeProperty rdf:about="http://mcep.fzi.de/name">

      <rdfs:range rdf:resource="&xsd;string"/>
```

```
</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/onMobileDevice -->

<owl:DatatypeProperty rdf:about="http://mcep.fzi.de/onMobileDevice">

  <rdfs:domain rdf:resource="http://mcep.fzi.de/EventResource"/>

  <rdfs:range rdf:resource="&xsd;boolean"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/privacyLevelsetting -->

<owl:DatatypeProperty rdf:about="http://mcep.fzi.de/privacyLevelsetting">

  <rdfs:domain rdf:resource="http://mcep.fzi.de/User"/>

  <rdfs:range rdf:resource="&xsd;integer"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/register -->

<owl:DatatypeProperty rdf:about="http://mcep.fzi.de/register">

  <rdfs:domain rdf:resource="http://mcep.fzi.de/User"/>

  <rdfs:range rdf:resource="&xsd;boolean"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/sensorName -->

<owl:DatatypeProperty rdf:about="http://mcep.fzi.de/sensorName">

  <rdfs:domain rdf:resource="http://mcep.fzi.de/SensingDevice"/>

  <rdfs:range rdf:resource="&xsd;string"/>
```

```
</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/userID -->

<owl:DatatypeProperty rdf:about="http://mcep.fzi.de/userID">

   <rdfs:domain rdf:resource="http://mcep.fzi.de/User"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/event/attributeType -->

<owl:DatatypeProperty rdf:about="&event;attributeType">

   <rdfs:domain rdf:resource="&event;Attribute"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/pattern/assignedValue -->

<owl:DatatypeProperty rdf:about="&pattern;assignedValue">

   <rdfs:domain rdf:resource="&pattern;Value"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/pattern/complexCondition -->

<owl:DatatypeProperty rdf:about="&pattern;complexCondition">

   <rdfs:domain rdf:resource="&pattern;Pattern"/>

   <rdfs:range rdf:resource="&xsd;string"/>
```

```
</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/pattern/conditionValue -->

<owl:DatatypeProperty rdf:about="&pattern;conditionValue">

   <rdfs:domain rdf:resource="&pattern;Condition"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/pattern/id -->

<owl:DatatypeProperty rdf:about="&pattern;id">

   <rdfs:domain rdf:resource="&pattern;Pattern"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/pattern/parameter -->

<owl:DatatypeProperty rdf:about="&pattern;parameter">

   <rdfs:domain rdf:resource="&pattern;ParOperator"/>

   <rdfs:range rdf:resource="&xsd;string"/>

</owl:DatatypeProperty>



<!-- http://mcep.fzi.de/stream/privacyLevel -->

<owl:DatatypeProperty rdf:about="&stream;privacyLevel">

   <rdfs:domain rdf:resource="&stream;Stream"/>

   <rdfs:range rdf:resource="&xsd;integer"/>
```

```
  </owl:DatatypeProperty>



  <!-- http://mcep.fzi.de/stream/sender -->

  <owl:DatatypeProperty rdf:about="&stream;sender">

    <rdfs:domain rdf:resource="&stream;Stream"/>

    <rdfs:range rdf:resource="&xsd;string"/>

  </owl:DatatypeProperty>



  <!-- http://mcep.fzi.de/stream/source -->

  <owl:DatatypeProperty rdf:about="&stream;source">

    <rdfs:domain rdf:resource="&stream;Stream"/>

    <rdfs:range rdf:resource="&xsd;string"/>

  </owl:DatatypeProperty>

  <!--
///////////////////////////////////////////////////////////////////////////////////////

  // Classes

///////////////////////////////////////////////////////////////////////////////////////
-->

  <!-- http://mcep.fzi.de/EventResource -->

  <owl:Class rdf:about="http://mcep.fzi.de/EventResource"/>



  <!-- http://mcep.fzi.de/EventSource -->

  <owl:Class rdf:about="http://mcep.fzi.de/EventSource"/>



  <!-- http://mcep.fzi.de/ExternalEventSource -->
```

```
<owl:Class rdf:about="http://mcep.fzi.de/ExternalEventSource">
    <rdfs:subClassOf rdf:resource="http://mcep.fzi.de/EventSource"/>
</owl:Class>


<!-- http://mcep.fzi.de/Goal -->
<owl:Class rdf:about="http://mcep.fzi.de/Goal"/>


<!-- http://mcep.fzi.de/SensingDevice -->
<owl:Class rdf:about="http://mcep.fzi.de/SensingDevice">
    <owl:equivalentClass rdf:resource="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice"/>
    <rdfs:subClassOf rdf:resource="http://mcep.fzi.de/EventSource"/>
</owl:Class>


<!-- http://mcep.fzi.de/User -->
<owl:Class rdf:about="http://mcep.fzi.de/User"/>


<!-- http://mcep.fzi.de/event/Attribute -->
<owl:Class rdf:about="&event;Attribute">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>


<!-- http://mcep.fzi.de/event/Event -->
<owl:Class rdf:about="&event;Event">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
</owl:Class>



<!-- http://mcep.fzi.de/pattern/AND -->

<owl:Class rdf:about="&pattern;AND">

  <rdfs:subClassOf rdf:resource="&pattern;BinaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/AggregationOperator -->

<owl:Class rdf:about="&pattern;AggregationOperator">

  <rdfs:subClassOf rdf:resource="&owl;Thing"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/AggregationValue -->

<owl:Class rdf:about="&pattern;AggregationValue">

  <rdfs:subClassOf rdf:resource="&owl;Thing"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/BinaryOperator -->

<owl:Class rdf:about="&pattern;BinaryOperator">

  <rdfs:subClassOf rdf:resource="&pattern;Operator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/Condition -->

<owl:Class rdf:about="&pattern;Condition">
```

```
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>


<!-- http://mcep.fzi.de/pattern/ConditionOperator -->
<owl:Class rdf:about="&pattern;ConditionOperator">
  <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>


<!-- http://mcep.fzi.de/pattern/EventOperand -->
<owl:Class rdf:about="&pattern;EventOperand">
  <rdfs:subClassOf rdf:resource="&pattern;Operand"/>
</owl:Class>


<!-- http://mcep.fzi.de/pattern/LengthWindow -->
<owl:Class rdf:about="&pattern;LengthWindow">
  <rdfs:subClassOf rdf:resource="&pattern;ParOperator"/>
  <rdfs:subClassOf rdf:resource="&pattern;UnaryOperator"/>
</owl:Class>


<!-- http://mcep.fzi.de/pattern/NOOP -->
<owl:Class rdf:about="&pattern;NOOP">
  <rdfs:subClassOf rdf:resource="&pattern;UnaryOperator"/>
</owl:Class>
```

```
<!-- http://mcep.fzi.de/pattern/NOT -->

<owl:Class rdf:about="&pattern;NOT">

   <rdfs:subClassOf rdf:resource="&pattern;UnaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/OR -->

<owl:Class rdf:about="&pattern;OR">

   <rdfs:subClassOf rdf:resource="&pattern;BinaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/Operand -->

<owl:Class rdf:about="&pattern;Operand">

   <rdfs:subClassOf rdf:resource="&owl;Thing"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/Operator -->

<owl:Class rdf:about="&pattern;Operator">

   <rdfs:subClassOf rdf:resource="&pattern;Operand"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/ParOperator -->

<owl:Class rdf:about="&pattern;ParOperator">

   <rdfs:subClassOf rdf:resource="&pattern;Operator"/>

</owl:Class>
```

```
<!-- http://mcep.fzi.de/pattern/Pattern -->

<owl:Class rdf:about="&pattern;Pattern">

   <rdfs:subClassOf rdf:resource="&owl;Thing"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/SEQ -->

<owl:Class rdf:about="&pattern;SEQ">

   <rdfs:subClassOf rdf:resource="&pattern;BinaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/TimeAt -->

<owl:Class rdf:about="&pattern;TimeAt">

   <rdfs:subClassOf rdf:resource="&pattern;ParOperator"/>

   <rdfs:subClassOf rdf:resource="&pattern;UnaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/TimeWindow -->

<owl:Class rdf:about="&pattern;TimeWindow">

   <rdfs:subClassOf rdf:resource="&pattern;ParOperator"/>

   <rdfs:subClassOf rdf:resource="&pattern;UnaryOperator"/>

</owl:Class>



<!-- http://mcep.fzi.de/pattern/UnaryOperator -->
```

```
  <owl:Class rdf:about="&pattern;UnaryOperator">
     <rdfs:subClassOf rdf:resource="&pattern;Operator"/>
  </owl:Class>


  <!-- http://mcep.fzi.de/pattern/Value -->
  <owl:Class rdf:about="&pattern;Value">
     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  </owl:Class>


  <!-- http://mcep.fzi.de/pattern/Variable -->
  <owl:Class rdf:about="&pattern;Variable">
     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  </owl:Class>


  <!-- http://mcep.fzi.de/stream/Stream -->
  <owl:Class rdf:about="&stream;Stream">
     <rdfs:subClassOf rdf:resource="&owl;Thing"/>
  </owl:Class>


  <!-- http://purl.oclc.org/NET/ssnx/ssn#SensingDevice -->
  <rdf:Description rdf:about="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice"/>
  <!--
///////////////////////////////////////////////////////////////////////////////////////
  // Individuals
```

```
///////////////////////////////////////////////////////////////////////////////////////

  -->

  <!-- http://mcep.fzi.de/BiggerEqual -->

  <owl:NamedIndividual rdf:about="http://mcep.fzi.de/BiggerEqual">

    <rdf:type rdf:resource="&pattern;ConditionOperator"/>

  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/SmallerEqual -->

  <owl:NamedIndividual rdf:about="http://mcep.fzi.de/SmallerEqual">

    <rdf:type rdf:resource="&pattern;ConditionOperator"/>

  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/AVG -->

  <owl:NamedIndividual rdf:about="&pattern;AVG">

    <rdf:type rdf:resource="&pattern;AggregationOperator"/>

  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/BiggerThan -->

  <owl:NamedIndividual rdf:about="&pattern;BiggerThan">

    <rdf:type rdf:resource="&pattern;ConditionOperator"/>

  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/COUNT -->

  <owl:NamedIndividual rdf:about="&pattern;COUNT">
```

```
    <rdf:type rdf:resource="&pattern;AggregationOperator"/>
  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/MAX -->
  <owl:NamedIndividual rdf:about="&pattern;MAX">
    <rdf:type rdf:resource="&pattern;AggregationOperator"/>
  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/MIN -->
  <owl:NamedIndividual rdf:about="&pattern;MIN">
    <rdf:type rdf:resource="&pattern;AggregationOperator"/>
  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/SUM -->
  <owl:NamedIndividual rdf:about="&pattern;SUM">
    <rdf:type rdf:resource="&pattern;AggregationOperator"/>
  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/SameAs -->
  <owl:NamedIndividual rdf:about="&pattern;SameAs">
    <rdf:type rdf:resource="&pattern;ConditionOperator"/>
  </owl:NamedIndividual>



  <!-- http://mcep.fzi.de/pattern/SmallerThan -->
  <owl:NamedIndividual rdf:about="&pattern;SmallerThan">

    <rdf:type rdf:resource="&pattern;ConditionOperator"/>

  </owl:NamedIndividual>

</rdf:RDF>
```

Appendix III

Evaluation Questionnaire

Usability

1. Is the mobile application easy to install?

☐Very easy ☐Easy ☐Moderate ☐Difficult ☐Very difficult

2. Is the mobile application easy to configure?

☐Very easy ☐Easy ☐Moderate ☐difficult ☐Very difficult

3. Is the mobile application easy to connect the sensors?

☐Very easy ☐Easy ☐Moderate ☐Difficult ☐Very difficult

4. Is it easy to obtain the feedback (alarm or recommendation) of monitoring from the mobile application?

☐Very easy ☐Easy ☐Moderate ☐Difficult ☐Very difficult

Reliability

5. Does the mobile application show alarm or recommendation at the right in time?

☐ Right ☐ Almost Right ☐Moderate ☐ Almost wrong ☐Wrong

6. Are the right alarms or recommendation shown by the mobile application?

☐ Right ☐ Almost Right ☐Moderate ☐ Almost wrong ☐Wrong

7. Are the alarms or recommendation shown by the mobile application useful to your situation?

☐Very useful ☐ Useful ☐Moderate ☐Not useful ☐Totally not useful

Robustness

8. Are the sensor connections stable (No disconnection during run time)?

☐Very stable ☐stable ☐don't know ☐sometime unstable ☐Very unstable

9. Did the mobile application close due to errors?

☐Never ☐ occasionally ☐ don't know ☐sometime ☐Very often

10. Did the mobile application stop working (not closed)?

☐Never ☐ occasionally ☐ don't know ☐sometime ☐Very often

Efficiency

11. Do you agree that the mobile application doesn't cause the quick battery consumption?

☐Totally agree ☐ Agree ☐ don't know ☐Not agree ☐ Totally Not agree

12. Do you agree that the mobile application doesn't affect the performance of other mobile apps?

☐Totally agree ☐ Agree ☐ don't know ☐Not agree ☐ Totally Not agree

13. Do you agree that the mobile application uses only a little internet traffic?

☐Totally agree ☐ Agree ☐ don't know ☐Not agree ☐ Totally Not agree

Bibliography

[AABC+05] Abadi, D,, Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Linder, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y. & Zdonik S. (2005) "The Design of the Borealis Stream Processing Engine," in Proceeding of the Second CIDR Conference, January 2005.

[ABBC+05] Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., & Widom, J. (2004) "STREAM: The Stanford Data Stream Management System," SpringerVerlag, New York.

[ACCC+03] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Erwin, C., & Zdonik, S. (2003, June). Aurora: a data stream management system. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of data (pp. 666-666). ACM.

[Addo13] Addo, A., & Seyram, Z. A. (2013). The adoption of mobile phone: How has it changed us socially?. Issues in Business Managementand Economics, 3, 47-60.

[AdEt02] Adi, A., & Etzion, O. (2002). The situation manager rule language. In RuleML (Vol. 60, pp. 36-57).

[AFRS+10] Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., & Studer, R. (2010). A rule-based language for complex event processing and reasoning. In Web Reasoning and Rule Systems (pp. 42-57). Springer Berlin Heidelberg.

[Agge11] Agger, B. (2011). iTime: Labor and life in a smartphone era. Time & Society,20(1), pp. 119-136.

[Agga13] Aggarwal, C. C. (2013). Managing and Mining Sensor Data. Springer.

[ANSI91] Standard Glossary of Software Engineering Terminology (ANSI). The Institute of Electrical and Electronics Engineers Inc. 1991.

[Appl10] iPad – Technical specifications and accessories for iPad, Apple Inc. January 27, 2010. Retrieved January 27, 2010.

[Bade10] Bade, D., (2010) Esper-Android: Event Stream Processing on Android. [Online]. Available: http://vsis-www.informatik.unihamburg.de/projects/esper-android/

[BaLG04] Badugu, R., Lakowicz, J. R., & Geddes, C. D. (2004). Excitation and emission wavelength ratiometric cyanide-sensitive probes for physiological sensing. Analytical biochemistry, 327(1), 82-90.

[BaMP09] Barnaghi, P., Meissner, S. & Presser, M. (2009) Sense and sensability: Semantic data modelling for sensor networks.. Proceedings of the ICT Mobile Summit 2009. pp. 1-9.

[Bbcr13] BBC Research. Global Markets and Technologies for Sensors 2013. BBC Research

[BeTG07] Behavior Tree Group, 2007, Behavior Tree Notation v1.0. ARC Centre for Complex Systems, www.behaviorengineering.org/docs/Behavior-Tree-Notation-1.0.pdf

[BeVa07] Bernhardt, T., & AlexandreV., (2007). Esper: Event stream processing and correlation, ONJava, in http://www. onjava. com/lpt/a/6955, O'Reilly.

[BiER08] Biger, A., Etzion, O., & Rabinovich, Y. (2008). Stratified Implementation of Event Processing Network. Fast abstract on DEBS.

[BrDu10] Bruns, R., and Dunkel, J. (2010). Event-Driven Architecture. Springer Berlin Heidelberg.

[BSIG07] Bluetooth, S. I. G. (2007). Bluetooth specification

[CBBG+12] Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., & Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. Web Semantics: Science, Services and Agents on the World Wide Web, 17, 25-32.

[CBCW+10] Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010, June). MAUI: making smartphones last longer with code offload. In Proceedings of the 8th International Conference on Mobile systems, applications, and services (pp. 49-62). ACM.

[CCDF+03] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., …& Shah, M. A. (2003, June). TelegraphCQ: continuous dataflow processing. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (pp. 668-668). ACM.

[CDML+75] Cooper, M., Dronsuth, R. W., Mikulski, A. J., Lynk Jr, C. N., Mikulski, J. J., Mitchell, J. F., …& Sangster, J. H. (1975). U.S. Patent No. 3,906,166. Washington, DC: U.S. Patent and Trademark Office.

[ChCC07] Chandy, K. M., Charpentier, M., & Capponi, A. (2007, June). Towards a theory of events. In Proceedings of the 2007 inaugural International Conference on Distributed Event-based Systems (pp. 180-187). ACM.

[ChEA11] Chandy, M. K., Etzion, O., & von Ammon, R. (2011). 10201 executive summary and manifesto–event processing. Event Processing, (10201).

[ChMa02] Chaubey, A., & Malhotra, B. D. (2002). Mediated biosensors. Biosensors and Bioelectronics, 17(6), 441-456.

[ChSc09] Chandy, K., & Schulte, W. (2009). Event Processing: Designing IT Systems for Agile Companies. McGraw-Hill, Inc..

[CIMN+11] Chun, B. G., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011, April). Clonecloud: Elastic execution between mobile device and cloud. In Proceedings of the sixth Conference on Computer systems (pp. 301-314). ACM.

[CMTC+08] Consolvo, S., McDonald, D. W., Toscos, T., Chen, M. Y., Froehlich, J., Harrison, B., & Landay, J. A. (2008, April). Activity sensing in the wild: a field trial of ubifit garden. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1797-1806). ACM.

[Cox11] Cox, P. A. (2011). Mobile cloud computing. IBM developer Works, 1-10.

[Culb94] Culbert, M. (1994, March). Low power hardware for a high performance PDA. In COMPCON (Vol. 94, pp. 144-147).

[DACH+12] Denning, T., Andrew, A., Chaudhri, R., Hartung, C., Lester, J., Borriello, G., & Duncan, G. (2009, February). Balance: towards a usable pervasive wellness application with accurate activity inference. In Proceedings of the 10th workshop on Mobile Computing Systems and Applications (p. 5). ACM.

[DGHR+05] Demers, A., Gehrke, J., Hong, M., Riedewald, M., & White, W. (2005). A general algebra and implementation for monitoring event streams. Cornell University.

[DGPR+07] Demers, A. J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., & White, W. M. (2007, January). Cayuga: A General Purpose Event Monitoring System. In CIDR (Vol. 7, pp. 412-422). [DiGG95] Dittrich, K. R., Gatziu, S., & Geppert, A. (1995). The active database management system manifesto: A rulebase of ADBMS features. In Rules in Database Systems (pp. 1-17). Springer Berlin Heidelberg.

[Drom06] Dromey, R. G. (2006). Formalizing the transition from requirements to design.Mathematical Frameworks for Component Software-Models for Analysis and Synthesis, 156-187.

[DuBS13] Dunkel, J., Bruns, R., & Stipkovic, S. (2013, March). Event-based smartphone sensor processing for ambient assisted living. In Autonomous Decentralized Systems (ISADS). IEEE.

[ElYM00] El-Sherif, M. A., Yuan, J., & Macdiarmid, A. (2000). Fiber optic sensors and smart fabrics. Journal of Intelligent Material Systems and Structures, 11(5), 407-414.

[EOLL+08] Miluzzo, E., Oakley, J. M., Lu, H., Lane, N. D., Peterson, R. A., & Campbell, A. T. (2008). Evaluating the iPhone as a mobile platform for people-centric sensing applications. In Proceedings of UrbanSense'08.

[Espe07] Esper Tech, Esper Performance, http://docs.codehaus.org/display/ESPER/Esper +performance, EsperTech 2007

[EtNi10] Etzion, O., & Niblett, P. (2010). Event processing in action. Manning Publications Co..

[FaMH13] Fahim, A., Mtibaa, A., & Harras, K. A. (2013, September). Making the case for computational offloading in mobile device clouds. In Proceedings of the 19th annual International Conference on Mobile computing & networking (pp. 203-205). ACM.

[Farl05] Farley, T. (2005). Mobile telephone history. Telektronikk, 101(3/4), 22.

[FGNW12] Fesehaye, D., Gao, Y., Nahrstedt, K., & Wang, G. (2012, September). Impact of cloudlets on interactive mobile cloud applications. In Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International (pp. 123-132). IEEE.

[Fox99] Fox, K. R. (1999). The influence of physical activity on mental well-being. Public health nutrition, 2(3a), 411-418.

[FoZa94] Forman, G. H., & Zahorjan, J. (1994). The challenges of mobile computing. Computer, 27(4), 38-47.

[Fran13] Frank, R. (2013). Understanding smart sensors. Artech House.

[Gart13] Gartner (2013). Forecast: Mobile App Stores, Worldwide, 2013 Update, Gartner 2013

[GBHF89] George, L. K., Blazer, D. G., Hughes, D. C., & Fowler, N. (1989). Social support and the outcome of major depression. The British Journal of Psychiatry, 154(4), 478-485.

[Gibs12] Gibson, J. D. (Ed.). (2012). Mobile communications handbook. CRC press.

[GiSt87] Gibson, Stephen W. (1987). Cellular Mobile Radiotelephones. Englewood Cliffs, Prentice Hall, 19–22.

[Goza08] Gozalvez, J. (2008). First Google's android phone launched [Mobile Radio].Vehicular Technology Magazine, IEEE, 3(4), 3-69.

[GPJD+08] Gomez, D., Preece, A., Johnson, M., De Mel, G., Vasconcelos, W., Gibson, C., Barnoy, A., Borowiecki, K., Porta, T. & Pizzocaro, D. (2008) An ontology-centric approach to sensor-mission assignment. 16th International Conference on Knowledge Engineering and Knowledge Management.

[Gros 04] Grossman, L. (2007). Invention of the year: The iPhone. Time Magazine Online, 1.

[HaGu13a] Hatler, M., Gurganious, D., & Chi, C., (2013). Mobile Sensing Sports & Fitness: A Market Dynamics Report. ON World's Research.

[HaGu13b] Hatler, M., Gurganious, D., & Chi, C., (2013). Mobile Sensing Health & Wellness: A Market Dynamics Report. ON World's Research.

[HBPW08] Honicky, R., Brewer, E. A., Paulos, E., & White, R. (2008, August). N-smarts: networked suite of mobile atmospheric real-time sensors. InProceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions (pp. 25-30). ACM.

[HKMS94] Herbst, H., Knolmayer, G., Myrach, T., & Schlesinger, M. (1994, May). The specification of business rules: A comparison of selected methodologies. InMethods and associated tools for the information systems life cycle (pp. 29-46).

[HoLM05] Hoang, H., Lee-Urban, S., & Muñoz-Avila, H. (2005). Hierarchical Plan Representations for Encoding Strategic Game AI. In AIIDE (pp. 63-68).

[IDC10] IDC Worldwide Quarterly PC Tracker, 2010

[IDC 13a] IDC Worldwide Quarterly Mobile Phone Tracker, 2013

[IDC 13b] IDC Worldwide Quarterly Smart Connected Device Tracker, 2013

[InDA99] Infrared Data Association. (1999). IrDA Object Exchange Protocol (IrOBEX) with Published Errata.

[IrDA08] Infrared Data Association. (2008). IrDA Specifications and Technical Notes. http://www. irda. org/displaycommon. cfm

[KaRR98] Kappel, G., Rausch-Schott, S., & Retschitzegger, W. (1998). Coordination in workflow management systems—a rule-based approach. In Coordination Technology for Collaborative Applications (pp. 99-119). Springer Berlin Heidelberg.

[KCCD+03] Krishnamurthy, S., Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., & Shah, M. A. (2003). TelegraphCQ: An architectural status report. IEEE Data Eng. Bull., 26(1), 11-18.

[KhSt09] El Kharbili, M., & Stojanovic, N. (2009). Semantic Event-Based Decision Management in Compliance Management for Business Processes. In AAAI Spring Symposium: Intelligent Event Processing (pp. 35-40).

[Kjel13] Kjeldskov, J. (2013). Mobile computing. The Encyclopedia of Human-Computer Interaction, 2nd Ed.

[KlCa06] Klyne, G., & Carroll, J. J. (2006). Resource description framework (RDF): Concepts and abstract syntax.

[KLCC+09] Lee, J. H., Cheol, R., Jo, J. C., & You, Y. D. (2009). Embedded CEP engine used in DDS-based mobile devices for differentiated services for customers. In Consumer Electronics, 2009. ISCE'09.

[KnEP00] Knolmayer, G., Endl, R., & Pfahrer, M. (2000). Modeling processes and workflows by business rules. In Business Process Management (pp. 16-29). Springer Berlin Heidelberg.

[Kobl11] Koblentz, Evan. "How it Started: Mobile Internet Devices of." Human-computer Interaction and Innovation in Handheld, Mobile, and Wearable Technologies (2011): 172.

[Kros08] Kroski, E. (2008). On the move with the mobile web: libraries and mobile technologies. Library technology reports, 44(5), 1-48.

[KuLu10] Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. Computer, 43(4), 51-56.

[LeMa02] Lee, S. W., & Mase, K. (2002). Activity and location recognition using wearable sensors. IEEE pervasive computing, 1(3), 24-32.

[LMLY+12] Lane, N. D., Mohammod, M., Lin, M., Yang, X., Lu, H., Ali, S., & Campbell, A. (2011, May). Bewell: A smartphone application to monitor, model and promote wellbeing. In 5th International ICST Conference on Pervasive Computing Technologies for Healthcare (pp. 23-26).

[Lock10] Locke, D. (2010). MQ Telemetry Transport (MQTT) V3. 1 Protocol Specification. IBM developerWorks Technical Library], available at http://www. ibm. com/developerworks/webservices/library/ws-mqtt/index. html.

[Lowe89] Lowe, C. R. (1989). Biosensors. Philosophical Transactions of the Royal Society of London. B, Biological Sciences, 324(1224), 487-496.

[Luck01] Luckham, D. (2002). The power of events. Reading: Addison-Wesley.

[LSAB+11] Luckham, D., Schulte, R., Adkins, J., Bizarro, P., Jacobsen, H. A., Mavashev, A., & Niblett, P. (2011). Event processing glossary-version 2.0.Event Processing Technical Society.

[Lyu96] Lyu, M. R. (1996). Handbook of software reliability engineering (Vol. 222). CA: IEEE computer society press.

[Matt06] Mattila, H. (Ed.). (2006). Intelligent textiles and clothing. Woodhead Publishing.

[MEJM06] Mohomed, I., Ebling, M. R., Jerome, W., & Misra, A. (2006, September). HARMONI: Motivation for a health-oriented adaptive remote monitoring middleware. In fourth international workshop on ubiquitous computing for pervasive healthcare applications (UbiHealth 2006), Irvine, California, USA, September.

[MGBM+02] Messer, A., Greenberg, I., Bernadat, P., Milojicic, D., Chen, D., Giuli, T. J., & Gu, X. (2002). Towards a distributed platform for resource-constrained devices. In Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on (pp. 43-51). IEEE

[MiLi11] Misra, A., & Lim, L. (2011, June). Optimizing sensor data acquisition for energy-efficient smartphone-based continuous event processing. In Mobile Data Management (MDM), 2011 12th IEEE International Conference on (Vol. 1, pp. 88-97). IEEE.

[Mitc97] Mitchell, S. W. (1997, July). A hybrid architecture for real-time mixed-initiative planning and control. In AAAI/IAAI (pp. 1032-1037).

[MPES09] Mouttham, A., Peyton, L., Eze, B., & Saddik, A. E. (2009). Event-driven data integration for personal health monitoring. Journal of Emerging Technologies in Web Intelligence, 1(2), 110-118.

[MPPB+09] Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., & Smith, M. (2009). OWL 2 web ontology language: Structural specification and functional-style syntax. W3C recommendation, 27, 17.

[MZZW+13] Ma, X., Zhao, Y., Zhang, L., Wang, H., & Peng, L. (2013). When Mobile Terminals Meet the Cloud: Computation Offloading as the Bridge. IEEE NETWORK, 27(5), 28-33.

[Niel03] Nielsen, J. (2003). Usability 101: Introduction to usability.

[NoCC92] Norris, R., Carroll, D., & Cochrane, R. (1992). The effects of physical activity and exercise training on psychological stress and well-being in an adolescent population. Journal of psychosomatic research, 36(1), 55-65.

[Ober11] Oberoi, S., (2011). Sensor fusion brings situational awareness to health device. Book chapter of Embedded system design, volume 24, number 5, June 2011

[PGDV+08] Preece, A., Gomez, M., De Mel, G., Vasconcelos, W., Sleeman, D., Colley, S., Pearson, G., Pham, T. & Porta, T. (2008) Matching sensors to missions using a knowledge-based approach. SPIE Defense Transformation and Net-Centric Systems.

[PHRM+12] Paraiso, F., Hermosillo, G., Rouvoy, R., Merle, P., & Seinturier, L. (2012, September). A middleware platform to federate complex event processing. In Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International (pp. 113-122). IEEE.

[Pogu98] Pogue, D., (1998). PalmPilot: the ultimate guide. O'Reilly & Associates, Inc.

[PPVA+13] Patiniotakis, I., Papageorgiou, N., Verginadis, Y., Apostolou, D., & Mentzas, G. (2013). Dynamic event subscriptions in distributed event based architectures.Expert Systems with Applications, 40(6), 1935-1946.

[QuAR11] Qureshi, S. S., Ahmad, T., & Rafique, K. (2011, September). Mobile cloud computing as future for mobile applications-Implementation methods and challenging issues. In Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on (pp. 467-471). IEEE.

[Rahu13] Rahul, G., (2013). Exploring the Floating Point Performance of Modern ARM Processors, http://www.anandtech.com/show/6971/exploring-the-floating-point-performance-of-modern-arm-processors

[RDGN04] Rasmusson, J., Dahlgren, F., Gustafsson, H., & Nilsson, T. (2004). Multimedia in mobile phones—The ongoing revolution. Ericsson review, 2, 98-107.

[Robi09] Robinson, S. (2009). Cellphone energy gap: Desperately seeking solutions.Strategy Analytics.

[RoBo06] Robin, A., & Botts, M. E. (2006). Creation of Specific SensorML Process Models. Earth System Science Center-NSSTC, University of Alabama in Huntsville (UAH), HUNTSVILLE, AL, 35899.

[Rozs08] Rozsnyai, S. (2008) Managing Event Streams for Querying Complex Events. Vienna University of Technology, Vienna, Austria.

[Saty10] Satyanarayanan, M. (2011). Mobile computing: the next decade. ACM SIGMOBILE Mobile Computing and Communications Review, 15(2), 2-10.

[SBCD09] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for vm-based cloudlets in mobile computing. Pervasive Computing, IEEE, 8(4), 14-23.

[SeSt10] Sen, S., & Stojanovic, N. (2010, January). GRUVe: a methodology for complex event pattern life cycle management. In Advanced Information Systems Engineering (pp. 209-223). Springer Berlin Heidelberg.

[SKDB+09] Stevenson, G., Knox, S., Dobson, S. & Nixon, P. (2009) Ontonym: a collection of upper ontologies for developing pervasive systems. CIAO '09: Proceedings of the 1st Workshop on Context, Information and Ontologies. pp. 1-8.

[SKPR12] Schilling, B., Koldehofe, B., Pletat, U., & Rothermel, K. (2012). Distributed Heterogeneous Event Processing. In Proceedings of the ACM International Conference on Distributed Event-Based System 2012.

[SLAZ12] Shi, C., Lakafosis, V., Ammar, M. H., & Zegura, E. W. (2012, June). Serendipity: enabling remote computing among intermittently connected mobile devices. In Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing (pp. 145-154). ACM.

[SMXS+11] Stojanovic, N., Milenovic, D., Xu, Y., Stojanovic, L., Anicic, D., & Studer, R. (2011, July). An intelligent event-driven approach for efficient energy consumption in commercial buildings: smart office use case. In Proceedings of the 5th ACM International Conference on Distributed Event-Based System(pp. 303-312). ACM.

[SSOG12] Stühmer, R., Stojanovic, N., Obermeier, S., & Gibert, P. (2012, July). Where events meet events: PLAY event marketplace. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (pp. 383-384). ACM.

[StBD13] Stipkovic, S., Bruns, R., & Dunkel, J. (2013, September). Pervasive Computing by Mobile Complex Event Processing. In e-Business Engineering (ICEBE).

[StRX13] Stojanovic, N., Riemer, D., Xu, Y., (2013): Demo: a system for dynamic real-time personal fitness monitoring. In Proceedings of the ACM International Conference on Distributed Event-Based System 2013: (pp. 341-342). ACM.

[StSX14] Stojanovic, N., Stojanovic, L & Xu, Y. (2014) Tutorial: Mobile CEP in Real-time Big Data Processing: Challenges and Opportunities. In the 8th ACM International Conference on Distributed event-based system. ACM.

[SXSS14a] Stojanovic, N., Xu, Y., Stojadinovic, A., & Stojanovic, L (2014) Using Mobile-based Complex Event Processing to realize Collaborative Remote Person Monitoring . In Proceedings of the 8th ACM International Conference on Distributed Event-Based System. ACM.

[SXSS14b] Stojanovic, N., Xu, Y., Stajic, B., & Stojanovic, L (2014) Demo: Mobile CEP Architecture: from Intelligent Sensing to Collaborative Monitoring. In Proceedings of the 8th ACM International Conference on Distributed Event-Based System. ACM.

[TaSt02] Tanenbaum, A. S., & Van Steen, M. (2002). Distributed systems.

[TKWW87] Turner, A. P. F., Karube, I., Wilson, G. S., & Worsfold, P. J. (1987). Biosensors: fundamentals and applications.

[Vikr07] Vikram, K. (2007). FingerLakes: A Distributed Event Stream Monitoring System.

[VSDD12] Verbelen, T., Simoens, P., De Turck, F., & Dhoedt, B. (2012, June). Cloudlets: bringing the cloud to the mobile user. In Proceedings of the third ACM workshop on Mobile cloud computing and services (pp. 29-36). ACM.

[Weis91] Weiser, M. (1991). The computer for the 21st century. Scientific american,265(3), pp. 94-104.

[Weis03] Weiss, S., (2003). Handheld usability, John Wiley & Sons.

[Welc13] Welch, C. (2013). Google: Android app downloads have crossed 50 billion, over 1M apps in Play. The Verge, July 24, 2013.

[WiGr10] Wilhelm, F. H., & Grossman, P. (2010). Emotions beyond the laboratory: Theoretical fundaments, study design, and analytic strategies for advanced ambulatory assessment. Biological psychology, 84(3), 552-569.

[Wils04] Wilson, J. S. (2004). Sensor technology handbook. Elsevier.

[WoPM99] Wong, D., Paciorek, N., & Moore, D. (1999). Java-based mobile agents.Communications of the ACM, 42(3), 92-ff.

[WSSM+08] Witt, K. J., Stanley, J., Smithbauer, D., Mandl, D., Ly, V., Underbrink, A., & Metheny, M. (2008). Enabling sensor webs by utilizing SWAMO for autonomous operations. In NASA Earth Science Technology Conference (ESTC2008).

[XSMA11] Xu, Y., Stojanovic, L., Ma, J., Anicic, D. (2011, July). Demo: Efficient energy consumption in a smart office based on intelligent complex event processing. In Proceedings of the ACM International Conference on Distributed Event-Based System. (pp. 379-380). ACM.

[XSSA+11] Xu, Y., Stojanovic, N., Stojanovic, L., Anicic, D., & Studer, R. (2011). An approach for more efficient energy consumption based on real-time situational awareness. In Proceedings of ESWC 2011. In The Semantic Web: Research and Applications (pp. 270-284). Springer Berlin Heidelberg.

[XSSC+12] Xu, Y., Stojanovic, N., Stojanovic, L., Cabrera, A., & Schuchert, T. (2012): An approach for using complex event processing for adaptive augmented reality in cultural heritage domain: experience report. In Proceedings of the ACM International Conference on Distributed Event-Based System 2012: (pp. 139-148)

[XSSK13]	Xu, Y., Stojanovic, N., Stojanovic, L., & Kostic, D. (2013, December). An Approach for Dynamic Personal Monitoring based on Mobile Complex Event Processing. In Proceedings of International Conference on Advances in Mobile Computing & Multimedia (p. 464). ACM.
[XSSS12a]	Xu, Y., Stojanovic, N., Stojanovic, L., & Schuchert, T. (2012). Demo: Efficient Human Attention Detection in Museums based on Semantics and Complex Event Processing. In International Semantic Web Conference (ISWC) 2012.
[XSSS12b]	Xu, Y., Stojanovic, N., Stojanovic, L., & Schuchert, T. (2012). A Demo for efficient human Attention Detection based on Semantics and Complex Event Processing, In Extended Semantic Web Conference (ESWC) 2012 (Posters & Demos).
[XSSS12c]	Xu, Y., Stojanovic, N., Stojanovic, L., & Schuchert, T. (2012). : Efficient human attention detection based on intelligent complex event processing. In Proceedings of the ACM International Conference on Distributed Event-Based System 2012: (pp. 379-380)
[XWSH10]	Xu, Y., Wolf, P., Stojanovic, N., & Happel, H. J. (2010). Semantic-based Complex Event Processing in the AAL Domain. In ISWC Posters&Demos.
[Zeph10]	Zephyr technology, (2010). HxM BT Datasheet, Zephyr technology.
[Zeph12]	Zephyr technology, (2012). BioHarness 3.0 User Manual, Zephyr technology.
[ZiAl06]	ZigBee Alliance (2006). ZigBee specification.
[Zimm96]	Zimmerman, T. G. (1996). Personal area networks: near-field intrabody communication. IBM systems Journal, 35(3.4), 609-617