Olivier Coutand

A Framework for Contextual Personalised Applications

kassel
university
press

Die vorliegende Arbeit wurde vom Fachbereich Elektrotechnik / Informatik der Universität Kassel als Dissertation zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften (Dr.-Ing.) angenommen.

Erster Gutachter: Prof. Dr. Klaus David (Universität Kassel)
Zweiter Gutachter: Prof. Dr. Klaus Schmid (Universität Hildesheim)

Tag der mündlichen Prüfung 13. November 2008

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar

Zugl.: Kassel, Univ., Diss. 2008
ISBN print: 978-3-89958-774-6
ISBN online: 978-3-89958-775-3
URN: http://nbn-resolving.de/urn:nbn:de:0002-7742

Printed in Germany

Abstract

Context awareness allows for the development of computing systems that automatically adapt to the user's situation. In the course of the research on context-awareness, a large variety of prototypical context-aware applications has been developed. Based on the use of context and the adaptation methods utilised to select the items presented to the users, one can categorise these applications into two classes we refer to as: *Contextual Personalisation* and *Contextual Selection*. The former class characterises applications that provide functionalities adapted to and matching their users' personal needs. Here, context enables the selection of certain the pieces of information that will govern the adaptation of applications. The latter class encompasses systems providing a set of operations that does not utilise any user preference, but rather provides a functionality based on a sole piece of context information. As such, they are not tied to any user specific preference. The focus of this work centers on contextual personalisation.

Current realisations have limited capabilities when it comes to unknown contexts, i.e. contexts in which the user preference has not been explicitly expressed. These systems mostly use rules that associate a context to an action to be performed by the service. However, in contexts where the rules (no longer) apply, no action can be carried out until the current rules are modified.

This work introduces Copernik, a framework for personalising contextual personalised applications. It provides a set of design principles for governing the development of new solutions to perform contextual personalisation.

Firstly, Copernik includes *a contextual personalisation cycle* that encompasses the required set of operations involved in the personalisation of an application. The personalisation process is supported by Case-Based Reasoning, which relies on a user's previous experiences with the application. Personalisation is enabled by inferring these user's preferences even in unforeseen contexts and without requesting user interventions. In addition, the cycle mandates that every time an application becomes personalised, an explanation is provided to the user. This allows the user to assess the validity of the action and to provide feedback, thereby permitting the user to be in control of the application. Furthermore, the framework makes the case for a central access point to enable user privacy. In order to address the users' concerns on the safety of their personal information (context and preferences) a functional separation is introduced between the applications providing adapted behaviours to users and an underlying system which is in charge of determining how the applications should to react. This frees applications from having to maintain and process all user information.

Secondly, a *retrieval function* is developed that fetches the most similar previous contexts. The function assesses a degree of similarity between the new context and those of the user's previous experiences. The development of the retrieval function is guided by three design principles: 1) similarity between contexts can be assessed differently for different users, 2) When assessing similarity between different pairs of contexts, the relevant features may differ, 3) Revealing aspects of context must be displayed to explain to the users the reasons for the previous experiences' selection.

Thirdly, the framework stipulates that the adaptation process be performed separately from the applications themselves. Because applications have their own characteristics that influence adaptation, a system should be provided with the application characteristic information, referred to as *application-specific knowledge*. Templates are developed to express this application-specific knowledge and communicate it to the system performing the adaptation.

Finally, to show the feasibility of the framework and evaluate the approach, a prototypical application is implemented: the Call Profile Manager. For this application, specific knowledge is investigated and defined.

Zusammenfassung

Kontextsensitivität hat zum Ziel, Softwaresysteme zu entwickeln, die sich automatisch der Benutzersituation anpassen. Im Rahmen der Forschung auf dem Gebiet der Kontextsensitivität wurden vielfältige prototypische Anwendungen entwickelt.

Anwendungen werden in dieser Arbeit anhand der Kontextverwendung und des Einsatzes der Adaptionsmethoden in zwei Klassen aufgeteilt: 1) Contextual Personalisation (auf Deutsch kontextuelle Personalisierung); 2) Contextual Selection (kontextuelle Selektierung). Contextual Personalisation beschreibt Anwendungen, die Funktionalitäten bereitstellt, welche den persönlichen Benutzerbedürfnissen angepasst sind. Der Kontext dient hier zur Auswahl der Benutzerpräferenzen, welche die Anwendung steuert. Die zweite Klasse, Contextual Selection, beschreibt Systeme, die basierend auf den Kontextinformationen ein Set an Operationen und Funktionalitäten bereitstellen. Diese Operationen und Funktionalitäten sind unabhängig von Benutzerpräferenzen. Der Schwerpunkt dieser Arbeit liegt auf Contextual Personalisation.

Existierende Systeme bilden über explizit aufgestellte Regeln Kontextinformationen auf Aktionen beziehungsweise Benutzerpräferenzen ab. Eine gravierende Einschränkung der bisher entwickelten Konzepte zur Contextual Personalisation liegt darin, dass der Benutzer Regeln für alle Kontexte aufstellen muss. Unbekannte Kontexte werden somit von keiner Regel erfasst und können daher keine Aktionen auslösen.

Die vorliegende Arbeit beschäftigt sich mit drei Schwerpunkten. Im Schwerpunkt eins wird Copernik – A Framework for personalising Contextual Personalised Applications – vorgestellt. Copernik analysiert die Anforderungen und stellt Lösungsstrategien zur Entwicklung neuer Contextual Personalisation Systeme zur Verfügung. Der zweite Schwerpunkt behandelt ein Verfahren zur Bestimmung von Kontextähnlichkeit, welches seinen Einsatz in Copernik findet. Der letzte Schwerpunkt setzt sich mit dem Thema des applikationsspezifischen Wissens im Rahmen der Adaption von Anwendungen auseinander. Desweiteren liefert die Arbeit Implementierungen eines Contextual Personalisation Systems und einer exemplarischen Applikation auf Basis des Frameworks.

Copernik schlägt die Separation von Applikation und Contextual Personalisation vor. Außerdem beschreibt es ein zyklisches Ablaufmodell eines vollständigen Contextual Personalised Systems. Dieses Ablaufmodell besteht aus einem Set von Operationen, die zur Personalisierung einer Anwendung erforderlich sind.
Um die privaten Benutzerinformationen (Kontexten und Präferenzen) vor Fremdzugriff zu schützen wird eine Trennung zwischen der Anwendung und dem System zur Auswertung von Kontextinformationen und Nutzerpräferenzen definiert. Dies verhindert, dass die Applikation allgemeinen Zugriff auf die Nutzerdaten hat.

Desweiteren wurde eine “Retrievial Funktion” entwickelt. Diese errechnet einen „Gemeinsamkeitsgrad“ zwischen dem aktuellen Kontext und Kontexten in denen bereits

Applikationsaktionen stattfanden.

Diese Funktion basiert darauf, dass: 1) Gemeinsamkeiten von Kontexten für unterschiedliche Nutzer unterschiedlich beurteilt werden können, 2) Relevante Kontextinformationen (d.h. Kontextfeatures) anderes sein können, wenn unterschiedliche Kontextpaare verglichen werden 3) Aufschlussreiche Kontextinformationen dem Benutzer angezeigt werden müssen, um die getroffenen Entscheidungen zu verdeutlichen. Diese Funktion wird in der Arbeit mit der Nearest Neighbour Methode verglichen.

Der dritte Schwerpunkt erläutert das Thema des applikationsspezifischen Wissens im Rahmen der Adaption von Anwendungen. Das Framework definiert die Trennung des Adaptionsprozesses von der Anwendungen. Da jedoch die Charakteristik jeder Applikation diese Adaption beeinflusst, müssen die anwendungsspezifischen Eigenschaften dem Softwaresystem bekannt sein. Diese Eigenschaften sind als Application-Specfic Knowledge definiert und werden in Form von Templates an das System weitergegeben.

Abschließend ist eine prototypische Anwendung – The Call Profile Manager - implementiert worden, um die Ausführbarkeit des Ansatzes zu demonstrieren und zu evaluieren. Für diese Anwendung wurde die Specific-Knowledge definiert.

Acknowledgement

This work was developed during my time as a member of the Chair for “Communication Technology” (ComTec) at the University of Kassel, Germany. Many people helped me to carry out my research and permitted me to gain a very exciting and valuable experience in Germany.

First of all, I would like to thank Prof. Dr. Klaus David for enabling me to work at his Chair and for serving as a supervisor of this thesis. I will always remember the time at his Chair as a very enriching professional period.

Furthermore, I would like to thank Dr. Sandra Haseloff for supporting this work in providing me with numerous advices, very helpful discussions and comments during my scientific work, as well as for proofreading earlier versions of this thesis. I am also deeply indebted to Dr. Olaf Drögehorn for enabling me to gain much valuable professional experience in working with him in international research projects and for acting as a very thorough referent in these projects. I will always remember the exciting conferences and project travels we had together.

The work presented hereafter has also been performed with the help of my colleagues at the “Chair for Communication Technology”. I would then like to thank all my former colleagues for the harmonious working atmosphere in our group, the inspiring discussions and their strong support in the project work. Thus, I would like to express my thanks to Alexander Bolz, Nermin Brgulja, Gabriel Cristache, Waltenegus Dargie, Christian Deist, Alexander Flach, Matthias Hildebrand, Hilko Hoffman, Niklas Klein, Immanuel König, Rico Küsber, Susanne Lacroix-Mehrmann, Tino Löffler, Timo Melchert, Guihua Piao, Andreas Pirali, Ulrich Siebert, Katrin Sieleman, Stephan Sigg, Michael Sutterer, and Björn Wüst.

The initial idea of this work was generated from discussions with Alejandro Martín Pajares, at the time he was working on his master thesis. Thanks to him as well.

I would like to thank my office *companions* Sian Lun Lau and Thomas Hohmann for all our interesting conversations about very diverse topics, sometimes far beyond Context-awareness.
I owe a special thank to Tom Norberg for proofreading and correcting many chapters of this thesis.

Finally, I would like to thank my Family and my friends for their continuous support during my work in Kassel.

Kassel, March 2008

List of Publications

In the course of my research activity on context-awareness and personalisation at the Chair for “Communication Technology”, I contributed to the following publications:

In conferences and workshops

S. Lau, J. Millerat, M. Sutterer, N. Brgulja, O. Coutand, O. Droegehorn. Integrating Expert Knowledge into Context Reasoning in Pervasive Computing. In *Proc. of the 7th international Workshop on Applications and Services in Wireless Networks (ASWN 2007)*. Santander, Spain. June 2007.

O. Coutand, S. Haseloff, K. David. Dealing with Application-specific Knowledge in Context-Aware Systems. In *Proc. of the 2nd Workshop on Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (SAKS '07)*, in KIVS 2007, pages 195-206. Bern, Switzerland, March 2007.

M. Sutterer, O. Coutand, O. Droegehorn, K. David, K. van der Sluijs. Managing and Delivering Context-Dependent User Preferences in Ubiquitous Computing Environments. In *Proc. of the Workshop on Next Generation Service Platforms for Future Mobile Systems (SPMS 2007)*. Hiroshima, Japan. January 2007.

O. Coutand, S. Haselhoff, S. Lau, K. David. A Case-based Reasoning Approach for Personalizing Location-Aware Services. In Proc. of *the Workshop at the 8th European Conference on Case-Based Reasoning (ECCBR 2006)*, pages 247-259. Ölüdeniz/Fethiye, Turkey. September 2006.

O. Coutand, S. Lau, S. Haseloff, O. Drögehorn, K. David. An Approach Using Memory-Based Reasoning for Determining the Behavior of Mobile, Location-Aware Services. In *Proc. of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006)*, pages 43-52. Kassel, Germany. June 2006.

O. Coutand, R. Haensel, S. Lau, D. Droegehorn, K. David. Interpreting Group Context for Personalized Mobile Services. In Proc. of *the 15th IST Mobile & Communications Summit*. Mikonos, Greece. June 2006.

R. Haensel, O. Coutand, R. Kernchen, R. Nanni, K Moesser, D. Droegehorn. Time Gems: Group Experience in Pervasive Computing Environments. In Proc of *the 15th IST Mobile & Communications Summit*. Mikonos, Greece. June 2006.

O. Coutand, S.L. Lau, M. Sutterer, K. David, O. Droegehorn. User Profile Management for personalizing services in pervasive computing. In Proc. of *the 6th International Workshop on Applications and Services in Wireless Networks,* pages 3-11. Berlin, Germany. May 2006.

G. Schultz, O. Coutand, R. van Eijk, J. Hjelm, M. Martin, R. Nani. Enablers for Privacy for Users Belonging to Groups. *16th Meeting of the World Research Forum (WWRF)*. Shanghai, China. April 2006.

R. van Eijk, O. Coutand, S. Holtmanns. Sharing of Preferences and Context in Groups of Mobile Users. In Proc. of *the 1st Workshop on Mobile Social Software, CHI2006*. Montreal, Canada. April 2006.

O. Coutand, S. Lau, K. David. Determining the behavior of mobile location-aware services based on user action history. *1st Workshop on "Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (SAKS '07)*. Kassel, Germany. March 2006.

O. Coutand, O. Droegehorn, K. David, P. Nurmi, P. Floréen, R. Kernchen, S. Holtmanns, S. Campadello, T. Kanter, M. Martin, R. van der Eijk, R. Guarneri. Group- & Context-Management in Mobile Environments. In Proc of *the 14th IST Mobile & Communications Summit*. Dresden, Germany. June 2005.

S. Campadello, O. Coutand, C. Del Rosso, S. Holtmanns, T. Kanter, C. Räck, B. Mrohs, S. Steglich. Trust and Privacy in Context-Aware Support for Communication in Mobile Groups. In Proc. of *the 1st Workshop on Context Awareness for Proactive Systems (CAPS) 2005*, pages 115-124. Helsinki, Finland. June 2005.

O. Coutand, O. Drögehorn, C. Deist, K. David. Modelling Design Decisions for developing mobile Service Platforms. *13th Meeting of the World Research Forum* (*WWRF*). Jeju Island, Korea. March 2005.

O. Drögehorn, O. Coutand, K. David. Platform for Service Driven Networks: Using a general architectural approach. In Proc. of *the 3rd International Conference on Networking (ICN'04)*. Gosier, Guadeloupe, France. March 2004.

O. Drögehorn, B. Wuest, O. Coutand, K. David. A Platform for Pervasive Computing - From Requirements (I-Cake) to Code. *8th Meeting of the World Research Forum* (*WWRF*). Beijing, China. February 2004.

As a book chapter

G. Schultz, O. Coutand, R. van Eijk, M. Miettinen, “Privacy, Trust and Group Communications”, Book Chapter in “Enabling Technologies for Mobile Services: The MobiLife Book”, ed. Mika Klemettinen, Wiley, ISBN: 978-0-470-51290-6, September 2007.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

It has been seventeen years since Mark Weiser published his visionary “Ubiquitous Computing” [Weis91], depicting the future of computing, where countless computers are seamlessly integrated into the environment, thereby becoming ubiquitous and ultimately disappearing into the background. “*The most profound technologies are those that disappear. They weave themselves into the fabric until they are indistinguishable from it*”, he explains in the preamble.

The vision of Ubiquitous Computing has motivated many research activities to address its different aspects. In order to become ubiquitous, computing systems have to overcome different challenges. Grounded on distributed computing, these systems have to be extended to provide e.g., an *effective use of smart offices* – offering evolved services to their inhabitants - or *some localized scalability* – constraining the provision of services to those located to a relevant physical distance from its users [Saty01]. However, to really become invisible and not even be noticed by their users, these systems have to run with a certain *autonomy*, thus performing tasks on behalf of their users and meeting their expectations.

Yet computing systems are far from being invisible or unnoticed to the user’s mind. A good approximation to this ideal is *minimal user distraction*, which can be achieved through context-awareness [Saty01]. Context-aware systems utilise information characterising a user’s context, encompassing e.g. where the user is, what he is doing, what objects are in the environment, and provide him with relevant information and services. Hence, these systems are able to respond to users after a minimal set of user interactions. This results in a more effective use of the user’s attention and concentration.

The work described in this thesis concerns the development of a framework to enable the personalisation of context-aware applications and strives to enable interactions between users and applications, mostly based on a simple positive-negative feedback basis. This chapter presents the context of the work and the issues it aims to address.

1.1.1 Personalisation in context-awareness

Providing applications tailored to the needs and preferences of the individual users has been the focus of research on user-adaptive systems for many years now. Users are already supported with e.g. recommender systems [Burk02], intelligent tutoring and online course [Brus01], access to information sources [FiKo02], and health-care assistants [HaBl+05]. Personal information about a user’s preferences, knowledge, abilities, emotional states and many other characteristics make it possible to facilitate the user’s experience through

adapted and reduced interactions. These systems support adaptation in different forms: For example, modifying navigation paths, filtering contents, selecting information, etc.

Pursuing this objective, numerous research activities in the field of context-awareness have already led to projects in various areas e.g. smart spaces, information retrieving and tourist guides (see [Kork00] for a review). All these systems share a commonality in that they utilise context to adapt the specific tasks they are designed for, according to the current user situation. However, they differ in the way they use context. Context can serve to ease the retrieving of information (e.g. [DaLö+05], [RaOu+05]), or trigger non-user specific tasks (e.g. [OjKo+03], [AaGö+04]).

This thesis focuses on the class of applications performing tasks on behalf of their users based on some user context-dependent information. Here, context is used along with user preferences to decide on the task to perform on behalf of the user. In fact, context is used to select the user's information indicating a specific task. Indeed, the relevance of a preference can vary highly in different contexts. For example, a user might be interested in cultural news while browsing the web at home, while he prefers scientific news when in the office.

Below, a typical use case scenario of personalised context-aware systems is presented. It features a "Unified Messaging Service" that allows a user to communicate via different modes depending on his current preference.

> *John is a businessman and he frequently travels either by car, train or plane. John's means of communication include phone calls, emails and SMS. Connectivity is a crucial factor for John in any situation. To keep an eye on his everyday business no matter where he currently is, John uses a "Unified messaging" service. This service allows John to remain connected with his business and private contacts at any time. It automatically manages inward and outward communication accesses, without John needing to explicitly specify anything. The decision is made for John by selecting the most appropriate approach to deal with communication needs with respect to John's preferences. In fact, John's mobile phone automatically adapts the service to his preferences in the current context.*
>
> *This morning John is on his way to a business meeting. He first travels by car and then by train. While driving his car he is able to receive phone calls on his mobile phone with his hands-free set in his car. Additionally, all phone calls to his office number, as well as emails, are redirected to his mobile phone. As our businessman is in his car at the moment, he does not wish to get emails in a written form, since he thinks it would distract him from driving. Instead he gets emails in audio form. Emails are transformed from text into speech beforehand by means of dedicated software.*

As soon as John boards the train, John's current context is updated and the settings of his communication device are modified accordingly. He can now read incoming emails and SMS, no longer needing speech transformation. In addition, phone calls are forwarded without modification.

Later on, John arrives at his destination to attend a meeting. In the meeting room his settings are automatically changed again. In order not to be disturbed, John prefers to be notified via text messages only when a set of pre-selected persons, such as his boss, and close relatives – wife, children – try to reach him. Incoming communications are thus filtered. Phone callers are invited to leave a message. The ones allowed to be transmitted are transformed into text by means of a module of speech recognition. All permitted texts are displayed on John's phone screen.

This simple scenario illustrates the role of personalised context-aware systems and the benefits they bring to everyday users. Today, using most software systems, users are still required to explicitly communicate their expectations, in pressing keys, typing text, etc, to select the task to be carried out. Context-awareness alleviates constraints on the user-system interactions and provides system functionalities with added-value for users. Tasks are performed as expected by users. Nevertheless, the interaction is kept simple and straightforward, and simultaneously user satisfaction is increased as context-awareness simplifies user experience. As pointed out in this scenario, this is made possible by considering context as clues to predicate user preferences, taking advantage of the relationship between user preferences and user context. Indeed, the user's expectations towards software systems change with the context: e.g. John prefers reading texts *when in a meeting*, but cannot afford to watch a screen for more than a few seconds *while driving*.

It is worth noting, the scope of personalised context-aware applications and systems does not limit itself to the single example of this "Unified messaging service". The following use cases exemplify the variety of such applications.

- **Desktop manager** - As a user starts his computer, his personalised context-aware system detects what his context is, e.g. where he is (whether in his office or at home), and what activity he is currently performing (reading, working, getting prepared for the next meeting, etc.). Applications the user usually accesses are launched automatically (e.g. email application, related previous documents, etc) and shortcuts on the desktop are reordered. As the context changes, (e.g. entrance of a colleague in the office), the applications available on the desktop and shortcuts are updated accordingly.
- **Music player** – As a user starts his music player, a playlist is composed of songs the user likes the most in the current situation. For example, when getting ready in the morning he likes to listen to rock music, whereas having come home after a long working day he prefers jazz music.

- **Call Profile Manager** – As a phone call comes in on the user's mobile device, context is analysed. The call is either blocked, when the user is in such a situation that he does not want to be disturbed, or the user is notified by different ring tones or vibration, in accordance with his current preferences.

1.1.2 Issues in personalising context-aware applications

Striving to build context-aware applications, researchers have faced great challenges, and have partially addressed some. This includes e.g., developing and using sensors to acquire information from the environment; building software infrastructure to process and interpret these pieces of information; developing management systems in order to manage and store contextual data for later retrieval; developing frameworks to address security and privacy in context-aware systems; developing mechanisms to reason with contexts; and representing contexts in a form suitable for machine processing [BaDu07].

This work treats personalisation of context-aware applications. It focuses on two key research issues: adaptation in context (how to enable the adaptation of some context-aware applications in unforeseen contexts involving minimal user directions) and privacy protection of user information towards applications (how not to divulge and to limit the handling by applications of user's personal information).

Adaptation in context

As discussed above, context-awareness aims at facilitating the user interactions with systems. It appears that the success and the proliferation of context-aware systems in daily life will depend on their acceptance by end-users. The effort to use such systems should be greatly outweighed by the benefits to users. While carrying out adaptation, these systems must cause minimal user distraction and not constantly request information to select behaviours.

Context-awareness can help personalisation by adapting the task to be performed based on the user's context. However, this implies that the systems in charge of the adaptation receive clues about 1) user preferences: what tasks the user expects to be performed and 2) the related context: the situation in which these preferences are relevant. Thus, information about a user, his beliefs and intentions and the contexts in which they are suitable are required.

These issues have already been investigated in previous research work [BaDu07]. However, the traditional way to address them consists of defining, before the system runtime, user preferences and their associated contexts and combining them into rules. The key problem with this approach is twofold. On the one hand, the traditional approach requires that the user thinks about his preferences in all possible future contexts, which can be arduous, since the number of situations is potentially very large and difficult to fully describe beforehand. On

the other hand, the rules associated with the traditional approach are not extendable. When a context occurs which has not been specified in any rule, no specific adaptation can be performed.

As a consequence, these existing solutions still hold limitations that make them unsuitable to tackle the issue of adaptation. In addition, though increasing the autonomy of systems through context-awareness decreases user's distraction, a user may still feel a loss of control over the systems. To mitigate this problem, context-aware systems providing personalised behaviours to their users must implement mechanisms suitable to balance user control and software autonomy [HaIn06].

Hence, a new approach to adaptation in contexts is required that would better address the peculiarities of the issue.

Privacy protection of user information towards applications

Context-aware applications rely on context to adapt aspects of their structure, functionality or interface. This thesis specifically considers context-aware applications, the behaviour of which can be adapted to match users' current needs. Adaptation is here, as in any adaptive system, governed by information such as the user's preferences and interests, which are contained in user models. However, information in user models of traditional adaptive applications is regarded as always-true and situation independent. In contrast, context-aware systems make use of user models which are context-dependent, since in various contexts a user can have different preferences.

Yet, dealing with context is a complex task [DeSa01]. A very popular approach to context-awareness consists of separating the acquisition and interpretation of context from the reaction to context by applications [DeSa01]. Following this approach, context-aware systems have been developed to provide middleware support to applications that adapt to user's context, e.g. [BiCa04], [FaCl04], [GuPu+04]. These systems deal with context information on behalf of the applications by:

1) Gathering sensor data from the user's environment and

2) Providing context descriptions in transforming sensor data into high-level and human understandable context information

In the current realisations, and following the separation of concerns from [DeSa01] context-aware applications are further provided with context descriptions. Applications use them to trigger adaptation (e.g. modifying the output mode - text or audio - or the types of information it displays) based on some context-dependent user models.

Adaptation relies on two types of user specific data: user preferences and user contexts. Thus, it appears that these two types of data are highly sensitive. Should this data become available to malicious parties, it could be very detrimental to the users. For example, knowing

the user's preferences and/or his current context enables unwanted parties to draw conclusions about their behaviours, habits, likings, etc. This threatens a fundamental right of users to privacy.

An approach is then required that would deal with this limitation and would prevent applications from obtaining knowledge of both the user's context and his preferences.

1.2 Thesis statement

In this research we aim to facilitate personalisation in the context-aware environment by addressing some of the aforementioned challenges.

The general goal is to develop concepts and methods for systems to facilitate user interaction with some context-aware applications by determining on behalf of the user the application behaviours to be performed.

As part of this work a framework for personalising context-aware applications has been developed, which calls for adaptation to be performed in a central context-aware system and infers a user's preference based on his previous experiences with an application. Thus, it enables personalisation to be performed in unforeseen contexts. It also prevents the user from inputting his preferences in contexts, making him interact with the application on a simple positive-negative feedback basis. Finally, it improves user data privacy, by preventing user personal data from being transmitted to the applications.

1.3 Thesis contributions

This research focuses on the personalisation of context-aware applications; in other words, it focuses on enabling the adaptation of applications based on context-dependent user preferences. The main contributions of this work are summarised in the following sections:

CBR-based Personalisation Framework

Firstly, I have developed a framework supporting the determination of the user's preferences in any contexts to enable the personalisation of context-aware applications. In this work, personalisation consists of selecting an application behaviour based on some user preferences and the description of the user's context.

The framework features two principles:

- Separation of the adaptation process from the applications

By applying the single access point architectural pattern [YoBa97], adaptation can be performed in a context-aware system rather than in applications. Then user data (context and preferences) are centrally processed and are not divulged to applications. This improves user data privacy. It also enables a central user model to be maintained where information is

stored in a non-redundant manner and serves several applications.

- Inference of user preferences based on previous user's experiences with the application

The approach implemented in the framework relies on analogical problem solving [GiHo80]. It is based on the assumption that in similar contexts, the user has similar needs with regards to an application.

When personalisation is required, the application behaviour to be triggered is determined using a user's previous experiences. Any user's experience is expressed as a case in two parts: a context description (problem situation) and the application behaviour that was then triggered (solution). Following the CBR cycle [AaPl94] the personalisation of context-aware applications is performed through the following steps:

- Step 1: A description of the user context is built and expressed as an *n-tuple*, where each feature characterises a distinct aspect of the context (e.g. location, time, etc).
- Step 2: The context description is compared to the previous user's experiences. A retrieval function is in charge of fetching a set of experiences that best match the new problem situation. The retrieval function is further discussed in the next subsection.
- Step 3: The solution parts of the retrieved experiences are utilised to determine the user's preference corresponding to the new problem situation. The application is correspondingly triggered. An explanation, motivating the system selection, is displayed to the user.
- Step 4: The user's reaction is monitored and the user's feedback is recorded. The application behaviour is corrected if needed, and the problem situation with the correct user preference is retained for further use.

Cluster-based retrieval function

I have developed a retrieval function in order to fetch the most similar previous contexts. The function compares and assesses a degree of similarity between the new context and those of the user's previous experiences. The development of the retrieval function is guided by three design principles:

- Similarity assessment between contexts is user-specific

Similarity between contexts can be assessed differently for different users.

- Similarity assessment between contexts is feature-specific

When assessing similarity between different pairs of contexts, the relevant features may differ.

- The results of the retrieval function must be explained to the user

Revealing aspects of context must be displayed to explain to the users the reasons for the previous experiences' selection.

The developed retrieving function addresses these three principles and determines the application behaviour in several steps. The method aims to define meaningful and general contexts or contextual states from several experiences. These general contexts are characterised by a subset of the features and not by all the features as in every single experience. To do so, the previous user's experiences are first classified according to the application behaviour to which they are related. Second, for each class, experiences are clustered in order to bring out their commonalities in terms of context features' values. The current context is then compared to all the defined clusters and the most similar cluster is retrieved and presented to the user.

Application-specific knowledge

As defined in the framework, the adaptation process is performed separately from the applications. However, adaptation arises differently for various applications. Indeed, applications have their own characteristics that influence adaptation, e.g. they may not be sensitive to the same context information. Therefore, to be able to perform adaptation of any application, a system has to be provided with the application characteristics. These characteristics are referred to as application-specific knowledge.

I have distinguished between: 1) the behaviour description, detailing all the behaviours the application can produce, 2) the context features, describing the context features along which the application can be adapted, and 3) the adaptation knowledge required by the reasoning mechanism that triggers the adaptation. I propose to express the knowledge with templates that are communicated to the context-aware systems by the applications.

Additional contributions

In addition to the aforementioned three main contributions, additional minor contributions have been made.

- Development of the Call Profile Manager application

To show the feasibility of the framework and evaluate the approach, I implemented a prototypical application, the Call Profile Manager. The application blocks or notifies the user of any incoming call according to the user's location. Notification is performed either through a vibration mode, a normal ring tone or a louder ring tone.

- Application specific knowledge for the Call Profile Manager

Specific knowledge to the Call Profile Manager has been investigated and defined. From the envisioned scenario, five features related to the user's location have been selected to represent context. In addition, similarity metrics were developed for the five different context features. For every type of specific knowledge, templates were also designed.

- Development of the framework

Finally, a system implementation of the personalisation framework has been developed.

However, it does not implement the operation consisting of gathering context. In order to perform this operation the personalisation framework can be integrated to the Foxtrot framework. The Foxtrot framework [LöSi+06] provides a flexible architecture for context-aware systems and has been developed by the Chair for Communication Technology, at the University of Kassel.

1.4 Thesis outline

The remainder of this thesis details the approach that has been developed to support the personalisation of context-aware applications and that addresses the limitations of current realisations. It is structured in nine chapters.

Chapter 2 introduces the main concepts related to the personalisation of context-aware applications. Starting from the general considerations on adaptive systems on which this work is based, it presents a review of the main activities in, and the current achievements of, the field of context-awareness. In this chapter, a taxonomy for context-aware applications is discussed. This work only focuses on one of these classes of applications, referred to as contextual personalised applications. This chapter then puts into perspective the fields of adaptive systems and context-awareness. We discuss how they can cooperatively support the personalisation of context-aware applications. Finally, a specific technique of Machine Learning is discussed: Case-Based Reasoning (CBR). This technique is used in this approach to address some of the aforementioned limitations. Of particular importance in this chapter is a presentation of definitions of fundamental terms relevant to this work.

Chapter 3 provides an overview of approaches related to this thesis. Firstly, the fundamental requirements to the personalisation of context-aware applications are presented. Secondly, the approaches currently addressing problems similar to the one treated in this thesis are summarised and their differences are confronted. Finally, their limitations with respect to the novelties we introduced in this work are highlighted.

Chapter 4 describes Copernik - Contextual Personalisation Framework - a framework, that supports contextual personalised applications via a set of methods and concepts. Firstly, the objective of the framework is clarified and the basic assumption made in this work is underlined. Secondly, the design principles governing the development of the framework are detailed. Fulfilling these principles in the framework makes it possible to tackle most limitations of current approaches related to personalisation of context-aware applications. A major part of the chapter details the so-called personalisation cycle that specifies the various phases any contextual personalised application in compliance with the framework has to go through, in order to provide users with adapted behaviours. Finally, the characterisation of the framework ends with the description of the approach taken to tackle the issue of privacy protection of user information, mentioned in this introduction chapter.

In **Chapter 5**, one central operation of the personalisation framework is discussed: the retrieving of past experiences. This is a crucial feature of the framework. It enables personalisation through the definition of an adaptation strategy that best matches the user's current needs. The strategy is inferred from the comparison between the current context and stored user's previous experiences. This chapter describes the approach in different steps. Firstly, the retrieving task is discussed and the design principles as well as the requirements that drive its development are detailed. Techniques, which appear as potentially good candidates to perform the retrieving task, are also reviewed and compared. The results of the comparison influence the selection for agglomerative clustering. Secondly, the fundamentals of agglomerative clustering are examined. Finally, the cluster-based approach that enables the retrieving of similar cases is detailed.

In order to deal with the issue of user data privacy, the framework features a separation between adaptation and applications, as presented in chapter 4. Consequently, knowledge about every single application is needed to perform adaptation. In **Chapter 6**, we examine what types of specific data are required and how data are made available to the system carrying out adaptation. Three examples of contextual personalised applications are presented. Differences in their adaptation process are highlighted. These differences are further exploited to characterise three types of data, which support contextual personalised applications. Such data are referred to as *application specific knowledge*. Finally, the chapter ends with a presentation of the templates permitting applications to deliver their specific knowledge to the system implementing the personalisation framework.

Chapter 7 deals with contextual personalised applications. First, potential scenarios for contextual personalised applications are given and a process for developing such applications is detailed. Then, the Call Profile Manager application is presented and its application-specific knowledge is detailed.

In **Chapter 8** a prototype implementation of the framework is described. The implementation demonstrates the feasibility of the framework's realisation. It also serves as a test-software to evaluate the approach. Thus, this chapter is structured in two parts. The main system requirements and the architecture of the prototype are first discussed. Following this, the general operations occurring between the system components are depicted in message sequence diagrams.

Chapter 9 evaluates the retrieval function, discussed in Chapter 5.

Finally **Chapter 10** concludes the thesis by summarising the main scientific contributions and giving an overview of the open questions and future research directions.

2 Foundations for Contextual Personalisation

This chapter introduces the fundamentals of this work. This thesis is at the crossroads of two research areas: user-adaptive systems (also referred to as user-modelling systems) and context-awareness.

On the one hand, user-adaptive systems aim to enable users to access information or services with adapted and personalised interactions. Issues in this field include the gathering and the utilisation of information about the preferences, knowledge, abilities, emotional states, and many other characteristics, of users to make systems more usable and to automatically adapt them to their users' needs. On the other hand, context-awareness promotes the use of information related to the user's environment to command computing systems. Though these two areas have been investigated separately and have evolved independently from one another, it appears that context-awareness can duly complement and enhance user-adaptive systems to support personalisation for Ubiquitous Computing systems.

This chapter outlines the fundamentals of contextual personalisation, a term coined in this thesis to designate the personalisation of context-aware systems. In the first two sections, the main characteristics and the fundamental issues of user-adaptive systems and context-awareness are presented. The third section emphasises how these research areas can complement each other in order to provide personalisation of context-aware systems. We discuss and define in these sections the key concepts. Finally, since a major part of this work relies on Case-based Reasoning (CBR), a technique of machine learning, the fundamentals of CBR are presented.

2.1 User-Adaptive systems

Human Computer Interaction (HCI) studies the interactions and the relationships between humans and computers. Improvising the usability of computer systems and providing users with experiences fitting their specific background knowledge and objectives is a fundamental goal of HCI research. Traditionally non-adaptive systems have considered a stereotyped user and have been providing a unique set of capabilities to anyone using them. However, in the 80s, this was identified as a flaw, as the need for improving human-computer interactions in providing behaviours adapted to their user was raised by system developers. Thus, the idea of adapting software to their users emerged in the software community. The advent of the WWW in the 90s, resulting in a huge amount of available information, has impelled the development of adaptive systems in domains such as e-commerce and web browsing [Kobs93] [Kobs01a]. These systems have been able to tailor their behaviours to individual needs, while catering to a wide range of WWW users. Today, following the rapid development of mobile devices, new adaptive systems are being investigated to cope with

specific requirements in the mobile environment, e.g. scarce resources of devices, user interfaces of small size.

Before the different functions of adaptive systems are presented, it is important to limit the scope of these systems and define the main concepts related to this field.

2.1.1 Scope of user-adaptive systems

The aim of user-adaptive systems is to accommodate the differing requirements of individuals or groups of users and their changing needs over time. To do so, user-adaptive systems are requested to provide adapted functionalities for each individual user, and thus must be able to alter aspects of their structure, functionality or interface on the basis of information about the user [BeIn+87].

This highlights the two basic characteristics of user-adaptive systems. First, such a system has the ability to alter its structure, functionalities, or part of them. Second, the aim of the system is to behave as a user expects, in a different way for different users.

Definition 2.1: **User-Adaptive system** from [Weib02].

A system is called adaptive only if it is a system that changes its behaviour depending on the individual user's comportment on the basis of information about the user.

Since this definition relies on the notion of system, a definition of the term must be given.

Definition 2.2: **System** from [IEEE00]

A collection of software components organised to accomplish a specific function or a set of functions.

The approach to adaptation can be classified into two categories: *customisation* and *automatic adaptation* [FiKo00]. The difference between these two classes can initially appear minute. Both allow users to get system functionalities that fit their needs and requirements. They differ, however, in the extent to which users can exert influence on the adaptation process of a system, whether explicitly or implicitly.

The first category pertains to systems that *offer* their users the capability to select some alternative presentations or interaction characteristics. This selection is made among the set of capabilities built into the system in an explicit way by users. The communication between computers and users remains as in the classical non-adaptive systems. Hence, though it alleviates the constraints due to the "one-fit-all" paradigm, customisation does not really improve usability. Systems enabling this type of adaptation are referred to as adaptable

systems [FiKo00]. Web portals, such as Yahoo.com [YAHO], are examples of adaptable systems. There, users manually choose the preferred fields of interest, selecting news from politics, economy, etc, as well as the favoured design of the web page. However, this approach suffers from the usually limited set of adaptation facilities, which are most of the time pre-packaged [Step01].

The second approach to adaptation is automatic adaptation. Here, systems should be capable of identifying the circumstances that necessitate adaptation and accordingly select and affect an appropriate course of action. This implies that systems possess the capability to monitor user interactions. Systems that adapt to the users automatically based on their assumptions about them are called adaptive systems [FiKo00]. Amazon.com [AMAZ] is an example of an adaptive site. Information about the user is collected with the help of different evaluation methods. In the case of Amazon a collaborative filtering technique is used that evaluates former purchases and suggests or recommends products that are supposed to be specifically interesting to the user.

This second approach differs radically from the traditional computer usage, which is modelled as a human-computer relationship in which the two are connected by an explicit communication channel. A user interacts with the machine via a user interface by e.g. entering a set of key strokes to determine the task to be performed. In contrast, automatic adaptation has explored an implicit communication channel, in which no “typical” user of a system is assumed that would select on his own the task to be performed. Rather, many different users are considered, with different goals and expectations with regards to the systems, based on their personal experiences, bodies of knowledge, state of mind, etc. This information about the user serves as inputs to the systems enabling them to operate, thus delivering information or services.

This implicit communication channel should support communication processes that require computer systems to be provided with the following types of knowledge [Fish01]:

- Problem domain knowledge that constrains the number of possible actions and describes reasonable goals and operations in the problem domains (i.e. how a system can be adapted);
- Knowledge about the communication processes that states when the user should be assisted or interrupted by the system (i.e. when adaptation occurs) and
- Knowledge about the communication agent (user) that states what the user knows and needs (how the user wants the system to behave).

Both aforementioned approaches to adaptation can contribute to easing and improving user experience with computers. However, it seems that automatic adaptation holds the greatest promise to construct systems that are capable of catering for the needs of users. A rapid review of literature on adaptation indeed supports this idea, as today research mostly seeks to address the issues and the characteristics of automatic adaptive systems (see, for

example, literature in user modelling and user interactive systems [UMUA]).

Over the last few years, the term personalisation has gained much attention and is often referred to as this class of adaptation. In this thesis, the term personalisation is preferred over the term automatic adaptation. As it is used extensively through all the following chapters, the following definition is proposed:

Definition 2.3: **Personalisation**

Personalisation is the task of providing adapted capabilities to the users of a system, on the basis of implicitly gathered user information.

The process of personalisation can be distinguished in two phases [FiKo00]: personalisation, which includes user modelling and adaptation decision (or user-adaptive feature).

- User modelling consists of acquiring user information (needs, goals, knowledge, interests or other characteristics), which is deemed relevant for the service. This information is stored in a user profile.
- Adaptation decision refers to the process that occurs as mathematical models or analytical techniques are applied to user model data in order to make decisions, aiming to improve selected aspects of the interactions between the user and the service. This process can be adaptation at e.g. content level, hyperbase level (i.e. navigation structure of web page), presentation level (i.e. the layout of each page together with user interaction facilities) [KaPr+03], or recommendation, which attempts to predict items that a user may find interesting [Burk02].

2.1.2 User modelling

The need for computer systems to automatically adapt to their current users is generally acknowledged, e.g. [GoAm05] [Kobs93] [Saty01]. It has also become evident that systems can acquire the ability to decide when and how to perform adaptation only if they possess information about users' background knowledge, their goals and plans in consulting the system, and their preferences, misconceptions and attitudes. Unlike classical non-adaptive systems, the interaction scheme is not governed by a set of user manipulations, but rather relies on user data. Thus systems require that a software agent maintains a model of the user, where the user information is contained.

A user model serves as a description of the system user and as a prediction of how it will behave and perform tasks. It enables services to be adapted to the given user, based on the selected relevant information.

Definition 2.4: **User model** [Fish01]

A user model is defined as a model that a system or several systems have of a user residing inside a computational environment.

The selected definition does not make any assumption about the types of information that comprise a user model. In fact, it is the specific demands for personalisation that will determine the characteristics which should be provided by a user model. Assumptions about the current user's goals will be deemed relevant for a plan recognition system that tries to determine all possible sequences of user actions to be taken to achieve an objective. In contrast, user preferences are much more significant for product recommender systems.

As a consequence, according to the area of the system and the form adaptation takes, a large variety of user data can be part of a user model. These data can include:

- Cognitive processes that underlie the user's actions (i.e. user's plans)
- Relevant differences between the user's skills and expert skills
- User's characteristics (e.g. user name, address)
- User's behavioural patterns (e.g. user preferences with regards to an application)

Research in the field of user modelling investigates how such assumptions can be automatically created, represented and exploited by the system in the course of an interaction with the user. In a nutshell, user modelling consists of identifying the pieces of information most relevant for making decisions on the system adaptation and creating those assumptions based on acquired user information, via various methodologies, methods and techniques. As Orwant observes in [Orwa96], the user modelling task is not specific to user-adaptive systems. In fact, it is very commonly performed by humans:

"Humans model each other all the time. I am modeling you as I write; my topics, presentation, and language are all aimed at a hypothetical average reader of this journal. If I have guessed well, you will enjoy this essay. If not, you will skip to the next one. That is what user modeling system do – they make guesses, and hopefully educated ones, about their users."

Acquiring user knowledge

Acquiring user information and inferring knowledge about the user is a fundamental issue in user-adaptive systems [Kobs93] [WePa+01]. User modelling involves inferring *unobservable* information about a user from *observable* information about him, his actions or utterances. Different methods can be deployed to compile user knowledge. These methods can be distinguished according to the way knowledge is acquired from users: whether explicitly or implicitly [HaSh+01].

- Acquiring user knowledge explicitly

Early user-adaptive systems relied on hand-crafted knowledge bases. A knowledge base is usually built by selecting and analysing at hand several instances of the problem, which are deemed to be representative. Knowledge about these instances can be obtained from user interviews, interrogating the users about their preferences and expectations. For example, users are required to fill out a paper or electronic form to answer a set of questions related to their areas of interest, or any other type of information that is needed for some specific problems.

However, this approach leads to a classical problem in Artificial Intelligence: the knowledge bottleneck problem. Knowledge is difficult to acquire since knowledge bases result from a resource-intensive process [BuBa+83]. As the knowledge bases are developed by some programmers whose task is to transform the user's knowledge (preferences, interest, etc) into code, this requires that the programmer learns enough about the domain to converse effectively with the user, i.e. he knows what user knowledge is relevant and how to efficiently express it in the system [BuBa+83]. This process is highly time-consuming. It is acceptable for application domains where a very limited number of users are domain experts, and provide expert knowledge to any other user; it is absolutely not appropriate for personalisation systems, where each single user is his own expert. Besides, once knowledge bases are developed, they are usually neither adaptable nor extendable [ZuAl01].

- Acquiring user knowledge implicitly

An alternative to the hand-crafted construction of user models is to build predictive statistical models [ZuAl01]. Such an approach relies on the recording of user behaviours and uses the observed sample results to make a statement about the user's future behaviours, preferences, etc. Two approaches are considered to build such predictive statistical models:

- Content-based acquisition knowledge

  In the content-based approach, the user's behaviours are predicted from his past behaviours when they are considered as a reliable indicator. It implies that the user's behaviours are constantly monitored and recorded to determine some data items. The relevancy of these items can then be learned according to the user's reactions, e.g. if the user selects the items or drops them. Hence, this requires no active user involvement in the knowledge acquisition task. Rather, the more the user interacts with the system, the more knowledge is acquired.

- Collaborative-based knowledge acquisition

  In the collaborative-based approach, the user's behaviours are predicted from the behaviours of other like-minded people. The approach is used when a user behaves in a similar way to other users. A model is then built upon data from a group of users and is utilised to make the predictions about an individual user.

- Mixed approach

The mixed approach lies between the explicit and the implicit approaches. Similar to the implicit approaches, it involves inferring past user's behaviours to make statements about new user model information. However, it requires minimal user involvement.

Document space is such an approach that was developed in the field of Information Filtering [FoDu92]. It is based on the creation of a field of documents that the user has previously selected as relevant (explicit user interaction). Any new incoming document is then compared to the documents existing in the space, via a similarity measure. Any document in the space, which is found having a similarity above a given threshold, is considered relevant.

In the early times of user-adaptive systems a popular mixed-approach to user modelling was the stereotype approach. Stereotypes were useful for application areas, in which a quick but not necessarily completely accurate assessment of the user's background knowledge was defined by the system developer [Kobs93]. As a first step, a user is asked to provide some information about himself. According to the information he is assigned to a predetermined class of users, referred to as stereotype. Hence a stereotype captures default information about a group of people and provides a source of initial default information to model the users, as no other information is available. The use of stereotypes can be combined with a more implicit approach in order to provide an initial user profile that can be enhanced and modified as the user's interactions are recorded.

Explicit approaches and mixed approaches hold severe limitations when acquiring user knowledge for user-adaptive systems. This has probably contributed to the rise of interest for implicit approach-based methods, which rely on recorded user behaviours [WePa+01]. Observations of the user's behaviours can provide training examples that a machine learning system can use to form a model designed to predict future actions.

Machine learning techniques for implicit user knowledge acquisition

Machine learning typically encompasses techniques where a machine acquires and learns user knowledge to make predictions or reasoning. Several techniques from machine learning have been used for both the content-based and the collaborative approaches (see e.g. [ZuAl01], [FrMa+05] for more complete discussions). Some of these techniques include:

- Linear models

Linear models take weighted sums of known values to produce a value for an unknown quantity. For example, considering the rating of news articles, a new user's linear model may be computed from the ratings assigned by other users and the similarity measure between the new user and the other ones (weights).

- TF-IDF-based models

The Term Frequency Inverse Document Frequency method (TF-IDF) is commonly used in the field of Information Retrieval to find documents that match a user's query. A document is represented by a vector of weights, where each weight corresponds to a term in the document. The similarity between two documents is measured by applying the cosine similarity function.

- Markov models

Such models are based on the Markov assumption that the occurrence of the next event depends only on a fixed number of previous events. Thus, given a number of previously observed events (e.g. previously visited Web pages) the next event is predicted from the probability distribution of the event, which have preceded it.

- Classification

Classification methods partition a set of objects into classes according to the attribute values of these objects. Classification methods are unsupervised, since no information about the class to which the new observation belongs to is known a priori.

- Rule induction

Rule induction consists of learning a set of rules from the previous observations. Rules are used to predict the class of a new observation from its attributes. Unlike classification, rule induction requires the class of an observation, in addition to its attributes. Models implementing rule induction represent rules directly (e.g. as an IF – THEN statement), as decision trees, or in terms of conditional probabilities.

All these approaches are potential techniques for performing adaptation in contexts. Their applicability to address the specificities of this work - contextual personalisation - are discussed in Chapter 5.

2.1.3 Adaptation decision

Adaptation decision refers to the phase where the system decides about the concrete adaptation steps that are to be taken. To do so, the system utilises user information determined in the user modelling phase. Different forms of adaptation strategies can be defined. For example, the graphical interface can be changed so that contents, commands, functions, and annotations are modified.

In the field of adaptive hypermedia, one distinguishes, for example, between two broad areas for adaptation*: adaptive presentation* and *adaptive navigation support* [Brus01]. Adaptive presentation is concerned with text adaptation - sorting, removing and altering text fragments to better fit the user's interests -, adaptation of multimedia technology, and adaptation of modality – selecting between the different information modes available: video, audio, text. Adaptation navigation support aims to facilitate the user's access to information in hypertext

through some changes in the link structures and their rendering, e.g. hiding, modifying, or annotating them. Taxonomies have been developed that present classifications of the various adaptation techniques in this domain (e.g. [Brus01]).

Such taxonomies are specific to the area of adaptive hypermedia and other techniques are employed to support adaptation for different activities. In fact, the choices of the adaptation strategies depend on many factors such as the application domain, the tasks to be performed, the target users, etc. As a consequence, an in-depth overview of adaptation techniques is not relevant in this thesis.

2.1.4 Application fields

Numerous cases in which adaptation techniques have been used to improve the performance of existing computing systems or to enable the exploration of new dimensions in computing have been pursued in various application domains. In particular, in the last decade, the provisioning of personalised advice and services has been widely investigated in the World Wide Web [FiKo00]. This has led to the development of many commercial and academic systems. [FiKo00] and [Weib02] have given a list of various domains in which user-adaptive systems perform tasks on behalf of users. Such a list is not meant to be exhaustive, but gives a good overview of the variety of the systems' applications.

- Adapting an interface

An adaptive system can be deployed to modify and personalise the display of meaningful information to the users. This can be profitable in environments where applications run on platforms (devices) equipped with a user interface of small size [Lang99] [Höök00]. [ArCo+01] presents a system that tailors the presentation of news to the user's peculiarities. This can improve the precision and the recall of, for example, news in a news filter system.

- Giving help

Depending on the user's expertise or knowledge, a system can provide help on commands [HoBr+98]. For example, the adaptation of the help menu in a text editor can significantly improve the task performance and decrease the error rate. Adaptation can also provide navigational aid when scrawling in hypertext.

- Helping the user find information (adaptive hypermedia)

Users can be overwhelmed by the amount of retrieved documents when searching large information spaces, such as the Web or literature databases. A major limitation of traditional "static" hypermedia applications is that they provide the same page content and the same set of links to all users. Personal agents can then be deployed to provide assistance to users in navigating these spaces. For example, an electronic encyclopaedia can personalise the content of an article to augment the user's interest [Milo97]. Other examples of such agents include [PaBi97], [GoAm05].

- Recommending products, items and services

Recommender systems assist users in making choices by suggesting suitable products, items and services that best match the customer's needs and preferences. A review of recommendation techniques can be found in [Burk02].

- Supporting learning (user adaptive tutoring systems)

Users may benefit from adaptive systems that seek to improve the overall learning progress. These systems, referred to as user-adaptive tutoring systems, provide personalised Internet-based courses and trainings or support standalone applications. They facilitate the learning process in, e.g. adapting the curriculum to the current user's knowledge [Brus01].

Other tasks such as conducting dialog, getting help with routine tasks, or supporting collaboration have also profited from the development of user-adaptive systems.

2.2 Context-awareness

A crucial theme in Ubiquitous Computing is the use of context to facilitate and mediate communication with computing systems [Weis91]. Context designates aspects of the user's situation, encompassing various facets, such as his activity, his location and time, to name a few. Built on the notion of context, the term context-awareness designates the capability of systems to incorporate context information into their behaviours. Thus, context-awareness examines and reacts to a user's changing context in order to help to mediate people's interaction with each other and their environment [ScHi+02].

The primary motivation for the use of *context* in computing systems is the lack of information that is transmitted from individuals to computing devices in traditional man to machine interactions. Indeed, human communication is not restricted to the words they exchange. A large amount of information resides in situational information that humans take implicitly into account while they communicate. For example, gestures and facial expressions give a lot of additional indications about one's feelings, state of mind, etc and play, as such, a crucial role in conversations. Should this information be absent, as during a phone conversation, communication is more difficult, sometimes leading to misunderstandings. Today's computers are not able to make use of such additional situational information. Rather, they require explicit information. By improving the computer's access to context, it is therefore possible to increase the richness of communication and make it possible to produce more useful computational services.

2.2.1 Context-awareness and context

The notion of context has played a significant role in numerous domains for a long time. To only name some of the domains related to artificial intelligence (AI), context has been investigated in natural language processing, databases and ontologies, vision, and of course

communication [Brez99], [Henr03] (pp.19-20). For each of these domains, researchers have discussed the very nature of context and its influential factors on the specific domain issues. As a consequence, reviewing and commenting definitions of the term context could easily be the subject of a doctoral thesis on its own.

In this work, the notion of context is treated in the framework of context-awareness, which appears to be a body of research within the field of HCI. It explores novel forms of human-to-machine or human-to-human interactions that can be achieved by making computer technology aware of the physical world in which it lives. Computing systems that support these interactions are called context-aware. In the remainder of this work, we adopt the definition given in [Dey00] and refer to any context-aware system as:

Definition 2.5: **Context-aware system** [Dey00]

A system is context-aware if it uses context to provide relevant information and / or services to the user, where relevancy depends on the user's task.

This definition is not restricted to any specific subtask to be performed by these systems, such as sensing and interpreting environmental information. It highlights the interaction forms, that are supported: human-to-machine interactions, enabled by the provisioning of information and services to users, and human-to-human interactions, via communication services.

However, the above definition relies, however, on the use of the term context. It is then worth looking for a definition of the term. The first approach to clarify this term simply consists of replacing it by some synonyms such as situation or circumstances. This helps to get an intuitive understanding of context, but falls short when a scientific definition is required to delimit the concept's boundaries.

Below, we briefly present the main directions and arguments for achieving a definition of context. However, note that this section does not attempt to be exhaustive.

The definitions of context used in the context-aware related literature fall into two categories.

The first attempts in developing context-aware applications have focused on defining context by listing specific entities that compose context. Hence, [ScTh94] refers to context as location, identities of nearby people or objects, and changes to these objects. Later Schilit et al. refine the definition in [ScAd+94] and state that important aspects of context include, where you are, who you are with, and what resources are nearby. The same aspects are present as in the previous one: location, identities and objects and people nearby. But the scope of these aspects is either on one side constrained, i.e. only people socially interacting with the user are considered, or on the other side extended, i.e. resources are not limited to physical objects any longer but can include e.g. software pieces. Explicitly enumerating

aspects of context is, however, too specific and is only suitable for a limited number of application domains. For example, it excludes the user's activity, while this has been proven to be relevant for characterising some situations.

Beside this operational view on context, researchers have also attempted to define context more formally. Pascoe defines context to be *the subset of physical and conceptual states of interest to a particular entity* [Pasc98]. A well-known and very often cited definition in the context-awareness area literature is the one given in [Dey00]: *Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves*. This definition explicitly introduces the two main stakeholders involved in communications supported by context-aware systems: the user and his application. But it does not limit itself to them only and can encompass additional stakeholders- entities - such as other people, objects. The definition also makes it possible to define different aspects of context for various application scenarios. If a piece of information appears to be relevant to characterise the situation of a participant in the interaction, then this information is context. The same piece of information can be assessed as irrelevant in another scenario.

The definition in [Dey00] addresses the limitations of the previous and aforementioned ones. Firstly, it does not delimit entities (e.g. location, activity) as part of context, thus not excluding de-facto any other that would be relevant for new types of applications. Secondly, it provides more than a simple equivalent, and as such does not defer the issue to the definition of another term.

Yet, a major weakness in this definition resides in the introduction of the term situation, which has never been precisely defined (though illustrated through examples) [Henr03], [Hase05]. To address this deficit, a definition of the missing term is likewise required. In common language, situation designates *the relative position or combination of circumstances at a certain moment* [MERR]. [Hase05] (pp 13) extends this notion and proposes to refer to situation as: *a part of the world state at a specific point in time or within a specific time interval*. This thesis adopts this definition.

Definition 2.6: **Situation** [Hase05]

A situation is a part of the world state at a specific point in time or within a specific time interval

---

This work proposes a definition of context that is more specific to the scope of this work. This is discussed in section 2.3.

2.2.2 Dealing with contexts

In the previous section, some definitions for context as coined in the field of context-awareness are discussed. Understanding and defining context, whatever the definition is, is a paramount step in the process of building systems for Ubiquitous Computing. However, a definition alone is not sufficient. Context has to be represented in such a way that it can be handled by software systems. In particular, context characteristics governing the construction of the context representation have to be recognised.

Context characteristics

Context is a highly dynamic construct, as many contexts are highly unstable – their descriptions can change anytime - hardly discernable and difficult to predict [Gree01]. This makes the concept of context particularly difficult to deal with. The characteristics of context have been largely discussed in the community [Dour04].

In the following points the characteristics of context that are considered in this work are discussed.

- Context approximates the user's situation.

Context is not the same as situation. An individual lives and inhabits a situation at any time. Context, however, is a computational artefact to describe a situation. Therefore, it highlights some of facets of a situation. It does not intend to exhaustively describe what the user is living, but rather, focuses on the aspects that are deemed relevant.

- Context is a form of information.

A consequence of the aforementioned characteristic is that context is a form of information and as such can be captured and known. Thus, as for other information, context can be encoded in software systems. This leads to the idea that context can be expressed as a set of features and that these features can be encoded and incorporated into a software system.

- Context arises from the activity.

A key issue is to determine the facets of a situation that are important to consider, for future usages. A clue for this can be given by the user's activity. Indeed, context and activity may not be separable, as context can be shaped by the specific action performed in the moment. Unlike situation, context is not just "here", but is actively produced, maintained and enacted in the course of the activity at hand [Dour04]. Therefore, even if there is no one unique way to represent context; the activity can help to determine what facets matter and what facets are not worth considering.

- Context has different representation levels.

However, agreeing on the features' representation is challenging, even when the activity is clear. For example, considering my current location, I could say that 1) I am sitting in front of my computer; 2) I am in a building of the University of Kassel; 3) I am *in* Kassel, etc.

Generally-speaking all three possibilities are equally valid. However, they may be interpreted very differently. The question is then how the proper representation can be selected. In fact, we can overcome this issue in considering the needs of the application with which one is currently interacting. Indeed, the objective of context is to facilitate these interactions. As a consequence, the acceptable or most suitable representation of the context is the one that allows an application to properly differentiate between the user's situations in order to provide adequate functionalities to the user. These features should be determined by the application developers, since they have a better understanding for the design of the applications themselves.

- Context is stable.

This is a characteristic of context as we consider it in our work. Considering context as stable concept puts forward that the relevance of any feature representing context can be made once and for all for an application. The set of features might vary from one application to another, but they do not vary for a given application. This means that for two context-aware applications, the designer can select two distinct sets of features (some features being common to both sets) that will always characterise contexts for each of these two applications.

Context features and context model

The characteristics of context make it then possible to express context as a set of features, each of them characterising a particular aspect of the situation. Formally we define a feature as:

Definition 2.7: **Feature** from [Stah03]

A feature is a pair (F_{Name}, F_{Range}), where F_{Name} is a unique label out of some name space and F_{Range} is the set of values that can be assigned to the feature.

The value of a given feature F identified by the label F_{Name} is $f_{Name} \in F_{Range} \cup \{undefined\}$.

Consequently, the term context feature has to be understood in the remainder of this thesis as:

Definition 2.8: **Context Feature**

A context feature is a feature that completely or partially characterises a context.

A context is described by a set of context features[1] (one or several of them). It is therefore necessary within an application domain to determine the subset of features that is realistically attainable by sensors, applications and users and able to be exploited in the execution task. A context model identifies this concrete subset. We consider in this work, that it is specified by the application developer.

Formally a context model is:

Definition 2.9: **Context Model**

A context model χ specifies the exact set of context features used to characterise a context.

$\{ (F_{1,Name}, F_{1,Range}) .. (F_{n,Name}, F_{n,Range}) \}$, with $n \geq 1$

Representing and modelling context

Different data structures have been employed to communicate contextual information in systems, and represent the context features. Detailed reviews of the various approaches can be found in [HeIn+02], [StLi04]. The main data structures include the following:

- Key-value models

Key-value models represent the simplest data structure for modelling context. In such a model, a context feature's description and its corresponding value are simply stored (as key and value, respectively). Due to its simplicity, such models have been widely used.

- Markup scheme models

A markup-scheme model uses a hierarchical data structure consisting of markup tags (e.g. XML tags) with attributes and content. Such an approach extends the key-value approach, as simple relationships between the stored information (context features) are implicitly defined in the document structure.

- Ontology based models

An ontology is a data model that represents a set of concepts within a domain and the relationships between these concepts. As such ontology-based models are similar to markup –scheme models[2]. But ontologies enable the formal expression of context elements (as concepts) and consequently reasoning techniques can be applied to infer about the ontology concepts. Markup languages such as OWL [OWL], RDF [RDF] or RDFs [RDFS] can be utilised to express ontology-based context models. Various context-aware frameworks use

---

[1] In the remainder of this work, as no other features than the context features are discussed, both terms will be used interchangeably.

[2] In fact, since the languages used to express ontologies are markup-scheme language (OWL, RDF), ontology-based models can be viewed as a special case of markup-scheme models.

ontologies to model context.

- Graphical models

Context can be graphically modelled. This approach allows the representation of the context's elements (as concepts) and the relationships between them. An extension to The *Unified Modeling Language* (UML), the de-facto standard for object modelling and specification languages, has been presented in [Henr03] and referred to as ORM (*Object-Role Modeling*). .

- Object oriented models

Existing approaches use various objects to represent different context types (e.g. temperature, location, time, etc) and encapsulate the details of context processing and representation. Accessing and processing context are enabled by well-defined interfaces.

2.2.3 Context-aware infrastructures

To support users with software systems dealing with context, various infrastructures have been developed. The first generation of these infrastructures was mostly influenced by the vision of Mark Weiser [Weis91] and the work at Xerox PARC on the PARC Tab project (see e.g. [ScAd93]). In these first attempts to utilise context information in order to improve human-computer interactions, monolithic applications were developed that, as in [ScAd+94], perform all functionalities required in their scenarios to deal with context.
An application is a special class of software system. Formally, it is:

Definition 2.10: **Application**

A software system that directly performs a task on behalf of a user, based independently on implicit or explicit user inputs.

To facilitate the building of context-aware applications, and provide reusable means for handling context, [Dey00] has proposed and developed a novel approach: a context-aware computing framework. The framework serves as a toolkit for supporting the building of context-aware applications, via a set of components. Different components are responsible for capturing, transforming and aggregating raw information. It is then left to the application designer to articulate the set of interesting contextual states ahead of time (i.e. foresee the contextual states and determine what information characterises them) and programme for each contextual state what appropriate action has to be taken by the system. This approach has been widely adopted in the context-awareness community, leading to the development of numerous systems to deal with context [BaDu07].

Operations

In fact, a set of common operations can be identified among these different context-aware applications and systems when they deal with context, though they differ from the methods, techniques and technologies they implement. These operations are present in number of context-aware infrastructures, regardless of how they are implemented, i.e. if all operations are realised by a unique application or if an underlying system provide some of these operations in a generic way to support several applications.

- Operation 1: Raw data retrieval

The first operation consists of retrieving raw context data from sensors located in the environment. Note that the term “sensors” does not only refer to sensing hardware here but designates any data source that is able to produce context information. [InSu03] distinguishes between three types of sensors: physical sensors, logical sensors and virtual sensors. Physical sensors are pieces of hardware capable of capturing physical data, such as thermometers (temperature), motion detectors (motion), etc. Virtual sensors provide data from software applications. For example, a computer can provide indications on the current CPU load, keyboard inputs, etc. Logical sensors augment previous both types of sensors by combining their data, e.g. data from databases. To be able to access any type of sensor, systems have to use appropriate drivers for physical sensors, or well-defined APIs for virtual and logical sensors. The query functionality is often implemented in reusable software components which provide abstract methods (getTemperature(), getPosition()).

- Operation 2: Context interpretation

The second operation aims to transform raw context data into a meaningful context description. A context model specifies what features compose the context description.

Interpreting context consists of transforming raw sensor data into human understandable high-level contexts. High-level contexts are composed of data from different context data sources or of different context types (location, temperature, etc). For example, interpretation can be carried out by using predefined rules as in [RoHe+02], [BiCa04]. A slightly different approach is presented in [GuPu+04], where context is modelled and interpreted based on ontology. Following this approach. context is represented as predicates written in OWL [OWL]. Context interpretation is performed by a context reasoning engine that supports RDF-S [RDFS] and OWL [OWL] reasoning and general rule-based reasoning. This layer also handles the storage and management of context information.

According to the architecture upon which the system is designed, context can be accessed by different clients via a public interface (context server).

At this stage of the discussion, it is worth noting that these first two operations have been implemented in different ways. In [Chen04] three different approaches are discussed on how to perform the gathering of context information.

- Direct sensor access

In this approach, the context processing logic and the sensors are tightly coupled, i.e. the software gathers context information directly from the sensors. Thus, there is no additional layer for processing sensor data and the required drivers to access these sensors are hard-coded into the system. Such an approach implies that the set of sensors, from which context information is collected, is known at design time. Indeed, it is not possible at system runtime to add new sensors.

- Middleware infrastructure

The middleware-based approach introduces a layered architecture into context-aware systems, and poses the advantage of hiding sensing complexity. In addition to the context processing logic and the sensors layers, an additional layer (middleware) is inserted in-between to connect them and manage the exchange of data. The advantage of this approach over the direct sensor access approach is that it enables the integration and use of new sensors at runtime.

- Context server

This approach extends the middleware infrastructure approach in implementing the client-server architecture in context-aware systems. It permits multiple client access (provided that they have access rights) to remote data sources providing context information. The gathering of sensor data is moved to the context server. Various clients (context processing logic unit) access them remotely. The approach offers the advantage to relieve clients of resource intensive sensor operations. This is all the more appreciable when clients run on platforms with scarce resources, i.e. mobile devices.

- Operation 3: Action inference

Inferring the actions to be performed by the applications is the third operation. To perform inferences, additional information to the context may be requested, such as information about the user (user models), predefined action rules, or previous contexts.

- Operation 4: Action triggering

Finally, the last operation consists of the action triggering. One or several functionalities implemented by the application are triggered with regards to the action that was inferred in the previous operation. A discussion of the various forms of actions that context-aware applications can carry out is given in the next section.

2.2.4 A taxonomy of context-aware applications

A large variety of context-aware applications has been developed (see [ChKo00], [Kork00] for some examples) and demonstrates the benefits of handling context for some application scenarios. These applications are very diverse in nature. However, it is possible to determine some commonalities in their functionalities. This has led to diverse categorisations of context-aware applications.

In the first attempt to develop a taxonomy of context-aware applications, Schilit et al. in [ScAd+94] have classified applications over two dimensions: whether the task is to get information or to execute a command, and whether the task is executed manually or automatically. Following these distinctions, the authors have defined four categories of context-aware applications. 1) Proximate selection applications are applications that retrieve pieces of information, based on the available context. Ultimately, the user can take in one piece of information manually. For example, a list of items relevant in the user context is presented, but the final selection is left up to the user. 2) Their counterparts, which recall information automatically, are referred to as automatic contextual reconfiguration applications. 3) In addition, applications that pre-select commands for the user based on the context are classified as contextual command applications. They provide executable services[3] made available based on the user context, but again it is up to the user to manually select the service that will be executed. 4) Finally, applications that execute commands for the user automatically use context- triggered actions.

Brown et al. have introduced a separation between continuous and discrete context-aware applications [BrBo97]. In continuous applications, the information presented to the user changes all the time. For example, as a mobile device is equipped with a location sensor and a digital compass, an application can continuously display an arrow on the screen to show in which direction the user should go to reach a predefined destination. Alternatively in discrete applications, separate pieces of information are attached to contexts and are displayed whenever that context is entered.

Later, Pascoe in [Pasc98] has identified four core features of context-awareness. 1) The first feature is contextual sensing and designates the ability to detect contextual information and present it to the user. This is quite close to Schlit's proximate selection. 2) The second feature is contextual adaptation, which describes the ability to execute or modify an application automatically based on the current context. It is similar to Schilit's context-triggered actions. 3) The third feature includes contextual resource discovery, which characterises the ability of an application to locate and exploit resources and services that are relevant to the user's context. It then maps automatic contextual reconfiguration. 4) The fourth and last feature is called contextual augmentation, the ability to associate digital data with the user's context. For example, a user can create a virtual note, which is activated and displayed to another user in the right context.

In [DeAb00] a categorisation combining ideas from Schilit and Pascoe's taxonomies has been proposed. It lists features that context-aware applications may support. Context-aware applications can have the ability to present information and services to a user. This feature is

[3] In this section, the term service should be understood as the results of a command execution. An application or a system can then perform different types of services, which are selected based on the user's context.

called presentation. Applications can also perform automatic execution of a service. Finally, some applications may use context for tagging context to information for later retrieval. In comparison to the previous taxonomies, no difference is made between information and services, arguing that in most cases it is difficult to distinguish between presenting information and presenting services (is a list of available printers a list of information or services?). The discovery and exploitation of local resources is also not considered as a distinct feature. Locating a service according to the user's context is really not different than choosing services based on context.

Focusing on the interaction between humans and technology, the authors of [BaDe03] have defined three levels of interactivity between a mobile computing device and its user: personalisation, passive context-awareness and active context-awareness. Personalisation does not involve context. It characterises applications that let the user specify his own settings for how the application should behave in a given context. Passive context-awareness presents updated context or sensor information to the user but lets the user decide how to change the application behaviour. Finally active context-awareness autonomously changes the application behaviour according to the sensed information.

Most aforementioned categorisations highlight the type of interaction between the user and the context-aware applications. Whether it is called proximate selection or contextual commands (Schilit et al.), contextual sensing (Pascoe), presentation (Dey et al.), passive context-awareness (Barkhuus et al.), all categorisations define one or several classes where actions on context-aware applications are ultimately triggered by their users. Similarly automatic contextual reconfiguration or context-triggered actions (Schilit et al.), contextual adaptation or contextual resource discovery (Pascoe), execution (Dey et al.) and active context-awareness (Barkhuus et al.) designate applications that perform tasks automatically. In addition Pascoe introduces a new class, contextual augmentation, renamed as tagging in [DeAb00]. It characterises the class of applications that associate digital data with the user's context and ease or enable information retrieval.

While emphasising the execution of the applications' tasks does make sense from the perspective of users, it may have little impact on the development of these applications. Whether the user is presented with one service – which is automatically triggered – or with a list of the *n* best available services – among which the user performs a selection – the choice does not strongly influence the required operations for dealing with context. These operations include: collecting information, inferring a description of context from the information, deploying strategies to decide on the actions, commands or pieces of information that are relevant in this context. It seems to be more like a design decision to present a list to the users – much like recommender systems – or to have the system select the most relevant item and trigger it.

Similarly, the differences introduced in [BrBo97] between continuous and discrete context-aware applications seem to be more strongly influenced by different context states' accuracy than from resulting technological and design decisions. Indeed, it appears that both types of context-aware applications have the ability to react to context changes (technological decision). However, the context states can be either very precise (e.g. defined within a range of few meters or centimetres) or broader (e.g. defined within a city range). The more precise the context state, the more often changes occur and updated information is presented to users.

As a result of these considerations, it appears that no classification is truly satisfying from a technical perspective. This has led us to the development of a new categorisation of context-aware applications. Here, we emphasise whether capabilities of context-aware systems can be augmented with personalisation.

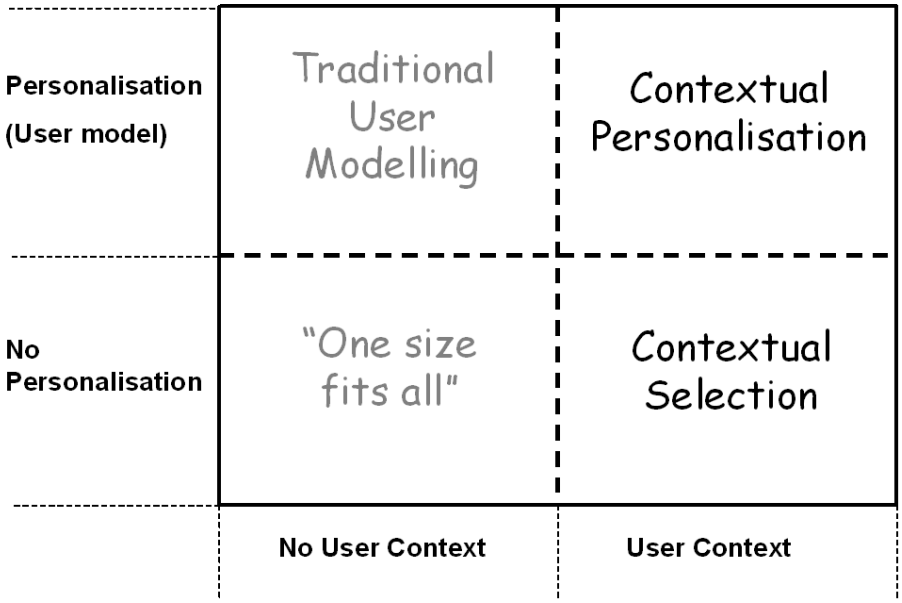


Figure 2-1: Application classification along personalisation and user context

In order to highlight the specificities of context-aware applications in comparison with other applications, Figure 2-1 depicts a classification of systems along two characteristics:

- whether the systems use context information (*User Context*) or not (*No User Context*)
- whether the systems permit personalisation (*Personalisation*) – thus relying on a user model as discussed in section 2.1 - or not (*No Personalisation*).

Today most applications do not permit personalisation, offering any user the same set of capabilities and do not consider the user's context. These applications follow the paradigm "one size fits all", i.e. one application for all users. Applications, following the traditional User Modelling paradigm discussed in Section 2.1 utilise user models to provide adapted

functionalities to their users. However, these applications do not consider user context, i.e. information about the current user's situation.

Conversely, context-aware applications use context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Considering the personalisation capability enables us here to distinguish between two sub-categories:

- Contextual Selection

Applications of this category do not permit personalisation. Rather they provide a set of operations - independently of the user needs - which they trigger when some predefined context-based events occur. Thus, they do not rely on or make use of user models.
For example, the application managing a smart room switches on the lights any time a user enters the previously empty room. The user's preferences with regards to the lighting are of little or no consequence; the task is just performed in reaction to the contextual information and in a user-independent fashion.
Alternatively, this class of applications also permits the use of context to perform some of the operations. For example in context tagging applications, context is used to facilitate the retrieving of documents. At each user's request the application performs the same operation: it compares the stored context tags with the current user's context, and retrieves some user documents. However, no user preference is utilised.

- Contextual Personalisation

This category characterises context-aware applications, which provide functionalities adapted to their users' needs. Context primarily enables to select user information from the user model to fulfil the user needs. It becomes then, similar to traditional user-adaptive systems, in that the information of the user model governs the adaptation. However, user context information can also be used during the adaptation process. For example, in the case of a restaurant recommender, the user preferences in compliance with the user's current situation can be used along with the user's current location to provide a suitable restaurant in the neighbourhood. In the remainder of this thesis we refer to these applications as contextual personalised applications. Their characteristics are discussed in more detail in section 2.3.

2.2.5 Application domains

Context-awareness is still in its infancy, as most applications remain laboratory tools today. Several application domains are, however, good candidates to develop commercial products.

- Intelligent environments (smart rooms)

Intelligent environments, also referred to as smart rooms, typically consist of a community of software agents that can coordinate and cooperate with each other to provide services to the

human users. [Chen04] (p2).

- Location-awareness

The location of a person was identified at an early stage of context-awareness research as one of the paramount sources of context information (e.g. [ScAd93]). Since then, a wide range of systems has extensively relied on location to facilitate interactions between humans and computers [ScBe+99]. Today, location based services (LBS) belong to the very few context-aware applications that have already made their way to the mobile computing market as products.

- Context-aware communication enablers

Context-aware communication designates the class of applications that apply knowledge of context to improve technology-based communication between individuals. In particular telephony has been recognised as an important application of context-awareness, as witnessed by the wide variety of activities in this field [Henr03]. In this domain, application types have been explored by the research community that address issues related to e.g. routing of calls, messaging – providing the right information at the right time, addressing – determining which people should be included in a communication [ScHi+02].

- Context-aware tour guides

Information retrieval has been widely studied in context-awareness. [BrJo01] have discussed general aspects of context-dependent information retrieval. But information retrieval has also been applied to the domain of context-aware tourist guides. Two of the most significant guide applications are GUIDE [ChDa+00] and Cyberguide [AbAt+97]. GUIDE is dedicated to Lancaster city visitors and provides them with information about nearby attractions, tailored city tours, based on their location and their preferences. Cyberguide provides similar services to visitors within single buildings and campus-wide settings.

2.3 Contextual Personalisation

Mobile communication and Internet technologies have been important research areas throughout the past decades. Products in both technologies have already significantly changed and improved our communication habits and access to information in general. Now research efforts are under way to merge these technologies and to address the scenarios of Ubiquitous Computing, where technology vanishes in the environment. Key requirements for these scenarios are autonomy and flexibility of computing systems. Indeed, as Information Technology has developed, user access to information has become increasingly greater and ubiquitous. Users are now able to access a wide variety of data relative to diverse purposes from different personal or public displays and devices. This calls for the need for filtering information and adapting services presented to the users.

- User-Adaptive systems

The traditional 'one-size-fits-all' approach in the development of applications is not appropriate to serve the fundamental objective of Ubiquitous Computing applications, namely, to communicate to users "the right thing at the right time in the right way". User adaptive systems (section 2.1) help to reduce the reliance on explicit user inputs and promote adaptation to dynamic factors in computing environment.

- Context-awareness

However, it is not enough to supply content or services that consider single user characteristics. As suggested by research work on context-awareness (section 2.2) environmental information provides valuable inputs to computer systems, as they disappear from user's consciousness. The fundamental objective behind the development of context-awareness is indeed to facilitate the user interaction with computing applications and systems, so that interactions are kept at a minimum and unobtrusive level.

As claimed in [Pope05] future communication systems are required to adapt not only to the specific demand of individuals but also to the individual's situational and environmental conditions. In fact, as any user encounters many situations everyday, context is likely to significantly vary within the course of the day, and consequently so might be the user needs towards any computing systems. Systems, which automatically address a potential wide range of varying user needs, might prove to be most useful.

In this work, we refer to the context-aware applications (see 2.2.4) that rely on user models as contextual personalised applications. In this section, we discuss more in-depth the characteristics of these applications.

Bringing personalisation and context-awareness together

In the framework of user-modelling systems, context can be seen as the reification of certain properties, describing the environment of the system and some aspects of the system itself, which are necessary to determine the need for "personalisation" [KaPr+03]. Thus Contextual Personalisation resides at the crossroads of two research areas, user modelling and context-awareness (see Figure 2-2), and the development of applications is governed by intrinsic characteristics of systems from both areas.

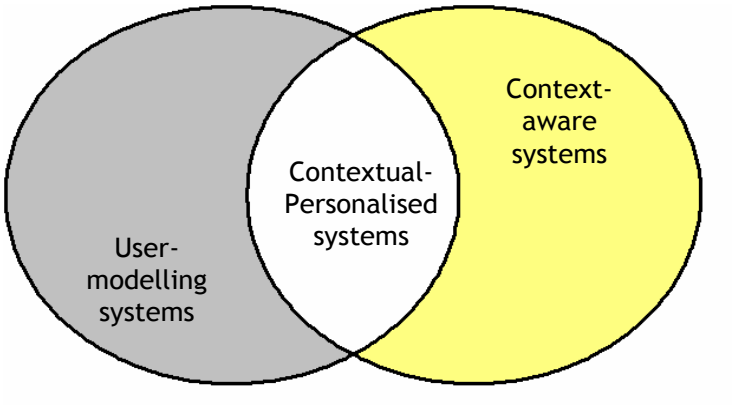


Figure 2-2 Contextual Personalisation, Context-Awareness and User Modelling

An application is a self-contained object that accepts one or more requests and returns one or more responses through a well-defined interface. We refer in the following to the set of all responses of an application as application behaviours. As discussed in [DoNi04], an application behaviour can be either *gross* or *detailed*. At design time, an application is conceived to provide one or a set of gross behaviours (playing music, enabling chat, displaying messages, etc). When it comes to adaptation, the gross behaviour of an application must remain the same and its code should not be changed. For instance, a music player must remain a music player.

Conversely, to personalise an application, i.e. have an application perform a task, as desired by a user, the detailed behaviour of an application can be modified. This can involve the passing of some parameters, when requested by a method. For example, the type of music played or the algorithm that is used to construct the play list may be adapted. Alternatively, personalisation can result from the triggering of one or several methods of the application interface.

Definition 2.11: **Application behaviour** from [DoNi04]

An application behaviour is a response to a specific request, provided by the application through a well-defined interface.

The combination of context with user modelling is the fundamental aspect of Contextual Personalisation. Taking context into consideration allows an application to adapt its capabilities to the user's current preferences. This way, it provides the user with the expected behaviour.

Thus, the major facet of adaptation in Contextual Personalisation consists of selecting from a user model the data that are relevant in the current context of use. In this work, we refer to the data which are part of the user model, as user preferences. A user preference can state a user need, goal, knowledge, interest, characteristic, etc. However, a user preference is information for carrying out personalisation, as the requested behaviour of an application is

at least partially specified by a unique or a set of these user preferences. This excludes the set of user preferences data that do not permit application adaptation, such as long-term user plans or application-not-related knowledge.

Formally, in this work we consider a user preference as:

Definition 2.12: **User preference**

A user preference, *P*, is a piece of data that governs personalisation in completely or partially specifying the selection of an application behaviour by a user, and that does not characterise the user situation.

Moreover, in the scope of this work and unlike in traditional user-adaptive systems, which consider always-true user information, user preferences are context-dependent. Consequently, context cannot be treated similarly as user preferences, since an important role of context is to trigger the selection of the user preference(s), which command adaptation. This view differs from some researchers' understanding, which treats user preferences as part of context, e.g. [Henr03], [KaPr+03].

To further underline the distinction between user preferences and context, we define context, as used in this work as follows:

Definition 2.13: **Context** adapted from [Dey00]

Context is information that characterises the situation of an entity with respect to a given application behaviour.

- Context conforms a defined context model, χ and can be expressed as a subset $\{f_1,..., f_n\}$, with $f_i \in F_{i,Range} \cup \{undefined\}$,for $1 \le i \le n$ and with $n \ge 1$
- An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves

---

We conform to the definition 2.6 for the term situation.

In contextual personalised applications, adaptation can take different forms. It can be performed in the following scenarios:

- selecting an action among a list of offered ones

An application, as in [NaKi+02], can offer different behaviours at any time. The adaptation objective is then to determine the behaviour to be triggered. Indeed, the selection, i.e. the action that has to be taken, differs for various users in a given context. For example, user A may expect the application to perform the action 1, while user B prefers action 2, when both

users are in a meeting, for example. The user's context helps to define the user's preference characterising the requested action.

- providing a set of parameters that will govern the reaction of the application;

Alternatively, the action to be perform can be evident (e.g. the application provides one single functionality) but requires information from the user's model and possibly context information. For example, in the case of a music player application that has access to a list of songs classified by genre, the system can trigger adaptation by passing as a parameter to the application the genre of the songs the user wants to listen to in his current context (e.g. he wants to listen to jazz music at night, but prefers listening to rock music in the morning before going to work).

Of course, both approaches can be performed conjointly, and adaptation can then involve 1) the selection of an operation and 2) the provision of parameters.

The discussion above aims to characterise the main traits of Contextual Personalisation. It enables users to interact with computing systems in a personalised way, thus providing flexibility of use. In the Chapter 3, the main approaches related to contextual personalisation are discussed.

Units in contextual personalised systems

As contextual personalised systems are influenced by both user-modelling and context-awareness, they inherit characteristics from the respective features of these two areas. In fact, we can determine four categories – units - of functional operations that take place in any contextual personalised system. The units of operations are depicted in Figure 2-3.

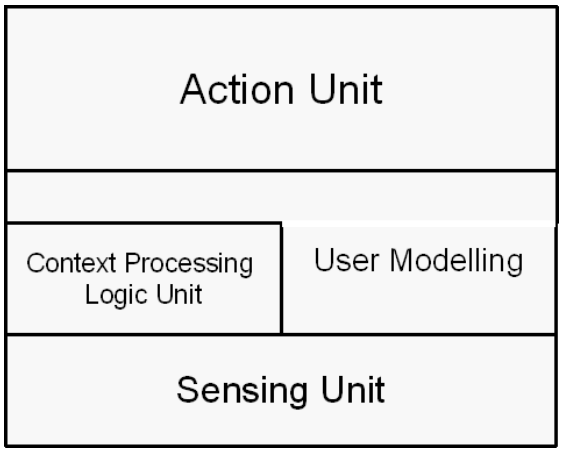


Figure 2-3: Units in contextual personalised applications

- Sensing unit

As discussed in 2.2.3, we can distinguish between three types of sensors: physical sensors, logical sensors and virtual sensors. As any context-aware systems, contextual personalised systems rely on a set of sensors that are used to capture information from the user's environment. These sensors are mostly physical sensors, but other sensors can also be

utilised. In addition, sensors are also required to gather information about the user, i.e. user preferences. Here sensors are mainly virtual sensors, providing data from software applications the user currently interacts with, or logical sensors complementing those virtual sensors.

- Context Processing Logic Unit

The context processing logic unit characterises the operations required to provide a description of the user context, as in traditional context-aware systems. Hence, we can further distinguish between the operation supporting the retrieval of sensor (raw) data, and those processing them. The result of this set of operations is a description of the user's context in compliance with a specific context model.

- User Modelling Unit

This unit encompasses the operations carried out to process sensed information into user preferences and store them into a user profile. Unlike in traditional user-adaptive systems though, the information about the user has to be complemented with some information about the user context, which characterises the relevance of the user preference in contexts.

- Action Unit

Finally, the last unit handles the operations to obtain a personalised application behaviour. Basically, obtaining a personalised application behaviour involves the retrieval of relevant user preferences in the current context, the determination of the required application behaviour, and the triggering of this application behaviour.

2.4 Case-based reasoning

Before this chapter is summarised, a last topic requires a discussion. In this thesis the realisation of a framework that supports context-aware applications, whose behaviour is personalised to the user's needs, is presented. The framework is grounded in a technique of artificial intelligence: Case-based Reasoning (CBR). Case-based reasoning is a popular technique for developing knowledge-based systems, i.e. systems containing information that a human, a formal system, or a machine can possibly use in order to perform a certain task or functionality, like solving problems [AaPl94] [Rich98].

In this section, the principles of case-based reasoning are briefly discussed. In particular, the main aspects of CBR that are of some avail to the remainder of this thesis are presented. This includes the case-based reasoning cycle and knowledge such as the symmetry measure [Rich98].

2.4.1 Principles of CBR

Case-based reasoning has roots in different disciplines: cognitive science, knowledge representation and processing, machine learning and mathematics [RiAa05]. While cognitive science focuses on the desire to model human behaviour and understand human reasoning capabilities, artificial intelligence fosters the development of techniques and methods to make computer systems more effective and autonomous by allowing them to learn and reason [Leak96].

Most traditional artificial intelligence approaches rely on the formalisation and the application of generalised knowledge about an application domain. For example, in a rule-based system, reasoning takes the form of a process that draws conclusions by chaining together generalised rules, starting from scratch. Knowledge of the domain can be expressed as rules (IF-THEN statements). The Case-based reasoning approach takes a different view.

In Case-based reasoning, knowledge does not consist of generalised rules. Rather CBR utilises the specific knowledge of experiences about episodes observed in the past. Episodes are referred to as cases and each of them typically consists of a problem description associated to a description of the corresponding solution.

The CBR approach is indeed inspired by the role of reminding in human reasoning, and relies on the core assumption that similar problems have similar solutions. As a consequence, solutions remembered for similar prior problems are a useful starting point for solving new problems. Case knowledge can then be used to solve a new but still unsolved problem. A new problem is compared to the problem descriptions of the previously available cases in the case base. Typically, the case with the most similar problem description is then selected and can be reused in the new problem situation (see Figure 2-4).

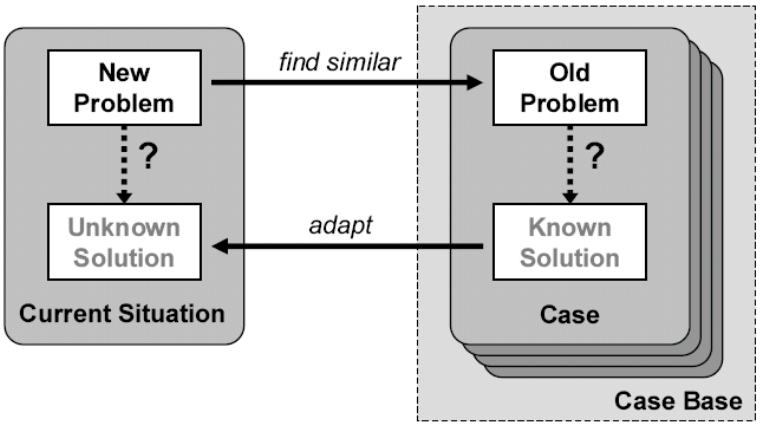


Figure 2-4: Case-based problem solving from [Stah03]

Reasoning in CBR is initiated as a new problem (new case) is given for which a solution is required. The new case can be characterised as $\{P_N, S_N\}$, where P_N is a description of the current problem and S_N is its associated still unknown solution part. It is usually unlikely that

the case base already contains a problem description that exactly matches the new problem. Then a critical task is to retrieve a case, the problem description of which is similar to the new problem. It requires understanding the problem to infer the problem descriptors, i.e. the features that are relevant to characterise any problem.

Using situation-specific knowledge in comparison with generalised knowledge (e.g. static rule libraries) offers three potential functional benefits [Leak96]

- It increases problem-solving efficiency.

By storing prior solutions rather than redefining knowledge from scratch, as in the generalisation process, the process efficiency can be increased over time. Indeed, additional cases are stored and become immediately accessible and available for the problem solving process.

- It handles incomplete domain knowledge.

Traditional rule-based and model-based approaches require a complete and a correct domain theory. However in practice, domains can be poorly understood so that no such domain theory is available. Consequently, employing generalised knowledge for solving problems can be constrained and can lead to unsatisfactory results, if one does not possess a complete and consistent model of the domain theory. On the other hand, reasoning based on experiences can provide reasonable solution proposals.

- It simplifies the acquisition of knowledge.

Unlike rule acquisition, which requires analysing the interactions between all individual factors in a situation, in case-based reasoning an entire episode can be treated as a unit from which to reason. CBR is indeed an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems.

2.4.2 The Case-based reasoning Cycle

In order to perform problem-solving, case-based reasoning methods have to deal with a set of central tasks, at a high level of generality. Thus, any CBR process is typically represented by a schematic model introduced in [AaPl94] referred to as case-based reasoning cycle. The cycle is structured around four consecutive steps, which are described below (Figure 2-5).

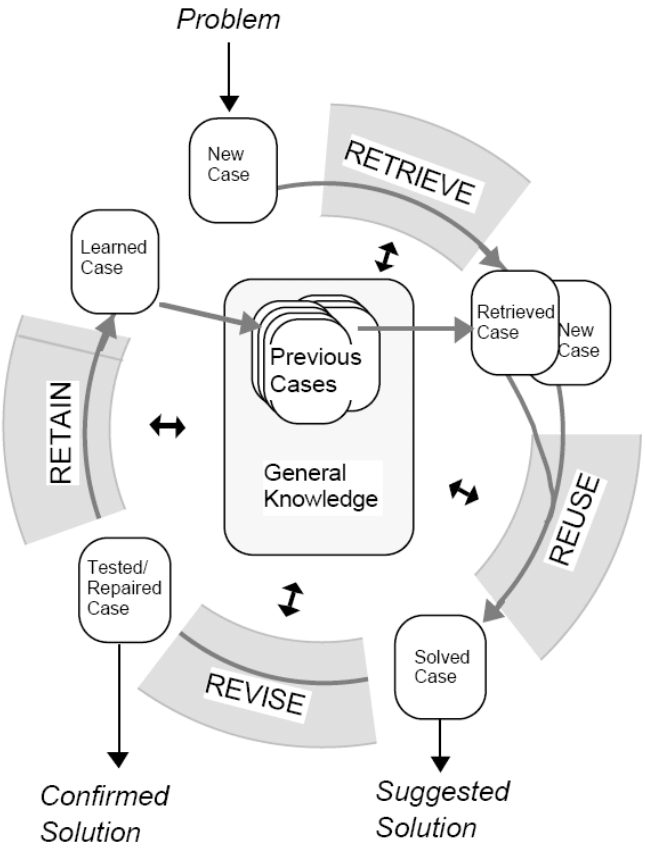


Figure 2-5: The CBR-cycle (from [AaPl94])

- Retrieve

The first phase consists of retrieving one or several cases from the case base that is or are considered to be useful and significant for solving the current problem. Therefore the retrieval phase involves the assessment of similarity. The task of the retrieval phase is to select cases with problem descriptions similar to the current problem's description. Assessing similarity between two problem descriptions is performed by means of a similarity metric. The phase ends with the selection of the most or the n-most similar cases from the case base.

- Reuse

In the reuse phase, the retrieved case solution(s) are applied to solve the new problem. For some domains, the differences between the new and the retrieved cases can be determined. Indeed, the differences may make it impossible to reuse a retrieved solution. By analysing these differences, a part of a retrieved solution(s) can be adapted to ensure it best matches the new problem situation. However, in other domains, even if the retrieved case does not exactly match the new case, its solution can be reused without any adaptation being performed. Indeed, the differences between the retrieved and the new cases are considered irrelevant for the new solution. No adaptation is thus performed, as the (unique) solution of the retrieved case(s) is then transferred (copied) to the new case as its solution.

Whether adaptation of the retrieved case solution is needed and how it is exactly performed depends on the application domain and scenarios. (The approach developed in this work for

contextual personalised application is described in Section 4.3.6) Whether the retrieved case solution is adapted or not, the reuse phase terminates when a solution for the new problem description is determined.

- Revise

However, no certainty can be given that the proposed solution adequately addresses the new problem situation. In fact, the case solution has to be evaluated and if the reuse phase generated an inadequate solution, the case has to be repaired. Case repairing consists of detecting the errors of the current solution and retrieving or generating explanations for them. Ultimately case repairing aims to perform new attempts to generate valid solutions. Thus, other adaptation alternatives, if they exist, can be carried out. Also other retrieved cases can be adapted or even new retrieval phase can be executed that will hopefully result in more useful cases.

- Retain

When the proposed solution is assessed as correct, after the revise phase, the new case composed of the problem description and the repaired solution, is retained in the case base for future usage. This phase involves selecting which information from the case to retain, in what form to retain it, how to index the case for easier retrieval later for similar problems, and how to integrate the new case in the case base. Also a failure solution, i.e. solution assessed as incorrect in the revise phase can be retained. When a failure is further encountered, the system can then receive a reminder of the previous failure case and improve its understanding of the present failure (e.g. how the failure was repaired) [AaPl94, Spal01].

2.4.3 Knowledge containers

The central notion of intelligent problem-solving is that a system must construct its solution selectively and efficiently from a space of alternatives. An expert's knowledge helps to spot useful data early and suggest promising ways to exploit them.

In CBR, different sources of knowledge are used for problem-solving. They are stored in knowledge containers [Rich98]. [Rich98] identifies several containers: Vocabulary, Case knowledge, Adaptation knowledge, Similarity measure.

- Vocabulary

Vocabulary defines which information is considered to be important in the respective domain and how the information can be expressed. Usually the decision can only be made by a domain expert based on his expertise.

- Case base

The case base contains the set of domain experiences, which are reused to propose a solution to the problem. The knowledge experience characterises the formalism that is used to represent the cases. Traditionally, CBR systems represent cases in two parts: *Problem*

part, *Solution part*, but additional information might also be appropriate for specific domains [Stah03]. In addition, different formalisms can be used to represent cases, including attribute-value based representations, object-oriented representations, first order logic based representation (rules) and graph representation.

- Adaptation knowledge

Adaptation knowledge is used to control the reuse of retrieved cases. It defines the operations performed on the retrieved cases to come up with a solution for the new problem situation.

- Similarity metric

In order to select previous cases, the problem part of which will be reused in a particular problem situation, CBR systems employ similarity metrics. The objective of such a metric is to estimate the utility of cases with respect to the current problem task [BeRi+01].

The problem solving task is strongly influenced by the general knowledge contained in the vocabulary, the similarity measure, and the adaptation knowledge. A fundamental problem for any CBR system is to define adequate general knowledge. Traditional CBR systems rely on domain expert-defined general knowledge. However, this has limitations, since it relies on the unique expert knowledge and understanding of the domain. When the domain is quite simple and well known, knowledge can be fully characterised. But in domains which are not so well understood, the task can be more difficult.

2.4.4 Similarity metrics

Similarity is an important concept in case-based reasoning, but does not limit to this field only. It also plays a fundamental role in theories of knowledge and behaviours [Tver04].

The general problem addressed by the CBR system has an underlying utility function (a function that gives solution to the problem). Generally, the utility function is totally or partially unknown. The utility of cases cannot be determined until the problem solving is finished. That is, we do not know a priori what case is the proper one to address the problem. Therefore, in order to be able to approximate the utility of cases, CBR systems rely on similarity knowledge. This knowledge is encoded in similarity metrics. Similarity metrics are used to compute the similarity between a new case and previous cases of the case base. In the CBR process, the first phase consists of retrieving useful cases, i.e. cases whose solutions can be reused for the problem situation.

Definition 2.14: **Similarity metric**

A similarity metric is a function $Sim : D_D x D_D \rightarrow [0,1]$, which assigns a level of similarity between any two elements defined in the domain D_D

A similarity metric is hence defined from a domain D_D to a continuous interval [0, 1]. The value given by the similarity metric indicates the degree or level of similarity. The greater the value is the more similar the elements are.

Besides, for any $(x, y) \in D_D x D_D$, a similarity metric has the following properties:

- Reflexivity: $Sim(x, x) = 1$
- Symmetry: $Sim(x, y) = Sim(y, x)$
- Triangle inequality: $Sim(x, z) \leq Sim(x, y) + Sim(y, z)$

To retrieve cases in CBR systems, the similarity measure has to be expressed in such a way that it can be processed by computers. The representation of similarity measures strongly depends on 1) the vocabulary for the CBR system and 2) the case representation.

In many systems the case representation and vocabulary are simple enough so that simple similarity measures can be employed (for example, hamming distance, city block, Euclidian distance, etc).

However, when case representation is more complex, consisting of features with more different value types, such simple similarity measures are inappropriate. Instead, [Stah03] suggests using the *Local global principle*. According to this principle, the entire similarity computation can be decomposed into 1) a local part, only considering local similarities between the single features' values and 2) a global part, computing the global similarity for whole cases based on the local similarity assessment.

2.4.5 CBR application fields

- Product recommendation and information retrieval

With the increasing success of e-Commerce Websites, the development of always more efficient recommender systems has become a popular area of research. As Case-based reasoning provides powerful techniques for realising similarity-based retrieval, it seems an appropriate technique for building such systems [BeTr+02]. However, the application of case-based reasoning is not limited to recommender systems and other information retrieval systems have been developed [LoCu+05].

- Planning

Planning consists of constructing a course of actions to achieve a specified set of goals, when starting from an initial situation. Planning is used to automate a variety of tasks including robotic control, process planning, transportation planning, and experiment planning. Case-based reasoning provides an attractive paradigm for planning as it can improve the effectiveness of the planner by applying old experiences to solving new planning problems. A review of the techniques used in case-based planning is given in [Spal01].

- Knowledge management – decision support

Case-based reasoning has been successfully used in the area of decision support, in

particular in intelligent call centres and customer services. The role of decision support is to help the decision maker in providing reusable and relevant knowledge for business processes of the organisation or the company. For example, this knowledge is used in Customer services by a customer agent to quickly and flexibly react to the client's requests. [HeAb06] is an example of such a system, but similarly case-based reasoning can also be applied to a slightly different scope, such as in [GöTh+06], where a system enables employees to solve business problems by helping them to obtain answers to their questions from knowledgeable colleagues. Generally, the output of a support system is not a solution, as opposed to recommender systems, but an advice or a useful piece of information.

2.5 Conclusion

This chapter discusses the fundamentals of two types of systems that form the basis for this thesis's work: user-adaptive systems and context-aware systems.

On the one hand, user-adaptive systems aim to accommodate the differing requirements of individuals or groups of users and their changing needs over time. They rely on a collection of implicitly acquired user data, e.g. user needs, goals, knowledge, interests or other characteristics, providing a model of the user about what is deemed relevant when interacting with the system. The system is thus said to be personalised. Personalisation is the task of providing adapted capabilities to the users of a system, on the basis of this implicitly gathered information. It is performed in two stages: Firstly, user information is acquired and knowledge about the user is inferred. This is carried through in recording user interactions with the system and using the observed sample results to make a statement about the user's future behaviours and preferences. Secondly, analytical techniques are applied to user data to make decisions and improve selected aspects of the interactions between the user and the system.

On the other hand, context-aware systems examine and react to a user's changing context. They help to promote and mediate people's interaction with systems, as well as with each other. Here, context designates aspects of the user's situation, encompassing various facets, such as his activity, his location, and time, to name a few. The characteristics of context make it possible to express a set of features, each feature characterising a particular aspect of the situation. A context model identifies a concrete subset of what data can realistically be collected by sensors and able to be exploited in the execution task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer.

In the course of the research on context-aware systems, a large variety of prototypical context-aware applications have been developed. We propose a classification for these applications, based on the use of context and the methods used to select the items presented to the users or triggered. We distinguish between the following:

- Contextual Personalisation

This class characterises applications which provide functionalities adapted to their users' needs. Here, context enables selection among the set of user model information. It is then, as in traditional user-adaptive systems, the information of the user model that governs the adaptation. It is this class of applications - we refer to as contextual personalised applications – on which we focus in the remainder of this thesis.

- Contextual Selection

The counterpart of contextual personalisation is contextual selection. Applications of this class provide a set of operations that does not utilise any user preference.

Personalising an application can be viewed as selecting one or several application behaviours. To have an application perform a task, as desired and expected by a user, it simply suffices to call one or several methods of the application interface. Then Contextual Personalisation results from the combination of context with user modelling. It consists of selecting from this user model the data that are relevant in the current context of use. Indeed, these preferences can be context-dependent, i.e. their relevance for adaptation can be influenced by the user's situation very much.

This chapter ends with a description of Case-based reasoning. The framework for contextual personalised applications we present in Chapter 4 is based on this technique. The CBR approach relies on the core assumption that similar problems have similar solutions. Solutions remembered for prior problems are a useful starting point for solving new problems. Similar case knowledge can then be used to solve a new but still unsolved problem. To do so, a new problem is compared to the problem descriptions of the previously available cases in the case base. The case with the most similar problem description is then selected and can be reused in the new problem situation.

3 Requirements and related work

As the path toward the realisation of Ubiquitous Computing has progressed, context-awareness has been identified as a critical issue. Thus, numerous system infrastructures and prototype applications have been developed (e.g. [ChKo00], [Kork00], [BaDu07]) to demonstrate the benefits context-awareness can bring to modern communication technologies. Much research has already been undertaken and is still going on in this field, but this thesis does not intend to give a complete description of it. Rather, this chapter examines existing approaches dealing with issues specifically related to Contextual Personalisation.

This chapter is organised as follows: firstly, the fundamental challenges of contextual personalised systems are discussed, secondly, selected approaches that are of any relevance to our own work with respect to these challenges are described and evaluated; finally, the challenges in the different approaches and their shortcomings are summarized.

3.1 Contextual Personalisation challenges

As discussed in section 2.2.4, contextual personalised systems are at the crossroads of both user-modelling systems and context-aware systems. Context was already discussed in section 2.2.2. In user-modelling, the aim is to capture user information such as preferences, beliefs, goals and intentions and to construct user models out of them. When personalising context-aware systems, user models are also utilised to determine the user requirements with regards to systems and to ultimately decide what actions are to be taken. However, for this class of systems, user preferences are context-dependent and their relevance has to be evaluated against the context.

This section presents the requirements we identified for the contextual personalised systems, and the stages any such system has to take to provide Contextual Personalisation. Both the requirements and the stages highlight the challenges that the systems have to address.

3.1.1 Requirements

Contextual Personalisation enables systems to be flexible, adaptable and capable of acting autonomously on behalf of the users. It intends to improve systems' usability and to provide more intelligent services. From a user perspective, any contextual personalised system is required to supply the following fundamental capabilities:

- Placing minimal demands on user attention

The first characteristic of such systems is that they must present an advantage in terms of usability in comparison to the traditional, explicitly-governed ones. User inputs should not be banned but must be kept to a minimum to facilitate the user interactions. However, enabling

autonomy must keep systems from falling into the “dark side of computing”, in imposing behaviours on users, thus acting against their will or in compromising their privacy.

- Performing at any time and under any circumstances

Besides, the right task must be carried out, irrespective of the user’s situation and at any time. To the user, it is not acceptable that the system does not perform correctly and does not react to some of his solicitations.

- Performing according to the user needs

An application must always perform as it is expected by a user. This is, in fact, quite a generic user requirement, though a crucial one. No user will ever utilise systems that provide results and tasks that were neither required nor needed. Here, the importance of this aspect actually results from the fact that users do not explicitly govern contextual personalised systems. Flaws in the adaptation process can cause the system to “misbehave”, i.e. the system presents users with an inappropriate behaviour.

- Privacy of user data

Privacy is a major user concern with interactive systems [Kobs07]. Users find it important to know how their personal information is being used and to have control over its usage, in order to deliver private data to parties they trust only. Contextual personalised systems are all the more affected by this concern in that they deal with two types of information, i.e. user preferences and user context. Systems that do not address this paramount requirement would surely suffer a loss of credibility among their users and eventually dissuade users from using them.

3.1.2 Contextual Personalisation stages

To provide a personalised behaviour to its user, any contextual personalised system has to go through a set of stages. As systems are to reduce the burden on users, the following stages have to be taken in such an autonomous way that they require little user participation.

1. Determining the current user context

The first stage consists of characterising the user situation. Based on sensor information and driven by a context model that specifies the relevant types of information, a context description is assembled.

2. Determining the set of user preferences that are relevant in this current context

In the second stage, user preferences are filtered. Only those that are relevant in the current context are selected. It implies that for each user preference, its *relevant context(s)* is known.

3. Deciding what application behaviour to perform

Based on the set of retrieved user preferences, the proper action is inferred via diverse strategies and mechanisms and eventually performed on behalf of the user.

3.1.3 Challenges

Contextual personalised systems have not yet made the transition from the laboratory to everyday use. This is surely a result of the limitations of the current realisations. In fact, the aforementioned three stages, as well as the user requirements, introduce some challenges that existing realisations have not addressed successfully yet, as discussed in the later part of this chapter. These challenges are detailed below:

- Collecting user preferences and defining their relevant contexts

Actions to be performed by systems are inferred from a set of context-dependent user preferences. Thus, the first challenge that is raised by contextual personalised systems - inherited from user-adaptive systems - includes the gathering of user preferences. However, the challenge here goes beyond this. Indeed, user preferences have to be evaluated against the current context, in stage 2, in order to determine which one(s) will drive stage 3. As a consequence, not only is it required to collect these preferences but it is also mandatory to know in what context(s) each preference is relevant, i.e. in what context(s) the preference can be utilised to govern the adaptation.

The challenge is important, because if no or too few user preferences and their relevant context(s) are available, the system will hardly make a correct prediction regarding the actions to take. When treating this issue, the system must, however, address the user requirements and perform with minimal demand on user attention, but under any circumstance and in an appropriate way.

- Determining contextual states as canonical forms of contexts

Context-aware systems rely on their own internal context models to represent and handle context. These models indicate the context features that are required to fully characterise the user situation in the covered domain. Context appears to be a highly dynamic construct, as many contexts are highly unstable, hardly discernable and difficult to predict [Gree01]. Conversely, in the desktop computing environment, Ubiquitous Computing promotes scenarios in which users are not limited to a specific location but can access computing systems from a variety of devices. As a consequence, the contexts of the user can vary. It is then very likely that no two contexts will ever happen to be exactly identical. Thus, this makes it difficult to select user preferences in stage 2, as a relevant context is evaluated against the current context.

It then becomes critically important to know how existing systems deal with this issue: how contexts are confronted and how different, though comparable, contexts can be identified as such. This system capability depends very much on the way context is inferred and recognised, i.e. how one obtains a context description. In particular, the ability of systems to group contexts together referring to the same situation must be investigated. Such groups of contexts have been coined in [Gree01] as canonical contextual state.

- Protecting user personal data

Privacy of personal data is a major user requirement in contextual personalised systems. When examining existing solutions for Contextual Personalisation, their ability to protect a user's private information must be considered and discussed.

In this chapter, we review how relevant systems have addressed the aforementioned challenges. It is worth noting that the first two challenges are not orthogonal. That is, techniques have been deployed in some systems that treat them simultaneously.

- Firstly, we discuss the most relevant realisations of contextual personalised systems and underscore the techniques to acquire user preferences and their relevant context(s). When needed, commonality between contexts is assessed.
- Secondly, we provide an analysis of the techniques to recognise context. In particular, we study if and how contextual states can be defined.
- Finally, approaches for enabling privacy are examined. However, as it has been very superficially addressed so far in context-aware systems, we also extend our review to privacy-enabling techniques in traditional user-adaptive systems.

3.2 Contextual personalised systems

3.2.1 System review

User preferences and contexts are utilised to perform personalisation of context-aware systems. They give indications about how users expect applications to react to contexts. In this section, we explore how existing realisations address the issues of both gathering and representing user preferences, to enable personalisation to be performed in any user context.

In section 2.1.2, we presented the various approaches to user modelling to identify the concepts most relevant for making decisions in user-adaptive systems. In a nutshell, adaptation knowledge can be acquired in the following ways:

- Explicitly

Knowledge is captured through techniques similar to user interviews. Users are asked to input settings, via user interfaces or configuration files, specifying how applications must react.

- Implicitly

Knowledge capture relies on the recording of user behaviours and uses the observed sample results to make a statement about the user's future behaviours, preferences, etc. One distinguishes between the following:

- Content-based acquisition knowledge

  The user information is derived from historical data using machine learning techniques.

- Collaborative-based acquisition knowledge

  The user information is predicted from the past behaviours of other like-minded people. The approach is used when similar behaviour patterns are found in other users.

- Mixed approach

Finally, the mixed approach lies between the explicit and the implicit approaches. Similar to the implicit approaches, it involves inferring past user behaviours to make statements about new user information. However, it requires user involvement.

Therefore, it comes as no surprise that the contextual personalised systems implement similar approaches. In the following section, we discuss rule-based systems that rely on explicitly acquired knowledge, as well as a set of other approaches (including CBR) that capture knowledge from the user's previous interactions. We also consider some research activities that have investigated the use of context-based collaborative filtering.

Rule-based systems

As context-awareness has attracted attention, many context-aware systems have been developed to provide a middleware layer to support applications based on context. These systems aim to provide techniques to simultaneously address various issues in the context-awareness area, including the gathering of sensor information, the interpretation and the management of contexts [BaDu06]. As such, they have also provided solutions to support the adaptation of applications. Most of these systems have adopted similar rule-based techniques to this end.

Probably the simplest method to adaptation consists of hard-coding in the systems actions to be performed in contexts, as in [HoSc03].

In [GuPu+04] actions are triggered by First Order Logic rules whenever the current context changes. Application developers write these rules and specify what method is invoked when a condition (i.e. context) becomes true. All the rules are saved in a file and pre-loaded into a so-called "context reasoner component." Rules enable personalisation since, in the condition part, it is specified to what the rule applies, e.g. as in (*John, socam:MyCar*), thus making it possible to define different actions for different users in the same context. However, users cannot write the rules themselves. First Order Logic rules are also used in a similar way in [RaCa03]. As opposed to [GuPu+04], the authors envision here the possibility of adding new rules, changing existing rules and deleting rules, via a GUI. In addition, potential conflicts between actions – as two incompatible actions can be taken in one context – are resolved by means of a priority-based mechanism.

In [KaRe+02], a “customisation” (i.e. personalisation) model is presented that enables the context-based adaptation of web applications. Personalisation is triggered as soon as a context change occurs. To specify personalisation, Event-Condition-Action (ECA) rules are used. A rule consists of an *event part* and a *condition part* that together specify the context to which the rule is applied. The event part determines the change that triggers the rule (e.g. change of bandwidth). The condition part is evaluated as soon as the rule is triggered by an event. It determines whether an adaptation is required. For example, a condition could be when the bandwidth falls below a certain minimum. Eventually, the action part activates the adaptation of the application. A very similar approach is carried out in [BiCa04], where an inference engine component based on CLIPS, which is able to select rules to fire, is used. Rules here are expressed in an ECA model as well. Typically, the complexity of a rule-based inference is *O(n)*, where *n* is the number of rules in the system. The system, however, reduces the computing complexity by introducing context hierarchy. This allows a subset of rules to be pre-selected and inferred.

[FaCl04] expresses rules in database tables. Records contain context features’ values and a corresponding action. In [HeIn05] a software infrastructure is discussed that performs management and storage of contexts and preferences also using relational databases. Each user preference takes the form of a *scope* and a *scoring expression*. The scope specifies the context and choices to which the preference applies (following a specific context model described in [Henr03]). The scoring expression produces a rating that indicates the suitability of the actions that match the scope within the given context. The rating takes the form of either a numerical value in the range [0, 1], where an increasing score represents increasing desirability or a string value (prohibit, oblige, undefined, etc). Preferences may be either simple preferences, which express atomic user requirements, or composite preferences, which specify how other preferences are combined to produce appropriate aggregate ratings. In the decision-making process, user preferences are evaluated against a set of context information, candidate actions and application state variables to yield an assignment of ratings to the candidate actions. The application selects zero or more of the candidate actions and carries them out. This model is used internally to manage preferences. However, in order for users to govern personalisation, user interfaces are deployed, on a case-by-case basis, i.e. the design of an interface is specific to each application. The formulation of the user preferences is, however, kept relatively complex in terms of manipulations, as the user is asked to input information via a set of radio buttons, check boxes, drop-down lists, text fields, etc.

A more intuitive and probably user-friendly approach is detailed in [TrHu04]. This approach radically differs from the traditional way to develop applications. Rather than having applications developed by software engineers with hooks for personalisation by users (via rules), it places the task of constructing applications and then specifying the requirements into the hands of users. An interface is presented that helps to bridge the “needs-technology

gap" by offering interactions that are technically simpler. The interface relies on the "magnetic poetry" metaphor, named after the decorative set consisting of small magnets, each with a word printed on it, allowing users to combine them to form "poems" or statements. Using such an interface, any user is able to express statements about his needs. For example, a user, "Billy", can express the following statements: *"when Jim Jane and Billy talk, record and remember for 20 minute"*. In the same vein, [DeHa+04] enables users to train a system to carry out desired actions by manually demonstrating the actions, when the appropriate context occurs.

Case-based systems

Case-based reasoning for context-aware systems is a topic of increased research. Indeed, the context representation and reasoning problem presents research challenges to which knowledge management through case-based reasoning methods and techniques is now being brought to bear.

[MaKi05] proposes a case-based reasoning approach to provide a set of automated services in smart homes. It supports scenarios depicting a smart home environment, where a set of actions (e.g. air conditioning and lighting on/off, TV channel selection) are taken. Context is modelled by a large set of features such as time, time sequences, room, status (leaving, staying, or entering), room temperature, ID, as well as user's habits. A new case is represented as a set of features' values corresponding to the current context. In order to retrieve similar previous cases, a two-stage similarity metric is defined. First, similarities between the same features of the two cases are computed. Then the global similarity is obtained by the weighted sum of these features' similarities. The adaptation is governed by the solutions of the best matching cases. To cope with the variety of application domains in the smart home scenario, several strategies to adaptation are applied. For example, to select a TV channel, one single case is retrieved as the best match. To set air conditioning, k cases are selected. When one single case is selected as best match, the adaptation is straightforward: the case solution is applied. However, when several cases are selected, adaptation results from a combination of the solutions. For example, to set the air conditioning in a room, the ten best cases are retrieved and the temperature is averaged over their solution values.

[MiKo04] and [KoAa06] outline a case-based reasoning system to automatic situation assessment in a mobile environment. The work postulates that there is a goal or task in any situation. Identifying the current situation, it is possible to provide personalised services to the user. When a description of a current situation is obtained, a case-based reasoning agent identifies it by determining what user goal has to be achieved. To do so, it retrieves a previous case and assigns its goal to the new case. This goal is further presented to a task decomposition agent, which defines the tasks and subtasks to perform. Finally, the new case

is stored in the case base in a triplet consisting of 1) the context characterising the situation 2) the goal associated with the situation and 3) the task and subtasks to perform to achieve this goal. For example, when the user is in such a situation that he is hungry, the task to perform is “get food”, which includes “find restaurants” and “match restaurants to preferences” subtasks, to fulfil the “user no longer hungry” goal. In this approach, a context description is decomposed into five main categories. Besides environmental context or spatio-temporal context information, it includes a personal context class, which characterises the user’s mood, expertise but also his preferences, e.g. Italian food.

[SaGa+05] reports on the use of case-based reasoning to learn the user preferences in a context-sensitive message filtering application. Users may not want to see messages sent to them right away. The application enables users to specify preferences concerning when he wants to receive different types of messages. In this work, a case consists of context information – the type of message (meeting, class announcement, food specials, etc), the sender of the message, the user’s current calendar activity, the user’s location, and the weather - as well as the related preference. When a new message arrives, prior cases are retrieved and their problem parts are matched against the current context description. The solution part of the most similar case (nearest neighbour) is adopted. Experiments were conducted. The reported results showed a significant improvement in the quality of the filtering decisions over predefined (rule-based) user preferences with regards to the feedback indicated by a user after each incoming message. This suggests that filtering preferences may be too complex to be correctly specified upfront.

Other approaches

In [Flan05a] a learning approach is used to find similar requirements of a user in a given context (context recognition is discussed in section 3.3.1, see [HiFl+03], [FlHi+03] and [Flan05a]). When the user interacts with an application, i.e. requests an application behaviour or accesses a Web page, the learning algorithm associates the action with the current context, without requesting any additional labelling from the user. A weight ([0, 1]) is associated with each action, representing the probability that the user requests this action in the given context. Several actions can be performed by the applications in the context. Thus, as the action is selected its weight is increased, whereas if the action is not chosen it is decreased. This leads to the possibility of an adaptive interface that uses shortcut menus to different applications. Hence, when the user is in a recognised context, the shortcut menu presented to the user is determined by the weights, and the order of the applications in the shortcut menu is determined by the magnitude of the weights.

In [NuSa+06] a modular system is presented that aims to serve a diverse set of applications, providing recommendations (modality, service category, information). The modular approach enables the system to not only rely on a specific user modelling technique, but to also accept

different techniques to be performed, possibly to address the different needs of various domains. The algorithms are independent of the application and entity for which the recommendations are made. However, only two techniques have been described: rule-based reasoning and augmented tree naïve Bayesian networks. The system encompasses different components to perform these recommendations. A usage record provider stores information on the behaviour of a user and the context in which the behaviour takes place. A recommender carries out all tasks related to user modelling, i.e. updating user models and making inferences with the user models. Finally, a profile management component manages the user models. The component provides support for context-aware applications by storing, maintaining and providing applications with context-dependent user information [CoLa+06]. This component is based on views, a concept of relational databases [ElNa00]. A view is a single virtual table that is derived from other tables. It is a way of specifying a table that we need to access frequently, even though it may not exist physically. In this sense, a view does not contain data but rather points to some of those stored in an underlying database. In the profile management system, the concept is extended and views are made context-based. A view refers to a set of application specific user preferences, i.e. preferences that have been gathered while the user was interacting with a particular application. A view is labelled by two types of so-called qualifiers: an application qualifier, i.e. a reference to the application, and a context qualifier, which refers to some contextual information that characterises the context for which the view information is relevant. The advantage of this approach is that it permits user preferences to be stored only once in a traditional database, while being referred to by various applications and for various contexts. This avoids data redundancy. The selection of context-dependent user preferences is enabled by the view. When an application requires a set of user preferences (e.g. type of music), views' qualifiers are searched and evaluated against the application name and a current context description. When a view is found and when it contains the demanded user preference, the preference is delivered to the application. If no view could be retrieved or if no preference corresponds to the application query, information from a set of default views, predefined by the user is delivered.

Collaborative filtering approaches

Collaborative Filtering (CF) is also a popular approach to realise personalisation. The goal of collaborative filtering algorithms is to predict the rating of an item for a particular user, based on how previous similar users rated the same item. When applied to context-awareness, the approach can be used to determine a user preference in the current context.

[Chen05] presents a context-aware recommendation system. Traditional recommender systems based on collaborative filtering use past experiences of like-minded users only, based on the assumption that people who like the same items are likely to feel similarly towards other items. The author applies this assumption here to leverage context

information: a user preference is not only predicted from opinions of similar users, but also from feedback of other users in a context similar to the user's current context. Thus, recommendations are based on similarities between users and similarities between contexts alike. The system performs recommendation in different stages. As in any traditional collaborative filtering process, the first stage consists of measuring the similarity between users. This is done by calculating the weight of the user against every other user with respect to the similarity in their ratings given the same items. In the context-aware environment, a user preference towards an item may change with the context. To capture the different preferences towards an item in different contexts, a second stage is performed that deals with the computation of similarity between contexts. The goal is then to determine which user ratings are more relevant for the current context. Context is modelled as a set of relevant features, where relevancy depends on the application scenario. Context similarity is obtained by assessing how two contexts are related according to the ratings on items users gave in these contexts. The assumption is that, if user preferences towards an item do not differ much in different contexts, then the ratings given in one context would also apply for the other. So if ratings for an item are similar for two different contexts, then these contexts are very similar to one another. Ultimately, the computations of the first two stages are combined. The rating of a user on a new item in a given context is computed from the ratings given by other users with respect to the relevance of the rating's context to the current context.

Similarly, the approach in [SiKa05] features a context-aware content delivery system, the aim of which is to ease user navigation in a Japanese book market and provide recommendation on shops to visit. It relies on user profiles expressed as data vectors. For each user, the user profile depicts his book interests (comic book, fashion magazines), culinary likings (Japanese food, Italian coffee, etc) and context (time, location, environment-related information and status). As well, the filtering algorithm exploits the user ratings on previously visited shops in the market, which are gathered either explicitly (user assigns a mark to the shop) or implicitly from user actions (purchase, reading time of book, etc). To predict the user's potential rating for an unvisited shop and possibly recommend him to the shop, the similarity of the user with other users who have rated the shop is evaluated. This evaluation is based on the user's profile vectors. The ratings of the *k* most similar users are selected to estimate what the user rating would be like.

3.2.2 Discussion

The systems presented in this section show the variety of approaches to adapt applications to user needs in contexts. Each approach has, however, its shortcomings, which limit its applicability to a wide range of contextual personalised applications.

Most context-aware systems rely on manually defined rules to determine the application

behaviour in different contexts. These rules can be predefined by the application developers, [KaRe+02], [BiCa04], [GuPu+04], or, alternatively, user-configured, [RaCa03], [HeIn05]. Static rules are inflexible and difficult to personalise for individuals. Even though it may be easy to characterise and describe some of the user's situations, for which a specific application behaviour can be requested, it is arduous to exhaustively define all user's contexts where the user expects an application behaviour. These systems are also unable to predict a user preference in an unforeseen situation. As a consequence, in contexts where the rules no longer apply, no action can be carried out until the current rules are modified. Rules can be newly input by the user. But this involves additional (cognitive) load on the user.

Leaving the user programming applications can, at first, appear to be a promising approach. However, as remarked by [HeIn05], user-programming techniques are suitable only when the application behaviours that the user needs to specify are reasonably simple. This idea is supported by the fact that the scenario presented in [DeHa+04] concentrates on simple tasks, such as loading presentation files in advance of a meeting.

Case-based reasoning approaches seem to overcome these limitations. Neither the system developers nor the users are requested to explicitly input any information. Rather, adaptation relies on the user's previous experiences, and thus enables personalisation in any context. However, these considerations are very much theoretical. Indeed, as discussed in section 2.4, similarity plays a paramount role in case-based reasoning, as it leads the retrieval of previous cases. In the aforementioned realisations, the way similarity is treated, i.e. how similarity between cases is assessed, is not explicitly discussed and demonstrated. In particular [MaKi05] does not indicate any experimental results. In [SaGa+05] the use of case-based reasoning is said to improve the message filtering accuracy (the percentage of filtering decisions that exactly match the preference indicated *a posteriori* by the user for each message) from 50% with user predefined preference (rules) to 80%. However, the authors recognise that more extensive experiments would need to be carried out to clearly establish the potential of CBR. Besides, the information considered in [SaGa+05] to assess similarities between cases appear to be rather insignificant to enable the realisation of truly context-aware applications. The authors of [SaGa+05] refer to this information as indices (and do not mention context information). These indices include: the type of incoming message (e.g. call for meeting, class announcement, food specials, etc), the sender of the message, the user's current calendar activity, the user's current location as well as the weather. However, it seems that the influence of traditional pieces of context information such as weather, user's activity and location, which characterise the real situation of the user, is counterbalanced by other types of information (type of messages, sender of message). These latter types of information may have a greater impact on the final adaptation decision, given the application domain and the characteristics of the empirical evaluation. Indeed, it seems questionable whether the information "weather" can have a real impact on the acceptance (or the

rejection) of incoming messages. Besides, in the evaluation environment (campus), information about the current user's location (in class, in corridor, outdoor) and the user's activity may have remained quite constant over the evaluation (Was the evaluation performed for diverse user's activity and in various user's location?). Sadly, because of the lack of details regarding the evaluation (the authors have provided a single bar graph) this question remains unanswered. Conversely, the approach, described both in [MiKo04] and [KoAa06], slightly differs, as the objective is to make service recommendations and determine the required subtasks to attain a specific goal (e.g. eating). However, this approach seems complex to bring to bear. The situation should be correctly assessed, i.e. the proper previous case should be retrieved, which can be difficult when numerous different context features come into play. Also, a difficulty resides in the task decomposition to define the subtasks. Finally, it seems arduous to gather user information in such an environment. How can the system assess, for example, that the user prefers Italian restaurant? What if in a "user hungry situation", the user once selects a Chinese restaurant?

Using weights to distinguish between application behaviours, as in [Flan05a], seems similar to considering the different solutions of n cases, as in [MaKi05], even though here, n is not fixed but rather depends on the number application behaviours, the weight of which is non null. Hence, this number can vary for different contexts. However, this solution appears not to be much more than a compromise, as it suggests several solutions to the users and as the ultimate choice has to be done by the user. [NuSa+06] proposes a modular approach that is not confined to a single application. Rather, it aims to provide personalised recommendations for a variety of applications and in various forms (service modality, information, etc). It makes use of a user profile that is able to store multiple user models, enabling each user to centrally store its different models (typically one model per application that the user accesses). Relying on views, the profile component avoids multiple storage of the same user information[4]. One peculiarity of the approach is that it allows several machine learning algorithms to make inferences with the user models. However, it is not clear why several algorithms are needed, as potential specificities of different problem domains are not discussed. Besides, no evaluation of the approach with a context-aware application is given.

Collaborative filtering based approaches radically differ from the aforementioned ones. Here, personalisation relies on information provided by like-minded users. It seems an appropriate approach for enhancing recommendation systems with context information. However, it is doubtful whether such an approach would be suitable for other application domains, where user preferences do not necessarily match a utility function.

[4] Note that the author has participated in the development of this profile component. Though it is not discussed here, a possible extension of this work could consist of integrating the profile management to the system described in chapter 8.

3.3 Context recognition

3.3.1 Techniques for context recognition

In order to support user needs in the context-aware environment, a description of the user context is needed. A fundamental question, when dealing with context, includes the way context is treated and how an accurate description is obtained. Indeed, if the user context is too finely defined, it is unlikely that another context will ever be identical, thus making it difficult to apply previous user information to the current situation. Conversely, if the context description is too coarse, the system will have difficulties to determine the preferences that are really relevant between all of them.

As argued by [FlHi+03] context can be described at many levels, from low-level context, which is information directly extracted from external sources (sensors, or software pieces), to higher level context, where information is transformed. The context level does not characterise the importance of the information in the context description. A piece of information can be low-level and bring knowledge about the situation of the user. On the other hand, a piece of information obtained after transformations may be of no relevance in the application scenario. Rather, the level reflects the number of sources from which the context is derived. At the lowest level a (low-level) context is obtained from a single source, for example a GPS device, which gives hints about the current user's location. Such information defines, however, only a small component of the user's context. Indeed, this can characterise both situations where a user is walking in the location, e.g. visiting a museum, and where the user is stationary in the location, e.g. waiting at a bus station. The aim is then to try to generate a more detailed description of a user's context based on a combination of lower-level contexts. Information which is not given by external sources may then be inferred from these lower-level contexts. For example, the user's activity (going to work) can be inferred from his location ("bus station" obtained from e.g. a GPS device) and the period of the day ("morning" obtained from e.g. a clock, provided no sensor or software piece delivers this information).

Machine learning approaches have been applied to sets of acquired sensor data to recognize contexts. We can distinguish between three categories of approaches to context recognition: knowledge based approaches, supervised learning approaches and unsupervised learning approaches.

Knowledge based approaches

Probably the simplest approach to context recognition has been implemented in various frameworks and consists of transforming raw sensor data into human understandable high-level contexts via predefined rules, as in e.g. [KaRe+02], [HoSc03], [RaCa03]. A slightly different approach is presented in [GuPu+04], where context is modelled and interpreted

based on ontologies. Here, context is represented as predicates written in OWL. An ontology-based context model is developed to provide a vocabulary for representing and sharing context knowledge in the application domain. The structures and the properties of different and distinct pieces of context information are described in an ontology, which includes descriptions of classes, properties and their instances, written in OWL. Thus each piece of context information (e.g. location, activity) is expressed as a statement: (*subName, predicate, objName*). Here, *subName* and *objName* are instances of ontology classes and predicate is a property relation defined by the ontology. This enables the expression of a context event such as “My car is approaching a supermarket” via an OWL description including “my car” and “supermarket” as *subName* and *objName* respectively, and “hasNearbyPlace” as a predicate. Information about instances (e.g. the fact that the user is in his car) is provided by external sources, such as sensors. Furthermore, context interpretation is performed by a context reasoning engine that supports RDF-S and OWL reasoning and general rule based reasoning relying on FOL (first-order logic) rules. This enables the expression of user preferences with regards to an application such as, IF socam:locatedIn(John, socam:MyCar) ^ socam:status(John, driving) THEN (forward incoming calls to voice mail). This rule indicates that when John, a user, is in his car and he is driving, then any incoming call on the user mobile phone must be forwarded to the phone voice mail.

Supervised learning approaches

These approaches aim to recognise clusters or segments that can show which combinations of context features form common patterns in the data. They rely on supervised data analysis methods. The algorithms that are utilised to formulate a contextual state require a user support to define the target class. That is, a user, usually the application or system developer but this also can be the application user himself, manually interacts with the system to name and characterise the current context (e.g. meeting).

[MäHi+01] and [HiMä+01] perform data mining to discover a mobile device user context from multidimensional sensor data, including three axis acceleration, light, temperature, humidity, skin conductivity. Several methods are used for analysing data, including minimum variance segmentation, k-means clustering, and neural algorithms PCA (Principal Component Analysis) and ICA (Independent Component Analysis). These methods reveal data patterns, thus characterising “rough” usage situations. Similarly, [MaAg99] apply a Bayesian framework to classify 12 different human activities from acquired visual information by manually tracking the position of the head on pictures. The approach in [KoKo+03] emphasises sensor fusion. Naïve Bayesian Networks are applied to classify contexts of a mobile device user in his daily activities. A wide variety of different features are combined. Thirteen contexts are extracted from the data derived from sensors embedded into the

device (including, e.g. speech, rock music, classical music, car, elevator, walking, running, etc). The current context is inferred based on a given set of context classes. Context is then inferred in classifying the input vector into one predefined contextual state. The classification task is performed by two separate networks, one for the audio-related contexts and one for the other sensors.

While in the aforementioned activities context recognition is performed in pure experimental settings and is thus driven by system developers, other approaches have searched for ways to bring context recognition into the user's hands, letting him decide how he wants context to be defined. The fact is many contexts are quite personal and can be very different from one person to another, e.g. "in the kitchen" or "running very hard". [VaCa00] proposes to address this by enabling *adaptive context recognition*. That is, the user can teach context descriptions by collecting context features that are clustered into several data patterns and naming each context (e.g. "in the kitchen", "running"). Sensor data are expressed in vectors, which serve as inputs to a Self-Organizing Map (SOM) algorithm. This algorithm enables ordering of the input vectors by assigning them to units of a map. Thus, within the units or neurons similar vectors are grouped together. Ultimately, each unit characterises a context. A supervision stage then follows this clustering. At this stage, the user decides what activities are learned and at what time. For example, as he enters a new context (e.g. the user stands) a list of possible labels is displayed and the user selects one of them (e.g. standing) for characterising the context. In addition, this stage supervises transitions from one context to another. It makes use of a probabilistic finite state machine architecture (Markov chains), where each context is represented by a state, and transitions are represented by edges between states. It results in a directed graph depicting the behaviour of the user with relation to the visited contexts (e.g. going from the kitchen to the stairs). [DeHa+04] extends this approach and enables users to "program" their desired applications. The way users demonstrate the actions to take has already been discussed in section 3.2.1. Similarly, users govern context recognition by literally recording a situation. When they identify a current situation or a context, they initiate the recording of context features. Similarly, when the situation ends (e.g. leave the room, stop running) the recording is stopped, and the relevant events are selected from the data logs. For example, for each sample of captured video, the volume level and a determination of whether there is someone talking are output. When a sufficient number of training examples has been provided, the system is capable to recognise the situation and will perform as demonstrated by the user at the next occurrence.

Unsupervised learning approaches

Conversely, the unsupervised learning approach does not require any target class beforehand. To the best of our knowledge there have been very few research activities that have applied this approach to date.

[BaFl05] investigates the extraction and the definition of high-level context from multiple information sources by clustering data from the information sources. The authors investigated raw signals of the Nokia Context Dataset, a publicly available data set, which consists of a set of features files for 43 different recorded sessions, featuring the same user carrying a mobile phone, sensor box and laptop PC, going from home to the workplace and vice-versa. During the journey, the user walks, takes a bus and Metro, and sometimes uses a car. The approach relies on the analysis of these measured data. Correlations between the changes in the characteristics of several sensor signals, e.g. temperature, cell ID, average user activity and average sound level, are found. These correlations drive the definitions of clusters, which represent contexts, e.g. “walk to and wait at the bus station”, “walk outside to metro” or “inside office”. Furthermore, the study shows that sequences of data sampled from the same user context are classified into the same context. On the other hand, sequences of data sampled from another user context are classified into a different context. [HiFl+03] and [FlHi+03] describe an approach, where context is recognised by fusing context features using a novel method: the Symbol Clustering Map (SCM), an algorithm for the unsupervised clustering of symbol string data based on adaptive learning. Extracted context features are here quantified and represented by symbols. The SCM algorithm performs clustering of these data strings (i.e. data represented by symbols) via source fusion (information available at time t from different sources is combined) and temporal fusion (information gathered over a time span T is combined). The SCM algorithm works similarly as SOM (Self-organising map). It is based on a two-dimensional lattice structure, each node *k* being referred to as a “node set” and associated with a symbol set and a weight vector, set to random values initially. Each cluster of data corresponds to a “high-level” context or contextual state. It has also been claimed that, as the memory requirements for the SCM are minimal, the algorithm can run on mobile devices. Thus, context recognition is not only performed on laptops or desktops but can also be applied in a mobile environment. The algorithm performs unsupervised clustering, as it does not require any user input to label the obtained clusters. However, the SCM assumes a fixed learning period and a random sampling of the input data. These assumptions are potentially limiting in real situations, as it is difficult to predict how long the learning period for an unsupervised learning algorithm should be in order to sample a large number of potential inputs and also to have a random sampling of the input data [Flan05a]. A modified version of the algorithm is proposed in [Flan05a], referred to as K-SCM, that addresses these limitations by not having any defined learning period or learning parameters that need to vary over time in any predefined manner.

3.3.2 Discussion

In the chapter’s first section, we argue for the importance of context recognition in Contextual Personalisation. As context is highly dynamic and ever-changing, it is unlikely that two context descriptions based on a set of features will ever be the same, thus making it difficult

to retrieve user preferences in the user model. The objective of context recognition is to define contextual states, encompassing several contexts and characterising a more generic situation than the one identified by a feature-based context description.

The very first approach to context recognition relies on knowledge bases, where rule based or ontology reasoning is utilised to characterise the user situation. In rule-based reasoning, a set of context features are associated, e.g. using a First Order Logic grammar, to define an application meaningful contextual state (as #People(Room 2401, ">=", 3) AND Application(PowerPoint, Running), in [RaCa03]). Similarly to the approach designed for adapting applications, this requires the system designer to determine the set of rules beforehand. Conversely, context recognition can be performed by reasoning and inferring on an ontology (using a descriptive logic). For example, a context "work meeting" (a very traditional example in the context-awareness area) can be defined when a group of people who are "colleagues", share a given "location", e.g. "meeting room". However, ontologies are based on high-level definitions of concept and knowledge about their interactions (i.e.. relationships). Thus, defining the same contexts for each user is a complex problem [Flan05b]. As remarked in [DeHa+04], ask five people to define a meeting and you will get five different answers.

Alternatively, other approaches relying on learning algorithms have aimed to group contexts into clusters to define contextual states. In the machine learning field, one distinguishes between supervised learning and unsupervised learning algorithms [ThKo03].

In supervised learning algorithms, human support is required to define the target class. These methods combine collected sensor data and reveal patterns, thus characterising usage situations. The patterns are further labelled with a human understandable form either by system developers as in [MaAg99], [MäHi+01], [KoKo+03] or by the user himself as in [VaCa00]. In [DeHa+04] the user does not explicitly name his context, but he intervenes to record the situation. These methods seem more suitable than the knowledge based ones, as statistical data mining methods can reveal the relevant features. However, it requires human involvement in the labelling process. Leaving the naming operation to the hands of system developers leads to a similar problem as in knowledge based approaches. It cannot be guaranteed that the label will be meaningful to all users. When labelling is performed by users, it involves repetitive efforts and does not really facilitate the user experience with the applications.

[BaFl05] has shown that it is possible to define some contexts without human hints about the situation they characterize, from a set of real measured data. [HiFl+03] and [FlHi+03] present an approach, extended in [Flan05a], to context recognition by fusing and clustering context features in an unsupervised way. This approach seems very promising, as it leads to the formulation of contextual states in an online manner and is performed transparently to the users. However, it relies on context features quantified and expressed in a symbolic form

only. In addition, the computed clusters appear as generic and easily distinguishable, since defined contextual states include: “walking outside”, “waiting at the bus stop”, “being at home”. It is questionable whether the method would be capable to recognise some more complex contextual states, such as “working in the office” vs. “having a talk in the office”. As remarked by [BaFl05], this would probably require some adaptation of classic clustering algorithms.

3.4 Privacy methods and techniques

3.4.1 Review

Invasion or loss of privacy has been recognised at a very early stage [Weis91] as a real risk in Ubiquitous Computing. Ubiquitous computing technologies change the privacy landscape by dramatically lowering the cost of data collection, making it easy to gather and share a wide range of data about individuals, all in real-time and in a manner that is machine readable and searchable. The risks posed by Ubiquitous Computing technologies range from everyday ones—such as intrusions by overprotective parents and overzealous marketers—to extreme ones, such as threats to civil liberties by governments as well as dangers to one’s personal safety by stalkers, muggers, and domestic abusers [Hong05].

Methods and techniques addressing the privacy needs in Ubiquitous Computing have been developed in three areas [Lang05]: 1) encryption and authentication, 2) anonymity and pseudonymity, and 3) transparency and trust engines. The first methods and techniques enable communication between two parties to be kept private and not to be divulged to parties not involved in the communication process. Encryption tools allow the prevention of eavesdropping from others in the information exchange, while authentication mechanisms can make sure that only authorised persons can access and operate on the data. The second set of techniques aims to facilitate the anonymous access whenever identification is not required. It also provides the means for using pseudonyms for repeated interactions. Finally, the last class of techniques increase user trust in a data exchange by providing background information about data collection and the data collector. Thus, it controls whether systems will disclose any personal information. A transparency protocol, which has gained much attention in the mobile computing area is the “Platform for Privacy Preferences” (P3P) standard [P3P]. It enables websites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents. It allows an agent to obtain answers to questions such as: “who will receive my personal data?”, “why are these data collected?” etc.

In particular, one of the greatest dangers to privacy brought about by Ubiquitous Computing lies in the extensive use of context information to provide users with relevant information and services. Context information is utilised to customise and filter items in order to deliver

information to users, but also can be disclosed to other users, e.g. interested members of a person's social network. [CoSm+05] [KhCo06] and [OlGr+05] have examined when and how people want to share their context with other people, such as colleagues, family members, etc. These studies have shown that people are very much concerned with privacy when releasing such personal information. Research in context-awareness has focused on guidelines, as well as methods and techniques to capture, process and share context information in a privacy-sensitive manner. Several approaches have been proposed that provide means to tackle aspects of the whole problem. To help designers to create applications that provide end-users with a reasonable level of personal privacy protection, privacy risk models have been investigated [HoNg+04]. They identify the threats for any application user and help developers to refine them into concrete and prioritised issues (e.g. through a series of questions to help designers to refine their understanding of problem space, and provide strategies for managing them). The concept of privacy models has been further extended in [HoLa04], where a framework and an extendable suite of privacy mechanisms to facilitate the development of privacy-sensitive applications are presented. The framework promotes the capture, storage and processing of context information on the end user's computer, as much as possible. Afterwards, end-users can choose what information to share with other entities. Policy-based approaches feature digital rights management to manage context information. Policies, e.g. based on P3P, allow an application to describe how it will store and use provided information [Lang05] and how context information is shared [JiLa02]. [SaGa+05] enables rules specifying what information the user is willing to disclose to others under different conditions, (e.g. "my location should only be visible to members of my team during week days between 8am and 5pm"). In [ScCo+07], a trusted middleware server approach has been deployed that enables users to communicate context to applications with respect to a certain trust level. Alternatively, solutions providing levels of anonymity have been researched. For example, [BeSt03] investigates privacy protection against the threat of being tracked by location systems. The true user identity is hidden from the applications receiving the location. This is enabled by frequently changing pseudonyms within a so-called *mix zone*. A mix zone is defined as a zone (i.e. a spatial area) for a group of users. In contrast *application zones* are defined as the area where users might access a location-based application (e.g. coffee shop). Applications do not receive any location information when users are in a mix zone. Rather, identities are "mixed" there. In fact, users change to a new, unused pseudonym whenever they enter a mix zone. Thus applications that see a user emerging from the mix zone cannot distinguish that user from any other that is in the mix zone, and cannot track back the previous user's locations.

But in Ubiquitous Computing and in particular in contextual personalised systems, context information is not only at risk from a user privacy perspective; the protection of user preferences must be considered as well.

User's privacy concerns are significant and also have a direct impact on user-modelling systems, especially those dedicated to the World Wide Web [TeKo03]. These systems rely on user model information. No silver bullet exists for radically enforcing privacy in user-adaptive systems, be it technical, legal, or social/organizational. However, to protect users, several technical approaches have been developed that can reduce privacy risks and make privacy compliance easier. An overview of these techniques is given in [Kobs07].

Most user-adaptive systems feature application-specific user models, where every application is responsible for collecting and maintaining its own set of user preferences. An approach to privacy in these systems consists of alleviating the explicitly identified link between a user and an application by means of pseudonyms. A users can employ a pseudonym (he freely defines or he picks from a list of available ones) in all his transactions, so as not to disclose his true identity. For example, several infrastructures have been proposed that provide such pseudonym-based interactions with web sites [IsAl+03] [HiKa+05].

Yet, an approach featuring network-wide individual user model servers was proposed (see [Kobs01a] for a review). Servers aim to supply several user-adaptive applications with user modelling services (performing the personalisation processes) for their users. This approach allows for a central user model, enabling the sharing of user data between applications. However, not only the user but also the user modelling may need to remain anonymous to the applications. [KoSc03] investigates the issue for users to remain anonymous to websites, nevertheless allowing websites to track users individually, drawing inferences about them and adapting to them individually. They propose a solution that features pseudonymous identification, which permits the user to choose a unique pseudonym to identify himself in subsequent interactions. Such an identification is realised through mix networks that dissociate a user from his user models residing on a server. Also, an encrypted channel between hosts ensures secrecy of exchanged information during transport. In addition, users can be given different access levels to various information of the models by means of a hierarchical role-based access control model, which distinguishes between 3 roles (consumer, producer and maintainer) and 3 degrees of trust (untrusted, verified and trusted).

As opposed to user model servers, a number of authors have worked on client-based personalisation [CaWo01], [HoLa04], [CoLa06]. They provide solutions, in which user data are located at the client rather than the server side, as well as the processes relying on these data to provide personalisation. The benefit of such an approach is two-fold [Kobs07]. Websites are not provided with user data but rather users are in control of their own personal information (since they reside and are processed on their own device), thus not making it subject to privacy laws. Moreover, users may possibly be inclined to disclose their personal data if personalisation is performed locally upon locally stored data.

Finally, privacy in user-adaptive systems can be supported in providing means to users to

express their preferences with regards to privacy, as requirements on privacy may differ between users or be differently implemented according to countries' laws. Several architectures have been suggested to address this issue, e.g. [Kobs01b], [Fait03].

3.4.2 Discussion

User studies on the disclosure of personal information have acknowledged individual privacy as a major concern in systems offering personalised interactions [Kobs07]. In Ubiquitous Computing, the distribution of invisible computer devices in our physical environment, which are able to communicate and store information, also poses some serious questions. Indeed, the ability of software systems to infer revealing information from sensed personal data can have troubling implications for individual privacy. A variety of methods and techniques have been proposed to address different aspects of issues both in user-adaptive and context-aware systems, including authentication through pseudonyms, trusted middleware servers and a collection of approaches to enable policy-based personal information control. However, no silver bullet exists for tackling the whole problem. As stated in [VaBo+03], privacy will have to be managed through a combination of technology, legislation, corporate policy, and social norms.

Since they inherit characteristics from both domains, contextual personalised applications are particularly sensitive in this regard. Personal information in contextual personalised applications encompasses both context and user preferences. To the best of our knowledge, the issue has never been investigated for this very type of application. One reason may be that these systems have been mostly realised as full-fledged applications, i.e. applications in which the user model is maintained and adaptation is performed. This makes it more difficult for personal data not to be divulged as well as to communicate it anonymously. To date, only [NuSa+06] has proposed a generic infrastructure to support multiple domain independent contextual personalised applications. However, it does not explicitly consider the privacy issue. A promising approach, though, seems to be the client-side personalisation approach developed in [CaWo01], [HoLa04], [CoLa06] for traditional user-adaptive systems. This approach has two disadvantages [Kobs07]: 1) it does not permit application of personalisation methods relying on the analysis of data from a user population (e.g. collaborative filtering and stereotypes) and 2) program code that is used for personalisation can incorporate confidential business rules and methods and must be protected from disclosure through reverse engineering. The authors in [Kobs07] recognise, however, that if these drawbacks pose no problem in the considered application domain, client-side personalisation should definitely be adopted, as it may make users feel more confident with regards to the protection of their information, and it does not generally constrain the system when collecting user information with privacy laws (e.g. only collecting personal information strictly required for the purposes stated in the application privacy policy).

3.5 Summary and conclusion

In this chapter, we presented existing systems related to Contextual Personalisation and discussed their limitations. In order to review these systems, we defined the main challenges posed by contextual personalisation.

The challenges and the limitations of the current approach in this regard are summarised in the following section.

Firstly, contextual personalised systems must be able to collect user preferences and define the associated relevant contexts. This should be performed without bothering the users. Secondly, systems must deal with this issue of confronting contexts and identifying different, though comparable, contexts. This system capability depends very much on the way context is inferred and recognised, i.e. how one obtains a context description. Finally, systems must protect user data, as users are concerned with the disclosure of their personal information to other parties. Systems that do not allow these data to remain private would surely suffer a loss of credibility among their users and eventually dissuade the users from using them.

To date, most solutions to provide Contextual Personalisation rely on rules, which are either manually defined by the application developers or user-configured. Static rules are inflexible and difficult to personalise for individuals. It may be possible to characterise and describe some of the user's situations and the requested specific application behaviours. However, it is arduous to exhaustively define all possible contexts and all associated user requests. In addition, these systems are unable to predict a user preference in an unforeseen situation. Case-based reasoning approaches seem to overcome these limitations. Adaptation relies on the user's previous experiences, and thus enables personalisation in any context, without user interactions. However, similarity plays a paramount role in case-based reasoning, as it leads to the retrieval of previous cases. In the current realisations, the way similarity is treated, i.e. how similarity between cases is assessed, is not explicitly discussed and demonstrated. Other machine learning based approaches have been proposed, but no real evaluation of the approach in the context-aware environment has been presented. In addition, recommendation systems can be performed through collaborative-filtering based approaches. It is, however, doubtful that such an approach is suitable for application domains, where user preferences do not necessarily match a utility function.

The objective of context recognition is to define contextual states, encompassing several contexts and characterising a more generic situation than the one identified by a feature-based context description. Firstly, context recognition can be performed via knowledge bases, where rule based or ontology reasoning is utilised to characterise the user's situation. However, this suffers from the limitation that it requires the system designer to determine the set of rules beforehand. Defining the same contexts for each user is also complex. Alternatively, learning algorithms have aimed to group contexts into clusters to define such

contextual states. When supervised learning algorithms are utilised, a human support is required to define the target class, i.e. name the contextual state of the user, e.g. “meeting”. When this is performed by system developers, no guarantee is given that the label will be meaningful to all users. When labelling is performed by users, it involves quite a repetitive effort and does not really facilitate the user experience with the applications. Conversely, unsupervised learning based approaches do not require human intervention to recognise contexts. However, the proposed solution processes context features expressed in a symbolic form only. The recognised contexts appear as generic and easily distinguishable (e.g. “walking outside”, “waiting at the bus stop”). It is questionable whether the method would be capable to recognise more complex contextual states.

In Ubiquitous Computing, the ability of software systems to infer revealing information from sensed personal data can have troubling implications for individual privacy. A variety of methods and techniques have been proposed to address different aspects of issues. They have, however, never been investigated for contextual personalised applications. In fact, none of the contextual personalised approaches we have considered provide a dedicated technique to address it, even partially. To protect a user’s personal data, a promising approach, though, seems to be client-side personalisation, which has been developed in user-adaptive systems but never performed in contextual personalised systems.

The examination of existing approaches has shown their limitations. In particular, no solutions exist that satisfactorily address the challenges posed by contextual personalisation, despite the flurry of interest in the field. In order to make the scenarios of Ubiquitous Computing come true and to participate in the advance of context-awareness, research efforts have to concentrate on concepts and methods to guide the development of contextual personalised systems and address these challenges.

4 Copernik: A Contextual Personalisation Framework

Existing solutions for Contextual Personalisation do not satisfactorily fulfil the requirements introduced in Chapter 3. In particular, many popular systems have limited capabilities to adapt to user needs in an unobtrusive way to their users, since they rely on static rules, which are inflexible and difficult to customise for individuals. More importantly, these systems are unable to predict a user's need in an unknown situation.

The consideration of these systems has motivated our work on Contextual Personalisation. Our main contributions are presented in this chapter.

In this chapter, Copernik (COntextual PERsoNalised applIcation frameworK), a framework for supporting contextual personalised applications is described. The framework defines and describes the set of operations to perform Contextual Personalisation in addressing the requirements identified in Chapter 3. The central concept of the framework relies on the application of case-based reasoning in order to determine the user's preference in a given context. In any previously unseen context, the user's preferences are determined based on the user's previous experiences.

In the following section we present the framework in detail. Firstly, the framework objective as well as the basic working assumptions are described. The design principles that govern the development of the framework are further presented. Secondly, the framework describes a set of operations - grouped in the so-called Contextual Personalisation cycle – that definesthe various phases to be taken when supporting a contextual personalised application. Thirdly, we argue for a separation between the software entity responsible for performing adaptation (i.e. the contextual personalised system) and the contextual personalised applications. Applying this principle enables the support of privacy in Contextual Personalisation.

4.1 Aim of the framework

The multiplicity of the approaches presented in Chapter 3 advocates the need to provide software support to this class of applications. However, these approaches feature quite limited capabilities. This is due to the lack of guiding principles to support the development of such systems. The actual limitations of these approaches have already been discussed extensively in Chapter 3. But the table presented hereafter summarises how the approaches address the requirements of Contextual Personalisation. In this table, the degree to which the requirement is addressed is indicated as "addressed", "not addressed", or "partially addressed" and a short explanation is given.

Requirements for Contextual Personalisation (Section 3.1.1)	Rule-based adaptation (Section 3.1.2)	CBR-based adaptation (Section 3.1.2)
Performing according to the user needs	Partially addressed When the rules are determined by the user, the application can perform according to the user needs. In some approaches, however, rules are determined by the application developers. As such, the application cannot be personalised to all the users.	Addressed The required behaviour of an application is determined from the user's previous experiences with this application. Thus, adaptation relies on knowledge about the user. The requirement is thus addressed.
Placing minimal demand on user attention	Partially addressed When the rules are developed by the application developers, no action is required from the user. In this sense, no user input is required. However, approaches, where the rules are configured by the users, require cognitive load on the user, which limits the system usability.	Addressed The user is not required to input any information. User experiences are learnt by the system.
Performing under most circumstances	Not addressed This approach fails short when it comes to predict a user preference in an unseen situation. Hence, in contexts, where the rules (no longer) apply, no action can be carried out until the rules are modified.	Addressed Adaptation relies in this approach on the user's previous experiences, and thus enables personalisation in any contexts.
User in control	Not addressed Users of the systems have no ability to control and provide feedbacks on the action triggered	Not addressed Users of the systems have no ability to control and provide feedbacks on the action triggered

Privacy of user data	Not addressed This requirement is not addressed by systems following this approach	Not addressed This requirement is not addressed by systems following this approach

In the following section, the principles of the framework designed to support Contextual Personalisation are detailed.

4.1.1 Framework objective

We aim to develop a framework for personalising context-aware applications, which would overcome the limitations of the existing realisations.

Personalisation: cognitive point of view

The fundamental question in Contextual Personalisation is to determine the relevant user preferences with respect to the current context. When the need for adaptation occurs, the system must have knowledge of the user preference(s) that is (are) meaningful in the current user's context.

From a cognitive point of view, adaptation can then be seen as a decision-making process. Indeed, when the user is in control of the application, prior to explicitly selecting the actions to be performed by the application, he has to make a decision about what he expects. Cognitive science has been interested in the study of mind processes to infer how cognitive processes, i.e. processes involving (human) brains take place. In particular, since the earliest developments in this field, scientists have studied how the decision process occurs in human beings. Two classes of approaches have been broadly proposed: Plan-based approaches e.g. [Alle79] and experience-based approaches e.g. [GiHo80]. In fact, neither approach can fully describe natural decision-making processes. Rather, they describe mechanisms that can lead to good approximation of human beings' cognitive processes.

Cognitive science has influenced machine learning, since the underlying objective is to (re)produce computer systems that behave as human beings. Therefore, the above mentioned approaches are good candidates for determining user preferences in contexts.

The plan-based approach implies that all knowledge about the user's preferences is known a priori, i.e. beforehand, or at least that learnt knowledge is generalised. This approach is implemented in rule-based systems. The overview of existing realisations in Chapter 3 has shown that a rule-based system to support contextual personalised applications is not without limitations. Indeed, this requires the complete knowledge to be acquired beforehand, and this does not allow actions to be taken when the user's context has not been envisioned previously.

Conversely, experience-based approaches seek to define analogies between disparate

domains as a guide to finding solutions for an ill-defined problem [GiHo80]. To do so, a representation of one domain is mapped onto the representation of the other one. The resulting mapping is then used to generate a parallel solution to the target problem. The experience-based approach is implemented in the machine learning field via the case-based reasoning technique.

Hence, this approach appears appropriate to perform adaptation for contextual personalised applications. Then, it becomes possible to determine the user preferences with regards to a given application from his previous experiences with the application.

Formulating the problem

With regards to the limitations of the existing solutions, the objective of the framework can be formulated as follows:

The general goal is to develop principles to guide the development of software systems facilitating user's interactions with contextual personalised applications by determining on behalf of the user the application behaviours to be performed.

These principles are expressed in a framework, thus providing the following guidelines to systems for supporting contextual personalised applications, so that it enables adaptation:

- to be performed in inferring a user's preference based on his previous experiences with applications;
- to be performed in unforeseen contexts;
- to minimise user interventions governing the system, e.g. preventing the users from inputting his preferences in contexts;
- to be performed with improved user data privacy in protecting user personal data.

4.1.2 Basic assumptions

Before presenting the framework, it is important to point out the basic assumptions we have made in this work.

- We assume the existence of an underlying context-aware system.

Context plays an important role in this work. However, it is not the objective of this thesis to provide any specific means to capture and handle context. Therefore, we assume the existence and the availability of a system that is responsible for gathering context information from the environment and for interpreting context, i.e. processing gathered data to represent context in a form specified by the context model.

- We assume the interpreted context is complete and correct.

The description of the context must be:

- complete, that is no piece of information for any of the features is missing.
- correct, i.e. no piece of information is interpreted from wrong sensor data, misinterpreted or comprises “uncertainty”.

Previous research work has investigated the gathering and the interpretation of context information focusing on the development of solutions for copying with uncertainty in context-aware systems (e.g. [RaAl+04]). This is, however, a research topic on its own. As such, we decided not to treat the issue in this work.

- We assume the application behaviours are “exclusive disjunctive” or are performed concurrently or in sequence.

This last assumption concerns the behaviours of an application. In the adaptation process, considered behaviours are either exclusive disjunctive, i.e. only one single application behaviour among all the possible ones can be performed, or conversely all behaviours are performed.

Formulating this assumption, we exclude the case where only a subset of the application behaviours is performed. Indeed, in order to limit the scope of this work, the Contextual Personalisation framework has been developed to support existing contextual personalised applications (four applications that we consider to be representative, are hence discussed in Chapter 6). Performing only a subset of the application behaviours can be envisioned in theory, however, no such case was actually found in reviewing contextual personalised applications from the literature.

4.2 Design principles

From the previous considerations on the aim of the framework, we are able to define principles to guide the development of the framework. We formulate three of them:

- Adaptation governed by a user’s previous experiences

As suggested by the cognitive point of view of personalisation, we must rely on a user’s past experiences with an application to determine in any context how the application has to perform. As a consequence, the user should be asked to neither input any of his preferences nor specify any context description.

- Central access point for user privacy

Privacy is a critical issue for context-aware systems. As discussed in Chapter 2, adaptation of contextual personalised applications requires both user preferences and also a description of the current user context, in order to select the user information that is currently relevant.

However, such information is highly personal and should be divulged only to trusted parties,

i.e. trusted applications. Obviously, user preferences can be sensitive information. By means of this information any malicious party can trouble users, i.e. by sending spams or by simply making the information available to others. Similarly when context information is accessed by malicious parties, conclusions about the user can be drawn. For example, if my superior is informed of my exact location during the day, he could notice that my lunch break exceeded the time I am allowed to take.

Moreover, the vision promoted in the context-aware community [Saty01] is that in the near future we will be able to connect such various and numerous context-aware applications. Then, as the number of applications grows, the risk of facing such malicious applications will increase.

We adopt the central access point as a design principle for the development of our framework. According to this principle, the adaptation decision (i.e. determining how to adapt) has to be performed independent of the applications, thus making it difficult for them to manage (e.g. gather, store, process) any of the aforementioned user data and delivering the only needed data to them.

- User in control

The fundamental attribute of user-modelling and context-aware systems is that they react primarily to implicit user information. A recognised problem with such systems, e.g. in [ChMi+02], [HeIn05], is that designers must carefully balance the way in which the systems adapt and react to these implicit triggers, with a desire for the users not to be presented with unpredictable and unexpected behaviours. This creates a paradox. Computer systems that enable autonomous tasks to be performed based on the usage of personal data are regarded as useful. However, at the same time they appear as dangerous to users and can lead to the abandonment of these systems by their users. These considerations have been reformulated into the following general design guidelines for developers when building context-aware systems:

- System designers need to ensure that the context-aware system maintains appropriate levels of predictability and provides sufficient transparency to enable users to trust the behaviours of the system, to understand what adaptation strategies are in place, and to override these adaptation strategies in those circumstances where such user control is warranted [ChMi+02].
- Feedback mechanisms should be provided, in order to let users i) see how their preferences are linked to actions and ii) override actions if necessary [HeIn05].

We follow these guidelines and adopt in our own work the following dual principles:

1) The system responsible for performing adaptation must present to the user a description of the contextual state in order to explain what has motivated the application reaction.
2) The user must be able to comment on any application reaction in giving feedback. Hence the user must be able to acknowledge when adaptation occurred correctly or deny a proposed application behaviour.

4.3 Contextual Personalisation cycle

In this section we describe the operations to be performed by a system to adapt any contextual personalised application. However, before a system goes through the different phases of the personalisation cycle, the supported application has to meet a set of fundamental requirements. These requirements are discussed in Section 7.1.

4.3.1 Contextual Personalisation cycle phases

Following the first design principle, contextual personalised applications must rely on previous user experiences. Thus, the fundamentals of CBR are employed to define the main stages that are taken to personalise such applications. The basics of case-based reasoning were provided in section 2.4.

The cycle that governs the personalisation follows the guidelines given by the CBR cycle and is depicted in Figure 4-1.

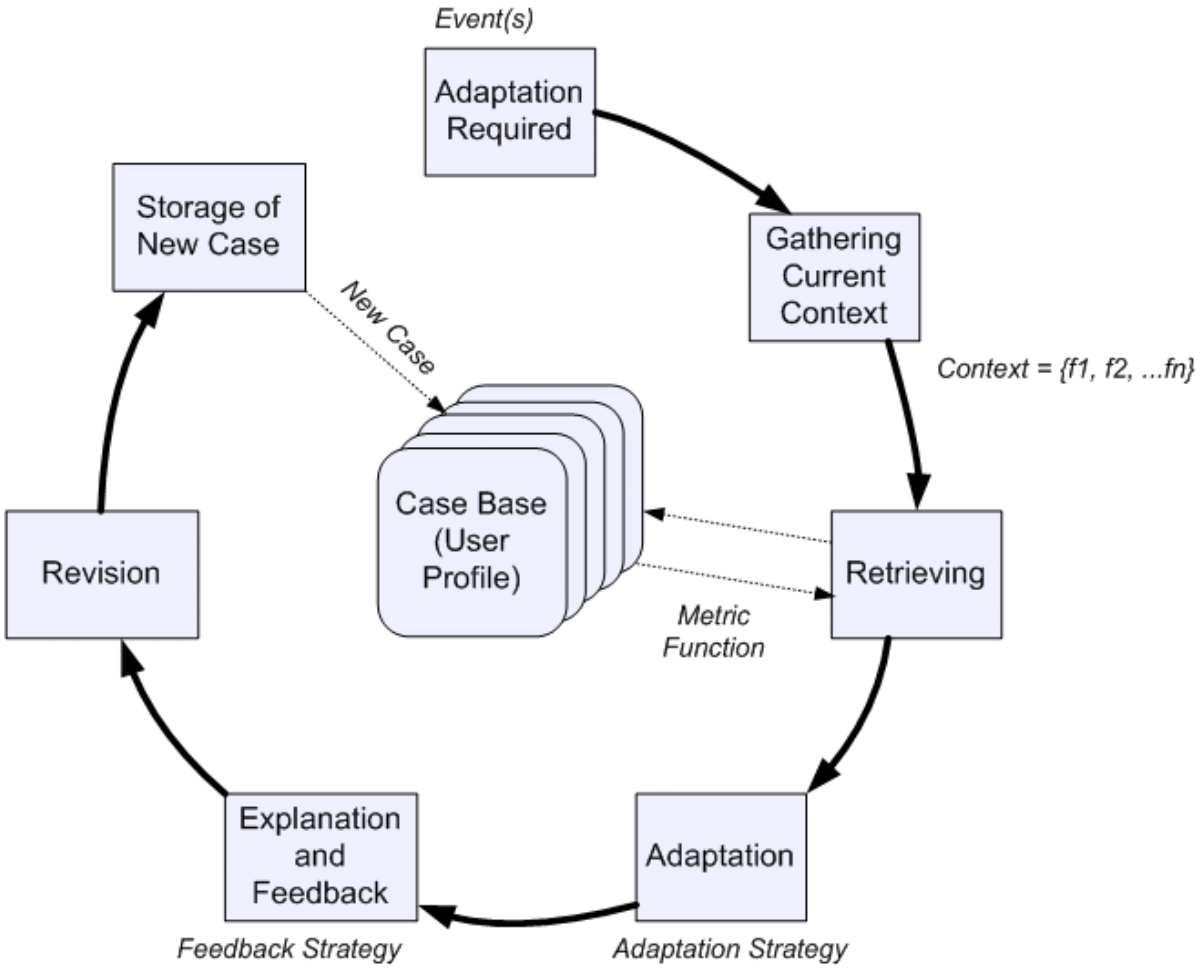


Figure 4-1: Contextual Personalisation Cycle

4.3.2 Case representation

Before discussing the different phases of this cycle, we should examine how cases are represented in the framework. A case comprises the system knowledge acquired from a user's previous experience with an application. As discussed in Section 2.4, a case is composed of a *problem part* and a *solution part*.

For any case handled by systems following the framework principles:

- The *problem part* characterises a user context. More specifically, the problem part contains the set of values for the context features (definition 2.8) specified by the application context model (definition 2.9).
- The *solution part* encompasses any piece of information required to specify the way the application has to be adapted.

Problem part

Context features can differ in their form. A context feature is characterised by the range of values it can take in all the various situations. The framework supports features whose

values belong to three categories.

- Nominal

In this category, features' values are unordered concepts. For example, the feature *activity* may be represented by values such as: running, reading, working, etc. The values of such a category cannot be meaningfully ordered or classified. This category also includes context features with binary values, such as the context feature "busy": yes/no or false/true.

- Ordinal

In this category, features' values are ordered concepts. A typical example of such a category is for the context feature: *temperature in the room* with values ranging from very cold to hot, such as {cold, mild, medium, warm, very warm}.

- Interval-scaled

This category is designed to context features whose values are numerical. One can further distinguish between value*s in* a *co*ntinuous range (e.g. $\mathbb{R}$ and any subset of $\mathbb{R}$) or discrete values (e.g. $\mathbb{N}$ and any subset of $\mathbb{N}$). An example of a context feature with *co*ntinuous range is *the temperature in a room* (e.g. 24.4 °C). An example of a context feature with a numerical range is the *number of persons* surrounding the user in a given location (e.g. 5).

The discussion on the context characteristics in Section 2.2.2 has made clear that context has different representation levels and that a suitable description of context may differ from one application to another. The exact subset of features that is relevant for an application has thus to be identified by the application developer. As a consequence, the number and the label of context features comprised in a case may change very much for different applications. However, the list of features ($< f1,..., fn >$) is unique for an application.

Solution part

The solution part encompasses the pieces of information that are required for characterising the user needs with regards to an application. Adaptation of contextual personalised applications can imply the following actions:

1) select the application behaviour to be triggered;

2) provide parameters related to the user needs;

3) select the application behaviour and provide required parameters.

Thus, the solution part of any case is made of a 2-tuple: $< Behaviours \mid Parameters >$

- *"Behaviours"* specifies the list of triggered application behaviours
- "*Parameters*" specifies (when needed) the values of parameters that are required by application behaviour. The parameters can be either user preferences (e.g. preferred type of music: rock-music), or pieces of user context (e.g. user location)

Case representation

Consequently, a case characterising the user's experience *U* with an application *A* is expressed as:

$$Case(A,U) = \{< f1^U,..., fn^U >;< Behaviours(A) \mid Parameters(A) >\}$$

In the next subsections the different stages of the Contextual Personalisation cycle are discussed.

4.3.3 Triggering personalisation

The first phase of the Contextual Personalisation cycle consists of characterising when personalisation is required, i.e. when the subsequent phases can be initiated. Knowing when to provide personalisation is important for two reasons.

Firstly, users expect an adapted application behaviour just when they want it and would feel bothered if the application reacts at an inappropriate time.

Secondly, the personalisation process consumes software resources. In particular, an up-to-date description of context is required every time. Depending on its very nature, context can often and quickly vary in time. For example, if the physical location is one required feature, the user's context can be updated any time the user is moving. If the user is driving a car, an accurate description of the user's location might change any second.
Getting a context update consumes system resources. Any time context is retrieved, sensors are required to gather information and the system must process it to provide the context description. If the system in charge of this task is running on the user's mobile device; this will require much processing power for a device with limited computing resources. On the other hand, if context is processed on a remote server, resources will still be used to support communication between the client device and the server. The more often an update is required, the more system resources are consumed.

Traditionally two approaches exist when it comes to triggering context-aware applications [ChMi+02]: the push-based approach and the pull-based approach. The approaches differ in terms of the triggering mechanism.

- Pull-based approach

In the pull-based approach, adaptation is triggered upon a user request. For example, when entering a new area of a museum, the user can press a refresh button on the user interface and instantly receive personalised information about the pieces of art in the area. As a consequence, the gathering of sensor data and the processing of context are controlled by the user. Similarly, contextual personalisation can be driven by a user request.

The advantage of this approach is that context processing is required only when the user triggers the application. This should save resources. The disadvantage is that information

required to process the context could be unavailable. For example, no sensor was deployed or is available in the user's environment. Thus, a full description of the context could be missing for the adaptation. More importantly, this approach relies on user requests, necessitating user interactions with the user interface. This is clearly against the framework's objective of minimising user interventions with the system.

- Push-based approach

In the push-based approach the decision to provide context is not up to the user but rather is made by a software entity. For example, when the user enters the room the lighting in the room can be switched on when the user wishes. Similarly, push approaches can be implemented in contextual personalised systems. It requires, however, that the software entity triggering personalisation knows to what events it must react and monitor them.

The push-based approach is adopted in the framework. But it is required that applications explicitly specify the events initiating personalisation. For example, the application, which is discussed in Chapter 7, requests personalisation when a call comes in.

4.3.4 Gathering current context

After personalisation is triggered, the subsequent phases of the Contextual Personalisation cycle aim to determine the application reaction the user is expecting.

Inferring the application reaction relies on the user information, which is available in the framework, i.e. user preferences and user context. Obtaining a description of the user's context implies the availability of an underlying system to perform two functionalities. As discussed in Section 4.1.2, in this work:

- We assume the existence of an underlying context-aware system.
- We assume the interpreted context is complete and correct.

Hence, the user context obtained from the underlying system defines the new problem situation. The new case (for the application A, and the user's experience U) has an unknown solution part and can be expressed as:

$$NewCase(A,U) = \{< f1^{U},..., fn^{U} >; < - | - >\}$$

4.3.5 Retrieval phase

The objective of the retrieval function is, as the name indicates, to retrieve meaningful case(s), which correspond(s) to similar previous user experiences. The strategy to retrieve the similar cases in the case base is threefold.

When personalisation is required, the user is either in a situation he has previously experienced with the application, or in a new situation. The strategy to retrieve meaningful case(s), i.e. cases that will help determine the new case's solution part differs then slightly.

- Searching for obvious similar case

The first stage of the retrieval function concerns the retrieval of what is referred to as *obvious similar case*.

An *obvious similar case* is a case whose problem part characterises a situation, which is identical to the current one. That is, the context description contained in the case is exactly identical to the one of the new case. To retrieve an obvious similar case, problem situations, i.e. contexts, of both the new case and the stored ones are compared.

- Searching for proximate match

When no obvious similar case is found, the second stage of the retrieval function searches for *proximate match* among the set of previous cases.

Features can be nominal, ordinal or interval-scale (4.3.2). In identical situations, features with nominal or ordinal values have to be exactly identical, as do features with discrete values. For example, assume that a feature is the number of persons surrounding the user. Two contexts cannot be identical if, on the one hand one person is with the user, or on the other hand two persons are with him. At best, the two contexts can be very close, if other features are identical. Nevertheless, two contexts may correspond to two very similar situations when the values they take for a continuous feature only very slightly differ. For example, consider room temperatures of 18.2°C and 18.5°C. Even if both values are not the same, the difference might not be relevant enough to characterise different situations. As a consequence, for interval-scale features with continuous values, a threshold value is introduced. When the difference between the features' values is smaller than the threshold, features are considered matching and the case is retrieved. If several proximate matches are found, the case presenting the highest global similarity with the new case is selected.

- Searching for similar cases

When no proximate match is found, the last step of the retrieval phase consists of searching for similar cases. Cases are retrieved based on some similarity measures. Several strategies for this could be envisioned. In Chapter 5, we discuss our own approach: the cluster-based retrieving method.

Converse to the previous two stages, the applied retrieval strategy can retrieve more than one single case, but the strategy must propose a mechanism to cope with the retrieval of cases having different solution parts.

In summary, the retrieval phase is then performed as indicated in the following algorithm:

```
GET NewCase
SET behaviour to null

// Search for obvious similar case
FOR each Case in CaseBase
      SET NbFeature and SimilarCase to 0
      FOR each Feature
            COMPUTE Similarity between Case and NewCase
            IF Similarity is 1
                  Increment NbFeature by 1
            END IF
            ELSE IF
                  Break
            END IF
      END IF
      IF NbFeature is equal to the number of Features
            GET Case
            SET SimilarCase to 1
            break
      END IF
END IF

// Search for proximate match
IF SimilarCase = 0
      FOR each Case in CaseBase
            Set NbFeature, NbCase, ProximateMatch to 0
            FOR each Feature
                  COMPUTE Similarity between Case and NewCase
                  IF Similarity < FeatureProximateValue
                        Increment NbFeature by 1
                  END IF
                  IF NbFeature is equal to the number of Features
                        GET Case
                        Increment NbCase by 1
                  END IF
            END FOR
      END FOR
      IF NbCase > 1
```

```
            COMPUTE Global Similarity (Case, NewCase)
            GET greater Global Similarity
            GET Case
            SET ProximateMatch to 1
      END IF
END IF

// Search for similar cases
IF SimilarCase and ProximateMatch = 0
      IF behaviour is null
            Apply retrieval strategy
            (e.g. the retrieval strategy presented in Chapter 5)
      END IF
END IF
```

If at least one case is detected, the retrieval phase is successful, and the subsequent stage can take place: the adaptation phase. However, when no case is found, e.g. the user has never used the application, personalisation is interrupted and the user is asked for support.

4.3.6 Adaptation phase

The adaptation phase is initiated when the retrieval phase has found one or several cases. In this phase, the solution parts of the retrieved cases are reused to provide a solution part for the new case (which at this stage, is only composed of a problem description, i.e. context part). Further, the solution part is utilised to trigger the application adaptation.

Identifying new case's solution part

According to the very nature of the found cases the approach to determine the new solution part can slightly differ.

- Adaptation for obvious similar case and proximate match

When the retrieval function ends up with an *obvious similar case* or a *proximate match*, the reuse phase is trivial. Indeed, the found case corresponds to an almost or completely identical match to the new case. When an exact match is found, there is no insight that the user need has changed, and one can reasonably consider that the old solution part can be transferred to the new case. Furthermore, when a proximate matching case is found, the differences are not significant. Indeed, thresholds assess when two values are close enough to be consider as almost identical.

- Adaptation for similar cases

When the retrieval function has identified a set of similar cases, two strategies could be implemented.

- Copy, i.e. reuse a solution part without any modification

  In this strategy, the differences between the new problem description and the problem description of the found case(s) are extrapolated. Since the retrieval process has highlighted some similarities between the problem situations, this is enough to transfer a solution part to the new case, without any modification. When the retrieved cases have the same solution part, the solution part to be selected is evident. When different solution parts are present, a voting process can be performed.

- Adapt

  The retrieval function has determined a set of similar cases to the new case. The differences between the new problem description and the problem description of the found case(s) are not extrapolated, but rather utilised to determine the new solution part. As presented in [AaPl94] the new solution can be constructed either through transformation or derivation. Derivational reuse, or planning, concerns how the problem was solved in the retrieved case(s). The retrieved case(s) contains information about the method used to resolve the problem situation (e.g. sub-goals). This information, which can be seen as a plan to determine the solution, is then applied to the new problem situation. As such, the derivational reuse can be seen as a pure analogy-based method. Conversely, transformational reuse concerns the knowledge used to solve the retrieved solution part(s). The knowledge represents operators that were applied to solution part to construct the problem situation. The same operators can thus be applied to the new problem situation to derive the solution part.

  Thus, both derivational and transformation methods rely on some underlying information to solve the problem situation. The transformational reuse relies on domain-dependent knowledge (plans), whereas the derivational reuse utilises information characterising the solution solving (operators).

In this work, the reuse of past solution parts is governed by the working assumption that:

In similar contexts, a user has similar needs with regards to an application.

As discussed in 4.3.2, a case is represented as:

$$Case = \{< f1,..., fn >; < Behaviours \mid Parameters >\}$$

On the one hand, the *"Behaviours"* specify the triggered application behaviours. The adaptation selects one or all the *n* predefined application behaviours. The (slight) differences between the new problem description and the problem description of the found case(s) have then little impact and can be extrapolated. Thus, the behaviours of a solution part are copied.

On the other hand, "*Parameters*" specify (when needed) the values of parameters that are required by the related application behaviours. The parameters can be either user preferences (e.g. preferred type of music: rock-music), or pieces of user context (e.g. user location). Similar to the behaviours, the parameters are copied. However, the information related to the user context is transformed according to the current settings (e.g. the current value of the user's location is applied). Indeed, it makes little sense to reuse a piece of information that is not related to the current user situation.

When several similar cases are retrieved, the applied retrieval strategy must amend the adaptation strategies to cope with it. For example, the retrieval strategy can select the solution part common to most cases (voting). This solution part would then be adapted as mentioned above. This differs from the retrieval function we describe in Chapter 5, as several cases with the same solution part are retrieved.

Table 4-1 summarises the strategies to be deployed to reuse the retrieved solution part.

Solution part elements	Strategy
Behaviours	**Copy**
Parameter (*preferences*)	**Copy**
Parameter (*context*)	**Transformation**

Table 4-1: Adaptation strategies

Triggering adaptation

Once a solution part has been determined, the application must be consequently triggered. How the triggering takes place, i.e. how the system communicates with applications to trigger a method, is specific to the technology used to implement the applications (e.g. web services, java application running locally, etc) and is not considered by the framework.

4.3.7 Explanation and user feedback

Following the application triggering, the new case must be stored for further use.

At this stage in the Contextual Personalisation cycle, a solution has been determined for the new problem situation and is proposed to the user. But it is unknown whether the proposal is correct or not, i.e. whether it corresponds to what the user currently expects. In order to include the case in the user previous experiences and consider it at the next system use, the case's correctness has to be evaluated.

The evaluation of the proposed solution can be performed by asking a teacher (e.g. human or simulation program), that has a good understanding of the correct solution parts [Stah03]. In this work, the assessment of the correct solution to the problem situation is user-specific,

i.e. different users have different and personal requirements with regards to an application. Then a teacher cannot provide an evaluation for different users with various needs. Consequently, the teacher approach does not appear appropriate.

Alternatively, the evaluation can consist of applying the solution in the real world and asking the user. In fact, only the user can assess the correctness of the proposed solution. This is performed in asking the user for feedback. Feedback is an important milestone of personalisation, since it allows users to maintain control of the system by manually correcting the proposed application behaviour, for example.

To fully address the user in control design principle, the framework calls for a twofold strategy:

Providing explanation

Firstly, an explanation of the system's action must be presented to the user. The ability to explain its results is often considered as one of the main advantages of CBR systems [CuDo+03]. As pointed out in [Cass04] explanation can be interpreted in two different ways. On the one hand, interpretation can deal with explanations to guide the reasoning process in CBR. For example, a diagnostic result can be supported by explanations in the course of the process to assess the validity of certain hypotheses. As the personalisation's objective is straightforward and requires minimal user intervention, such explanations are not required. On the other hand, explanation deals with the results of the reasoning process or the usage of the results. This aspect of explanation is relevant in contextual personalisation. Indeed, it provides insights to users about the system's conclusions, but does not ask for guidance during this process.

The central question is to determine what comprises a good explanation for contextual personalised systems and in what form it should be displayed to users.

As pointed out in [SoCa04] the explanation form depends on the goal of the user. In the context of this work, the user aims to have the application react according to his needs in the current situation. Then, the validity of the proposal to a user depends on the system's capacity to correctly assess similarities between contexts and retrieve similar previous experiences.

Knowledge in the framework is expressed as cases. The most obvious approach to explanation would consist of presenting the retrieved case(s) as such. However, this may be of little avail to users. Indeed, a case is composed of a problem situation part, which describes context through a set of features' values. Users may then have little understanding to what the values refer. Besides, as detailed in Chapter 6, all features may not be relevant to characterise the current situation.

In fact, the system reasoning process, as underlined in the retrieval phase (4.3.5), is driven by the comparison between the new case's problem situation (i.e. the current context) and the previous cases' problem situations. As a consequence, the framework calls for explanations that would display to a user a description of the current context (i.e. the current situation as perceived by the system) and a description of the retrieved cases' context.

In addition, as these descriptions are dedicated to users, both have to be presented in human understandable ways, meaning that:

- only features deemed relevant for characterising the user's situation must be presented;
- the semantic (i.e. what they refer to, e.g. location, time …) must be indicated.

However, as a consequence of the principle of central access point for user privacy (4.4), explanations have to be performed independent of the applications. Indeed, if applications are responsible for giving explanations to users, they have to get a description of the user's context. However, in defining a central access point for performing contextual personalisation, we prevent too much personal user data (including context information) from being delivered to the applications. Here, applications do receive only information that is required to personalise the application.

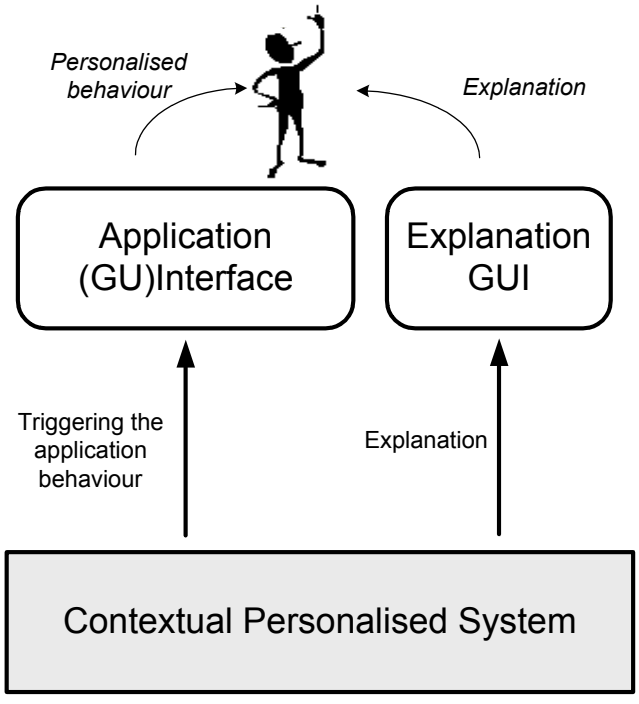


Figure 4-2: Application vs. Explanation Interfaces

Thus, an Explanation GUI is required to display the system's explanations independent of the contextual personalised applications. This is depicted in Figure 4-2. Note that the framework does not constrain the way feedback is gathered. Following the application-specific feedback mode, feedback can be collected from the application GUI or from a distinct GUI.

Gathering user feedback

Following the explanation, the user's comment on the system's decision has to be gathered via user feedback. Users are thus able to acknowledge when adaptation occurred correctly or conversely, to deny a proposed application reaction.

In a nutshell user feedback can be expressed in two ways: explicitly or semi-explicitly.

- Explicit user feedback

As the name indicates, feedback here is given explicitly by the user. Every time an application is personalised, the user is asked to indicate whether he agrees with the selection or not. This can be performed e.g. through a pop-up window.

The advantage provided by explicit feedback is twofold. Firstly, it permits the system to know with a high certainty how the proposal is assessed. The system knows then whether the proposal is correct or if it has to be revised. Secondly, it fully addresses the user in control design principle: personalisation never occurs without the user knowing it.

However, this requires an action from the user any time personalisation is performed. It can then bother the user when personalisation occurs too often and/or when the advantage brought by the personalisation in terms of user's investment seems to be too little. Indeed, if the user has the impression (it may even not be true) that he barely benefits from a personalised application (as he has to input feedback too often) this can dissuade him from regularly using the application. Or in worse scenario, he may prefer a non-personalised application altogether.

- Semi-explicit user feedback

Unlike explicit feedback where the user interacts each time with the system, with semi-explicit user feedback the user specifies only one type of feedback. The other feedback is selected by default, when no user's reaction is monitored.

The user can give feedback on a *reward basis*. When the application behaves as expected, the user notifies his acceptance, e.g. by pressing a button. No reaction from the user is on the contrary perceived as a negative feedback.

A more natural way of giving feedback is on an *interruption basis*. Negative feedbacks are given explicitly by the user, e.g. pressing a button, closing the application, etc. Conversely, no reaction monitored means user's acceptance.

Semi-explicit feedback on reward basis and interruption basis leads to opposite results. After a certain learning time, the system is expected to correctly personalise an application. Then the number of positive feedback should become larger than the number of negative feedback. Using the semi-explicit feedback on a reward basis, the user will tend to react more and more often to the personalisation process. Eventually, he will be asked to give an explicit feedback (a reward) anytime personalisation is performed. On the other hand, using

an application implementing semi-explicit feedbacks on an interruption basis, over the application's uses, the user will react less and less with explicit feedback. This can increase personalisation acceptance. However, it reduces the user's feeling of being in control.

It appears then that there is no unique preferable way to gather user feedback. Rather, application developers must consider the application characteristics and evaluate the trade-off between effort and benefit to users in order to decide what gathering solution is best. For example, when an application is only rarely (few times a day) adapted, explicit user feedback can be appropriate. Developers should take great care in selecting the feedback mode, as it can have a great influence on the personalisation's success. User feedback is the only domain knowledge the system can acquire to assess the validity of the proposed application's reaction, as the user only knows what he expects in his current context. Thus, getting correct user feedback is paramount. If feedback is erroneous, wrong application behaviours will be provided but also an incorrect case will be retained and used in the further personalisation process (a user can mistakenly input a positive feedback instead of a negative feedback, or vice versa). Also, he can overlook a personalised event and consequently not give any feedback. If the application implements the semi-explicit feedback, the system will still handle it as either user's acceptance (interruption basis) or denial (reward basis). Different notification means can be thus deployed by the system (short "beep", smooth vibration, screen briefly alighted) to signal a feedback need.

4.3.8 Revision

When a solution generated by the reuse phase is not correct, the system receives a negative feedback. The solution has to be revised, i.e. a new solution to the problem situation has to be given. Again different strategies for revision can be deployed.

Revision strategies

- User correction

Though the system has received negative feedback, it does not suggest any new behaviour. Instead, the user himself can select the application behaviour that best matches his preferences for the current problem situation.

- System correction

The system computes an alternative proposal, based either on the set of retrieved cases or in performing the retrieval phase again, if the retrieved cases cannot provide a lead to a different solution part. Then, the previous retrieved cases are ignored. Thus, the new system's proposal needs to be validated by a user's feedback.

This strategy appears to be advantageous since it does not require any manipulation from the user apart from providing feedback. However, it can go wrong when the system proposes several incorrect application behaviours consecutively. It seems then required that the user,

after a limited number of system corrections, should regain control of the selection.

- User-supported correction

Here, the selection of the best application behaviour for the current problem situation is eventually performed by the user, but the process is facilitated in using the results obtained from the previous retrieved phase. For example, the user can be presented with a graphical interface ranking the remaining *n-1* application's behaviours, in a decreasing order according to the similarity assessment computed: the solution part of the found second case(s) being on the top, etc.

Following a positive feedback (obtained directly or after revision), a new case is formed. The new case is composed of the current problem situation (i.e. the current user's context) and the correct application information (behaviours and parameters).

Similar to the user feedback mode, the choice of the revision strategy is application-specific and should be specified by the application developers.

4.3.9 Storage of new case

In the last phase, the new case is incorporated into the existing knowledge.

The retaining phase differs slightly according to the type of cases found in the retrieving phase. When no exact match was found, the new case is retained and added to the case base. No other operation is requested. In contrast, when an exact match was found, the new case is retained while the exact match is deleted, in order to avoid storing multiple identical cases.

It is worth noting that the retaining phase can be delayed, as long as no user feedback has been received.

4.3.10 Degrees of freedom in the framework

The Contextual Personalisation cycle describes a set of operations that any system required to support contextual personalised applications has to implement.

However, for some no specific recommendation on how the task has to be performed is made by the framework. Indeed, it appears that some operations can be performed slightly differently according to the application to be supported. Hence, different applications will require different mechanisms. Besides, the results of the retrieval phase are influenced by the retrieval strategy (metric function) deployed to assess similarity between cases. The specific function is not discussed within the framework, as different alternatives can be proposed.

In Table 4-2, we list all the phases of the cycle and for each of them we indicate, when applicable, the framework's degrees of freedom. A degree of freedom defines a design

decision, which is left to the system developers. For each of them, the right column indicates the chapter where the decision related to the degree of freedom is mentioned.

Cycle phases	Degree of Freedom	Chapter
Triggering (4.4.4.)	**Application events the personalisation is triggered by**	**Chapter 6**
Context gathering (4.4.5)	**none**	**--**
Retrieving (4.4.6).	**Retrieval strategy for assessing similar cases implemented by the system (e.g. Nearest neighbour, cluster-based, etc)**	**Chapter 5**
Adaptation (4.4.7)	**Complement of the retrieval strategy when several cases are retrieved.**	
Explanation (4.4.8)	**Specific to the retrieval strategy for assessing similar cases implemented by the system**	**Chapter 5**
Feedback (4.4.8)	**Explicit user feedbacks** **Semi-explicit user feedbacks -reward basis** **Semi-explicit user feedbacks - interruption basis**	**Chapter 6**
Revision (4.4.9)	**User correction** **System correction** **User supported correction**	**Chapter 6**
Storage (4.4.10)	**none**	**--**

Table 4-2: Degrees of Freedom vs. Cycle's phases

4.4 System-based Contextual Personalisation

Contextual personalised applications can provide to their users personalised support by offering them the behaviour they expect from applications. Thus, it prevents them from the tedious operation that consists of manually inputting information on a user interface. Adaptation is performed using user personal data, where personal data refers to any information relating to an identified or identifiable natural person [VaBo+03].

In the scope of contextual personalised applications, personal data are twofold:

- User context, which characterises the situation of the user.
- User preferences, which give indications about what the user likes, expects, thinks, etc, with regards to the context in which they are relevant.

4.4.1 Central access point for user privacy in Contextual Personalisation

Before detailing our approach for improving privacy in contextual personalised systems, we provide a rapid review on how the existing realisations are structured and how the access to personal data is managed.

In Chapter 2, we introduced the term contextual personalised applications and presented this class of applications as a subclass of context-aware applications.

In a first category, applications are implemented as monolithic structures, just as the first context-aware applications were presented, e.g. [ScAd+94]. These applications both handle context, relying on a set of sensors to provide a description of context, and control the acquisition and the management of user preferences.

A breakthrough in the area of context-awareness occurred with the work of [Dey00]. Not only did [Dey00] provide a novel definition of context, but he also introduced a separation of concerns between context acquisition and the use of context in applications. Following this principle, context-aware systems have been developed that aim to support applications via various mechanisms in 1) gathering sensor data from the user environment and 2) building a description of user context. The second category of applications encompasses the contextual personalised applications that are supported by such a system. They are no longer in charge of collecting and processing context information to build a context description. However, they are responsible for the user preferences, as no specific module is dedicated to this task in the underlying system. Moreover, these context-aware systems communicate the user context, when this is requested for performing personalisation.

It appears then that, whatever the approach followed by the application, i.e. whether it is implemented as a monolithic software entity or it relies on a context-aware system to handle context, context-aware applications are provided with the user context as well as the user preferences.

In order to improve the privacy of user personal data, we develop a solution that relies on the principle of separation of concerns between the entity in charge of adaptation and the application. To the best of our knowledge, this solution has not been proposed so far in the literature on Contextual Personalisation. The motivation for this is indeed that, the best way to constrain applications to not process user personal data is to avoid giving it to them.

Adaptation for the considered class of applications is performed following a three-step process: 1) user's current context is determined 2) the context is used to determine the

relevant user's preferences that give hints about the user needs with regards to the application 3) the selected application behaviour is triggered. Consequently, the central access point design principle argues for logical software separation between contextual personalised applications and contextual personalised systems and redefines the computational responsibilities for these two entity types within an infrastructure:

- Contextual personalised system

A contextual personalised system extends the traditional context-aware system in providing additional capabilities. No only does it handle context, until it provides a description of context, but also it handles user preferences and selects those that are relevant in the user current context. Thus, it performs the first two steps of the personalisation process. It ends when the system has determined the application behaviour currently requested by the user and the required user information has been selected.

- Contextual personalised applications

Here, the functionalities are much more limited with respect to personalisation. Indeed, as the requested application behaviour is communicated by the system, the only task of an application is to correctly trigger the mentioned behaviour.

The responsibilities of the different entities and the flow of information are depicted in Figure 4-3.

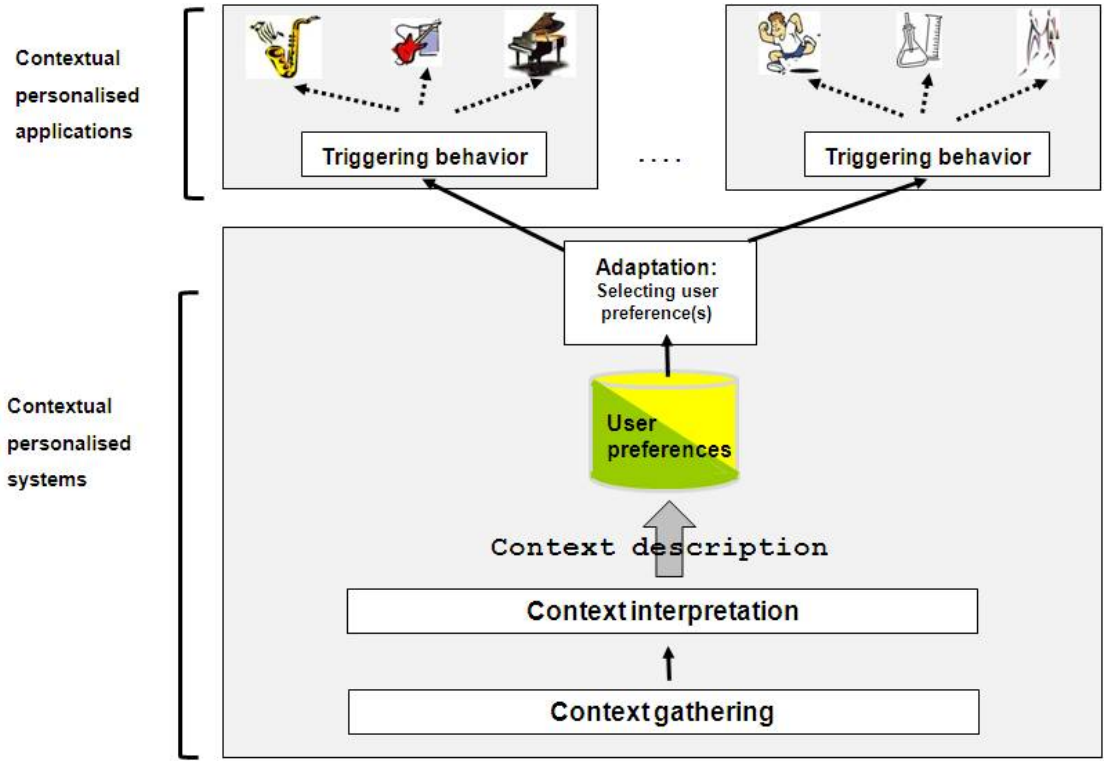


Figure 4-3: Structural view of contextual personalised systems including context handling and adaptation

The separation of concerns provides several advantages with regards to privacy of user personal data:

- Applications receive a very limited subset of user's personal data.

Separating the adaptation process from applications ensures more privacy of user data. No user wants to divulge private data to parties he does not trust. Having personalisation performed independent of the applications prevents applications from having knowledge of all personal user data. Applications simply know what application behaviour is at the time requested by the user and get (when needed) a minimal subset of user information. They are neither notified of the user's situation nor do they get the whole set of user preferences and their relevance assessment in contexts. In fact, it seems that the piece of information which is communicated is the minimal set of mandatory data required for personalisation.

- Facilitating anonymity

The separation of concerns principles does not prevent, however, an application from deploying on its side mechanisms to identify the user and get knowledge of part of the user's situation. Nevertheless, the fact that the application is not in charge of processing the context and user preferences facilitates the deployment of policies and methods for anonymity. Indeed, when an application maintains a user profile, it is difficult to avoid the user from being authenticated by the application [Kobs07]. Through the proposed principle of separation of concerns, anonymity is facilitated. User information can be delivered to the application, but it does not require the user to be authenticated.

- Logical central processing unit to maintain privacy

The separation between application and system also enables the application of a well-known architectural pattern used for privacy: the *single access point* architectural pattern [YoBa97]. The single access point provides a central place to handle and process user personal data.

This pattern is for example used in UNIX telnet and Windows NET for enabling applications to log into the system [YoBa97] and in user-adaptive systems such as [CaWo01], [CoLa06], as discussed in Section 3.5.

Also it produces a central place to apply additional security patterns [YoBa97], such as

- Check point

  Taking advantage of the single access point, developers can design checks to determine if an entity is trying to break into the system. As the system offers a single entry point checks can be easily managed and maintained.

- Role

  Role objects can be developed that define permissions and access rights for an external entity. For example, based on a trust model, the system can determine if the application that is about to be personalised is trustworthy.

The different functionalities of a system - gathering, interpretation and adaptation - can be

distributed over the network. Privacy enabling technologies must be consequently deployed to provide secure communication between the nodes. A large variety of such technologies has already been developed and investigated to improve privacy in distributed environment. For this reason, they are not discussed in this work. However, when the entire system is deployed on the user's personal device (e.g. smart phone) an additional advantage of this approach is that it is generally not subject to privacy laws [Kobs07].

Uniquely applying this principle of separation of concerns does not fully address the issue of privacy of personal data. In particular, it does not prevent malicious software agents from accessing user information that is processed in the system. However, it provides a solid ground in reducing, at a minimum, the user personal data that are communicated to an application and allowing the applications to manage their own user profiles. In addition, the principle enables the deployment of various additional privacy mechanisms that facilitate the clear separation between both the system and application's functionalities.

However, this central access point presents a drawback, which should be mentioned at this stage. The consequences of an attack against a central processing unit from any malicious agent, thus successfully accessing the user's personal data, are much greater as opposed to those of the same agent against an application maintaining its own user profile. Indeed, user's personal data regarding to-date all accessed applications can be divulged.

4.4.2 Additional advantages of the approach

Besides, as personalisation is governed by the user's preferences contained in user profiles, performing adaptation in contextual-personalised systems requests that the user modelling task must be performed independent of the applications. Therefore, it implies several advantages of generic modelling systems [FiKo00].

- Maintaining a central user model that serves several applications. Performing adaptation in contextual personalised systems allows information about the user to be maintained in a central repository and to be accessed by several applications at the same time. For example a data, specifying the preferred type of music, can be utilised for both a music player and an online music store application.
- Ensuring information is stored in a non-redundant manner. Each user's preference can be stored only once. This can ensure consistency of user information [CoSu+06].
- Inferring new pieces of information. User data that serve different applications can also be inferred to determine new user data (for the same or other applications).
- Facilitating the development of contextual personalisation applications. Developing applications is simplified, as they no longer need to perform operations on the user profile. Thus, a contextual personalised application being supported by a system does not implement the operations of e.g. collecting user information, maintaining an application-specific user profile and inferring the application behaviour. The development

of contextual personalised applications is discussed in Chapter 7.

4.5 Conclusion

The limitations of existing realisations for Contextual Personalisation have been discussed in the previous chapter. To address these limitations, we proposed a framework, the aim of which is to drive the development of new solutions to perform contextual personalisation. The development of the framework is governed by a set of design principles.

The first two design principles drive the determination of a set of required operations to provide adaptation of contextual personalised applications. These operations are grouped into a so-called Contextual Personalisation cycle.

- Relying on previous user experiences

Personalising applications by relying on previous user experiences allows them to perform adaptation by inferring user's preferences even in unforeseen contexts and without requesting user interventions to e.g. input his settings.

- User in control

Among the set of operations, some are specifically dedicated to allow the user to be in control of the application. In particular, every time an application gets personalised, an explanation, providing a comparison of the user's current context and a previous context used as a reference, is displayed. Additionally, the user assesses the validity of the action by giving feedback.

In order to evaluate the correctness of the defined guidelines and operations of the framework, we have developed a contextual personalised system. This demonstrates how applications can profit from the support following the framework. The architecture of the system is described in Chapter 8.

- Central access point for user privacy

The concern of users for the safety of their information (both context and preferences) is addressed in calling for software functional separation between contextual personalised applications, which provide adapted behaviours to users, and an underlying system, which is in charge of the determining how the application has to react, i.e. the adaptation decision. This prevents applications from maintaining and processing all user information. Rather, only information mandatory to the adaptation and required by their specific operations is passed to the applications. However, this makes the system more sensitive to attacks from malicious software agents, as all the user's personal data can be accessed, instead of the subset maintained by each distinct application.

5 Retrieving contexts

In Chapter 4, a framework for supporting contextual personalised applications was presented. The aim of the framework is to present guidelines for supporting the personalisation of context-aware applications. One central operation consists of retrieval of a user's previous experiences with the application, whose problem parts are similar to the user's current context . To this end, a three step-based strategy was introduced. When neither identical case nor proximate match can be found, a retrieval strategy is applied that determines what cases are similar to the current one. However, at the stage of this thesis, no concrete strategy has been presented to perform this specific operation.

In this chapter, an approach for retrieving similar cases from the case base is presented in detail. Firstly, the task of retrieving similar cases is discussed and the design principles as well as the requirements that drive the development of such a strategy are detailed. The retrieval strategy, which is further detailed in this chapter, is based on the agglomerative clustering technique. The principles of agglomerative clustering are then further highlighted. Finally, the retrieval strategy, referred to as "cluster-based function" is presented.

5.1 The retrieval task

In the following section we discuss the task of identifying a set of cases similar and meaningful to the current user's context. In particular we discuss the main principles that guide the development of the retrieval function developed in this work.

5.1.1 Objective of the retrieval task

As underlined by Tversky in [Tver03], similarity plays a fundamental role in theories of knowledge and behaviour. It serves as an organising principle by which individuals classify objects, form new concepts and make categorisations. To determine their preferences or expectations towards an application in a new situation, individuals might consider the situation and try to relate it to some previous ones by considering the similarities. The objective of the retrieval task within the framework can be seen as an approximation of the internal cognitive process of individuals.

The framework described in Chapter 4 encompasses a phase in which previous user's experiences with an application are processed to ultimately lead to the determination of the new and for the current context relevant user preference(s). This retrieval phase, as defined in the framework, occurs right after a description of the current user's context has been gathered.

The fundamental objective of this phase is thus twofold:

- To retrieve meaningful cases

  It aims to search among all the user's previous experiences (cases) and to select those that are deemed to have a high level of similarity with the current context.

  The retrieval phase is therefore the main step in the problem solving process. The general procedure is discussed in section 2.4. When the first two steps do not come up with an obvious case or a proximate match, the retrieval function has to search for similar cases.

- To provide explanation

  The framework also calls for providing users with explanations about the application adaptations. Explanations consist of displaying to users a description of the current context and a description of the context of the retrieved case(s), presenting only features deemed relevant for characterising the user's situation.

Thus, the retrieval phases starts with the search for similar cases and ends up with a description of the context of the retrieved case(s). The retrieval phase is further complemented by the adaptation phase. Any case is composed of a problem part, describing a user's context, and a solution part, encompassing information characterising the preferred application behaviour. The retrieval phase starts with the current context's problem description and is responsible for determining one or several relevant cases. The adaptation phase utilises the solution part(s) to determine the solution part of the current case. This is depicted in Figure 5-1.

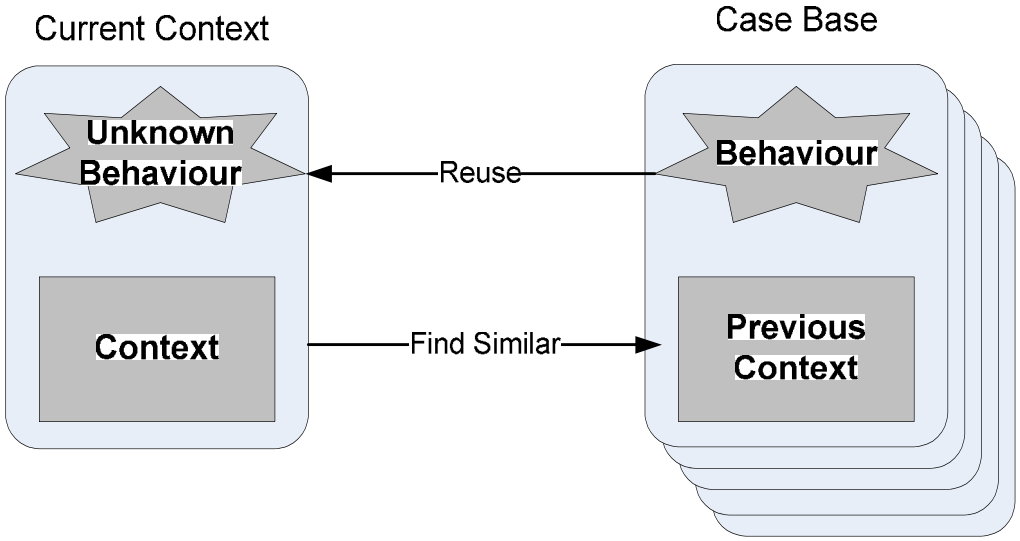


Figure 5-1: Objective of the retrieval function

The strategy for assessing similar cases implemented by the system can be diverse. It may consist of selecting and retrieving a single case (the most similar) or a set of cases.
The fundamental question of the retrieval phase is to determine how to select relevant similar previous cases. When dealing with similarities, one must specify in what respect two objects are similar or the statement is empty. Indeed, two objects such as “a leather couch” and “wooden chair” can be assessed as very similar when focusing on their use (i.e. one typically sits on them), while they may appear quite dissimilar considering their texture (leather vs. wood).

The framework manipulates objects consisting of the problem parts of the cases. Each case’s problem part is represented as an n-dimensional vector (<f_1, .., f_n>), whose features indicate values of the defined application-specific *context features* (see definition 2.8). One difficulty we face when dealing with context concerns its description. As discussed in Chapter 2, there is no clear and general vision of what pieces of information can be part of a context description. It depends both on the application domain at hand and on the “pieces of information gathered from the environment” upon which the user bases his decision regarding the preferred application behaviour.. Thus, context descriptions can be very different for different applications. A context model thus guides the description of contexts in specifying what elements are to be part of it. Similarities between contexts must be assessed based on the context features only.

As introduced in Chapter 2, there is a fundamental difference between situation (see definition 2.6) and context (see definition 2.13). A situation is a part of the world state at some point in time. A situation is rather *perceived* more than *described in extenso* by human beings. In contrast, context characterises the situation, conforming a context model expressed as a subset of features. Context is thus a computational construct that approximates a situation by representing this situation via a limited (though possibly very large) and predefined number of features. Only this way (by conforming a context model) can a context be handled by machines. The pieces of information that are part of the context description are then fixed for a given context model. This means that no additional feature (e.g. room temperature) can be considered for a particular context, when it is not defined as such by the context model. In the current realisations (including this work) a context model serves at least one application [BaDu07]. This differs very much from the way situations are cognitively handled by individuals. Human beings select the relevant pieces of information they consider relevant for every specific situation.

5.1.2 Design principles and requirements

In order to cope with the context peculiarity for the retrieval task, we define design principles to be addressed by the retrieval function of the framework.

Design Principle 1: User-specific similarity assessment

In traditional CBR systems, the retrieval phase relies on a similarity metric (also referred to as function) (section 2.4). The similarity metric remains the same for all uses of the system. However, in this work, no unique similarity function can be used.

Indeed a user A can assess the contexts “reading a book at home in the morning” and “reading a publication in the office in the afternoon” as the two most similar contexts, whereas a user B can prefer assessing “reading a publication in the office in the morning” and “writing an article in the office at night” as most similar. It appears that for user A the activity “reading” is determinant when comparing the two contexts. On the other hand, user B brings more value to his current place (“office”).

Thus the retrieval phase must allow user-specific assessment of similarity, enabling different users to have different concerns when comparing contexts.

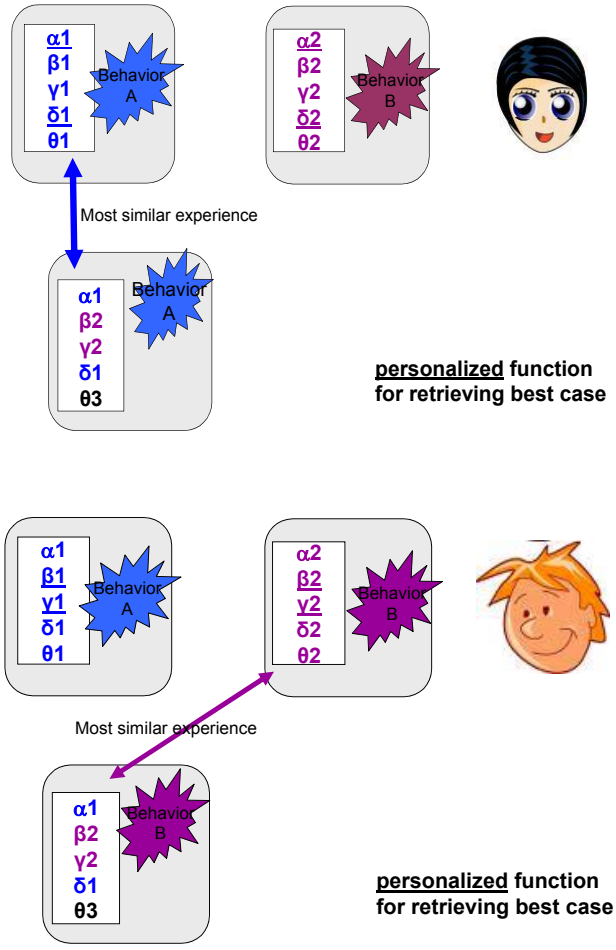


Figure 5-2: Example of personalised functions for two users

Figure 5-2 depicts the comparison of the exact same contexts for different users. Assume two users have experienced the same two situations and expressed for each of them the same preference. User A and user B have for the context <α1, β1, γ1, δ1, θ1> the preference *Behavior A*, while they have the preference *Behavior B* for the context <α2, β2, γ2, δ2, θ2>. Further assume they experience the same new situation, and thus are in the same new context <α1, β2, γ2, δ1, *θ3*>. If the metric does not allow results to be personalised, the context similarities must be assessed in the same way for both users. It would then not be possible for different users to have different preferences in the same contexts.

Design Principle 2: Feature-specific similarity assessment

The second guideline concerns the way similarities are assessed by one single user within different contexts.

When perceiving and cognitively assessing a situation, individuals might not always focus on the same pieces of information. In fact, individuals tend to "intuitively" perform a selection of the currently relevant pieces of information for distinguishing a situation. For example, an individual might not mention the room temperature to describe his situation when being in a meeting. He might consider the location and the other individuals in the room as highly characteristic of the situation though. Besides, the relevance of the pieces of information can change according to the situation at hand.

Consequently, features defined by the context model are not equally relevant for all contexts. Some context description features can be significant, whereas others are not. We refer to this as *feature specific relevance*. For example, a user might assign high value to his current activity at some point. However, he might later consider "time of the day" as more significant. And thus, when comparing the contexts: "*listening to music in the morning*", "*reading in the morning*", "*watching a movie at night*", he will thus assess the first two contexts as the most similar.

It is then important to retrieve similar past cases in such a way that for different experiences, different features can be regarded as significant.

Design Principle 3: Presenting the user with an explanation

One of the design principles of the framework indicates that the system performing adaptation on behalf of the user must provide an explanation. This is discussed in depth in section 4.3.7. Following the retrieval phase, the system must present the user with a human understandable description of the context assessed by the system as similar to the current one. As a consequence of the previous guideline, the explanation must consist of a description of the context made of the features relevant to the user. Furthermore, in order for these features to be of some avail to the user, their semantic (i.e. what they refer to, e.g. location, time …) must be also indicated.

Hence, following their retrieval by the retrieval function, the problem parts of the cases must be transformed in order to provide the explanation.

These three guidelines serve to define a suitable approach for retrieving cases, whose solution part is similar to the current context.

In Chapter 2, a set of machine learning techniques for acquiring implicit user knowledge is described. These techniques aim to obtain knowledge about users in order to make predictions about the user preferences. Hereafter, their applicability to the problem addressed in this thesis is discussed. The reader is invited to refer to Chapter 2 for a description of the actual techniques.

- Linear models

Linear models are applicable when two vector spaces can be mapped onto one another. A relationship is thus established between the input vector space and the output vector space. In the specific case of contextual personalisation, applying a linear model would require the definition of a set of functions in order to map the vector space of *contexts* onto the vector space of *user preferences*. Searching to define such functions is arduous and could be an entire research topic. In order to limit the scope of this work, this approach was not further investigated.

- TF-IDF-based models

The TF-IDF-based model approach is used to determine the similarity between two documents. It uses a set of features to represent an item (i.e. document). Each feature describes a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. While, this technique is not directly applicable to contextual personalization, it still has some commonalities with the approach described in Chapter 4. Context is indeed represented as a set of features. However, the similarity between two documents is computed here in applying the cosine similarity function. Hence, this approach does not address the second design principle stipulating that for a given user, features defined by the context model are not equally relevant for all contexts.

- Markov models

Markov models are applicable to the analysis of complex systems, which operate in different states. This analysis yields results for both the steady state of the systems and the time-dependent evolution of these systems (transitions). However, in contextual personalisation, the different contexts in which the user might find himself cannot be considered as different

states of a same system. Rather, the contexts are independent of and unrelated to one another. In short, the selection of a specific user preference is not influenced by the previous user preference.

- Rule induction

The limitations of rule induction are actually discussed in detail in Chapter 3. A review of the current research developments indicates that rule induction is currently the preferred method for personalising applications in context (see discussion in Section 3.2). Rule-based systems are however unable to predict a user preference in an unforeseen situation. As such, the rule induction technique will not be further considered in this work.

- Classification

Classification designates the set of methods where a set of objects are portioned into classes according to the attribute values of these objects. The method discussed in the subsequent section falls into this category.

5.2 Agglomerative clustering principles

Cluster Analysis, also called data segmentation, has a variety of goals. All relate to grouping or segmenting a collection of objects (also called observations, individuals, cases, or data rows) into subsets or “clusters”, such that those within each cluster are more closely related to one another than objects assigned to different clusters. Central to all of the goals of cluster analysis is the notion of degree of similarity (or dissimilarity) between the individual objects being clustered. Clustering algorithms may be divided into the following two major categories [ThKo03]: *hierarchical clustering* and *sequential algorithm*. Sequential algorithms produce a single clustering, i.e. a single partition of the data set. An example of such clustering algorithms is the k-means algorithm.

The method developed in this work relies on hierarchical clustering. Its principles are discussed in the following sections.

Hierarchical clustering

Hierarchical clusterings differ from sequential clusterings in that they produce a hierarchy of clusterings instead of a single clustering. Data are not partitioned into a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to *N* clusters each, containing a single object. In fact, hierarchical clustering algorithms are subdivided into two main categories: *agglomerative hierarchical* algorithms, which proceed by series of fusions of the *N* objects into groups, and *divisive hierarchical algorithms*, which separate *n* objects successively into finer groupings.

Any agglomerative clustering is, however, based on the following set of operations: find the

two objects which are closest to each other (clustering criterion), merge them into a single new cluster, and repeat this process until no two objects cannot be further clustered. Thus, hierarchical clustering algorithms produce a hierarchy of nested clusterings. More specifically, for a set of *N* data, these algorithms involve *N*-1 steps and produce *N-1* nested clusterings.

The structure among the clusters is displayed in a dendrogram. It shows the multidimensional distances between objects in a tree-like structure. Objects which are closest to each other in the multidimensional data space are connected by a horizontal line, forming a cluster which can be regarded as a "new" object. The new cluster and the remaining original data are again searched for the closest pair, and so on. The distance of the particular pair of objects (or clusters) is reflected in the height of the horizontal line.

Dendrograms are heavily dependent upon the measure used to calculate the distances between the objects. Figure 5-3 represents an exemplary dendogram obtained from five data.

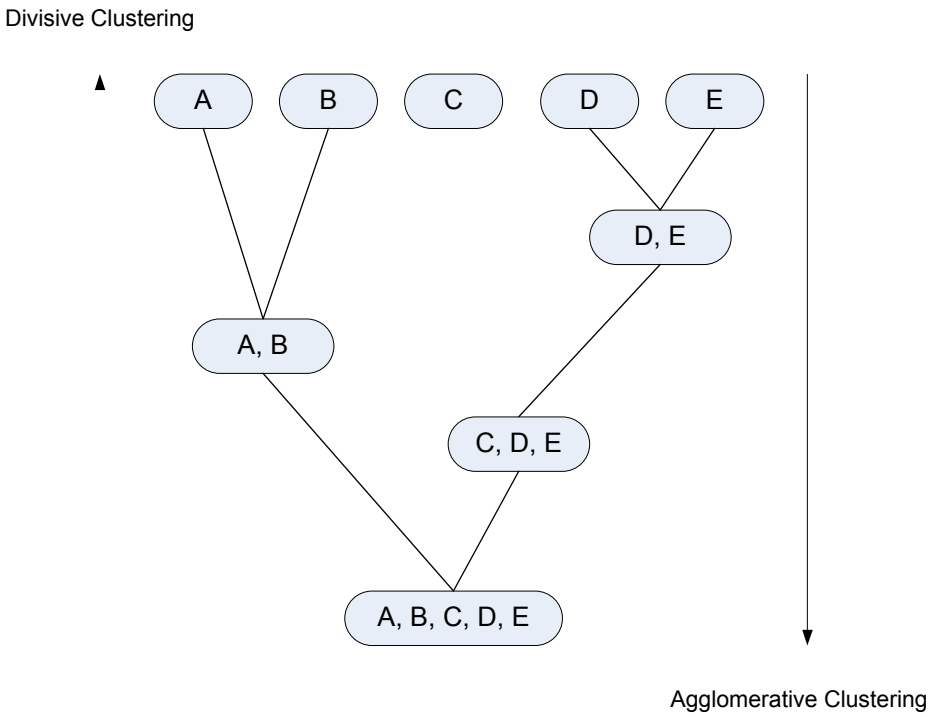


Figure 5-3: Exemplary dendogram obtained from 5 data

Below, we discuss agglomerative clustering in more detail.

Agglomerative methods

Agglomerative clustering algorithm produces a hierarchy of nested clusterings for a data set of *N* elements. At each step *t*, a new clustering is obtained based on the clustering produced at the previous *t-1* step. The initial clustering consists of *N* clusters each containing a single

element of the data set. Next, an additional cluster is produced, containing the most two similar clusters. This procedure continues until the final clustering is obtained, which contains the whole data set. Therefore, two clusters which come together in one cluster at level *t* of the hierarchy will remain in the same cluster for all subsequent clusterings.

The first iteration is quite straightforward, since each initial cluster contains only one single data. Thus, determining what two clusters must be merged together is achieved in computing similarities between pairs and selecting the pair having the greatest result. However, as soon as a cluster contains several data, several methods can be considered. For any of these methods, the distance (or similarity) – $D(Q, C_1 \cup C_2)$ – between any two clusters Q and $(C_1 \cup C_2)$ is expressed by the Lance-Williams formula [LaWi66]:

$$D(Q, C_1 \cup C_2) = \alpha_1 \times D(C_1, Q) + \alpha_2 \times D(C_2, Q) + \beta \times D(C_1, C_2) + \gamma \times \left| D(C_1, Q) - D(C_2, Q) \right|$$

, with:

C_1, C_2, Q three clusters, C_1 and C_2 having been agglomerated into a new cluster $(C_1 \cup C_2)$

$D(C_1, Q)$ is the distance (or similarity) between the cluster C_1 and the cluster Q.

α_1, α_2 , β, γ: the method parameters.

Most types of clusters are single linkage, complete linkage and average linkage. Parameters for each of them are given in the Table 5-1. However, additional methods have been suggested such as median, centroid, ward and the flexible strategy [Thko03].

Type of clusters	**α1**	**α2**	**β**	**γ**
Single linkage	½	½	0	-1/2
Complete linkage	½	½	0	½
Average linkage	½	½	0	0

Table 5-1: Parameters value for different types of clustering

In this work, the single linkage method is employed.

- Single linkage clustering

The defining feature of the single linkage method is that the distance between two clusters is defined as the distance between the closest pair of objects, where only pairs consisting of one object from each group are considered. This is depicted in Figure 5-4.

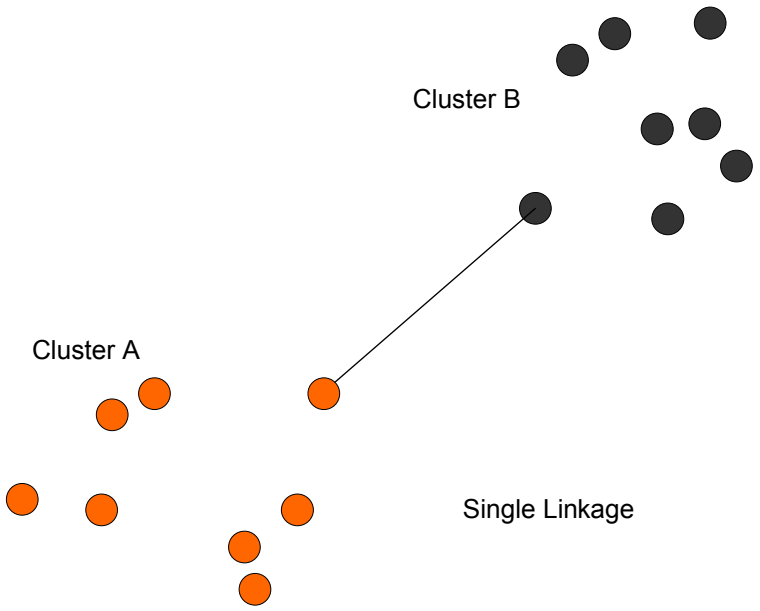


Figure 5-4: Single linkage

Thus, in the single linkage method, $D(C_1, C_2)$ is computed as:

$$D(C_1, C_2) = \min_{(i,j)} D(i, j) \quad \text{(E 5.1)}$$

, where:

i is an object in the cluster C_1

j is an object in the cluster C_2

Thus, the minimum value of the distance between any pair of object *(i, j)* is said to be the distance between clusters C_1 and C_2. In other words, the distance between two clusters is given by the value of the shortest link between the clusters.

At each stage of hierarchical clustering, the clusters C_1 and C_2, for which $D(i, j)$ is minimum, are merged.

5.3 Cluster-based approach

In Section 5.1 the principles of a retrieval function to find similar contexts have been presented. The retrieval function must assess similarities between contexts, based on *user-specific similarity assessment* and *feature-specific similarity assessment*. In addition, an explanation of the current context must be provided to the user, after similar cases have been retrieved and the user preference has been determined.

To address these principles, we introduce a method that clusters cases, in order to determine a more general context that encompasses all problem parts of the clustered cases. This new context is more general since it is solely described by a subset of features. For example, a general context could be “any noisy public place”, in which case, “noise level”

and “ownership” features play a significant role, unlike the other features. Based on the identification of relevant features, a meaningful description of this context - assessed as similar to the current one - can be provided to the user.

The underlying idea is that in similar contexts, a user has a similar preference with regards to an application. Thus, the approach aims to highlight and bring out commonalities in the clustered cases’ representation (features’ values) between cases leading to the same solution (i.e. cases with the same solution part).

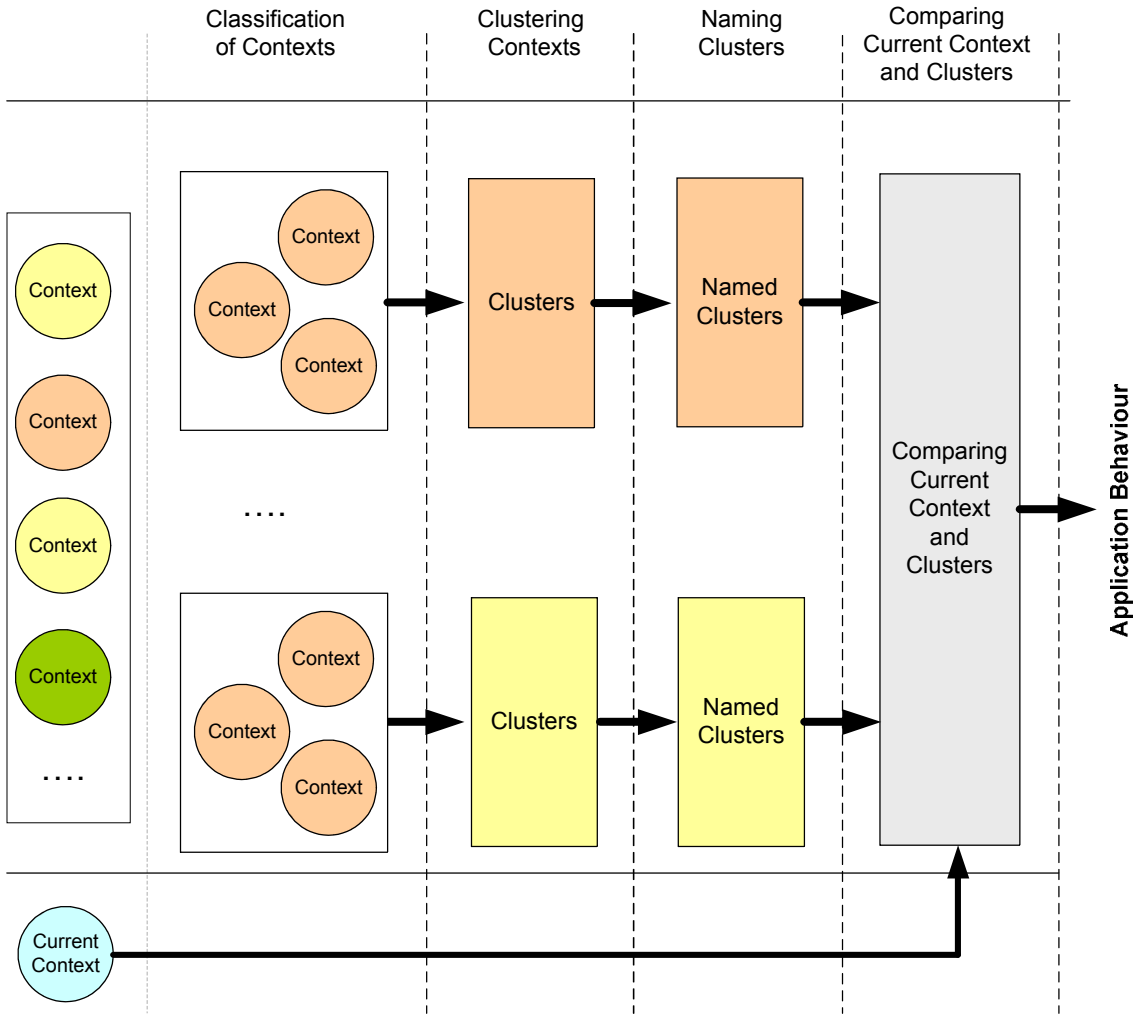


Figure 5-5: Representation of the cluster-based approach

The approach is discussed in detail in the following section. As depicted in Figure 5-5, it is divided in four phases: 1) classifying cases; 2) clustering cases; 3) naming clusters 4) searching for the best cluster.

5.3.1 Classifying cases

The first phase of the retrieval function consists of classifying the different cases that characterise the user’s previous experiences with the given application.

As introduced in Chapter 4, a case is expressed as a problem part and a solution part, as:

$$Case(A,U) = \{< f1^U,..., fn^U >; < Behaviours(A) \mid Parameters(A) >\}$$

With:

- $< f1^U,..., fn^U >$ providing a description of the user's context
- Behaviours(A) specifying the list of application behaviours
- Parameters(A) specifying (when needed) the values of the parameters that are required by the application behaviours. These parameters include either user preferences (e.g. preferred type of music: rock-music), or pieces of user context (e.g. user location).

In the first step of the algorithm, cases are classified according to their solution part: cases with a common solution part are grouped together. An application may have different behaviours, and each of them may require parameters (e.g. user preferences or context information). The categories then correspond to the different configurations of behaviours and parameters depicted by the cases.

When an application provides different behaviours to the users, and none of these behaviours requires any additional parameters, categorising the cases is straightforward: each distinct behaviour, which occurs at least once in the solution part of a case, is a category. For example, if an application provides *n* behaviours, and in previous situations *m* (with m ≤ n) behaviours have been displayed to the user, *m* categories are defined.

Alternatively, one or several behaviours may require additional information (parameters) about the user to match the user's needs and provide a personalised operation. These parameters may also determine categories along which cases are classified. For example, assume an application has one single ability (and as such one single behaviour): it plays music. However, the application can get personalised according to the type of music played, e.g. rock, classical, etc. When several parameters are required for an application behaviour, the different combinations of the parameters' types define categories. Thus, with two parameters having *k* and *l* types respectively, the maximum number of possible categories is $k \times l$.

All cases corresponding to the user's previous experiences are then separated and stored into as many distinct (sub) case bases as there are categories. When being stored, the cases are reproduced as they are. Neither the solution part, nor the context description is changed.

5.3.2 Clustering cases

The second phase of the retrieval function consists of applying an agglomerative clustering to all the solution parts of the cases within each created case base.

By applying an agglomerative clustering algorithm, a tree structure between the contexts is defined. The agglomerative clustering algorithm starts with all contexts defined as independent clusters. Each cluster *contains* one single context. As the algorithms proceeds, the most similar distinct clusters are grouped together. This ends with a unique large cluster, encompassing all initial contexts and the intermediary created clusters. The resulting dendogram presents at an early stage (top level) the most similar clusters grouped together, whereas more dissimilar ones are merged together at a later stage (the method for agglomerating the clusters is described in Section 5.2).

The rational for applying the agglomerative clustering is as follows. The objective is to define contextual states to which the distinct user needs can be associated. A contextual state has a broader scope than a single context, as presented in Chapter 3. In fact, it can be defined by a varying number of features and it can encompass multiple (possibly infinite) single-defined contexts. By clustering together the previous user contexts based on their similarities, we attempt to define these contextual states.

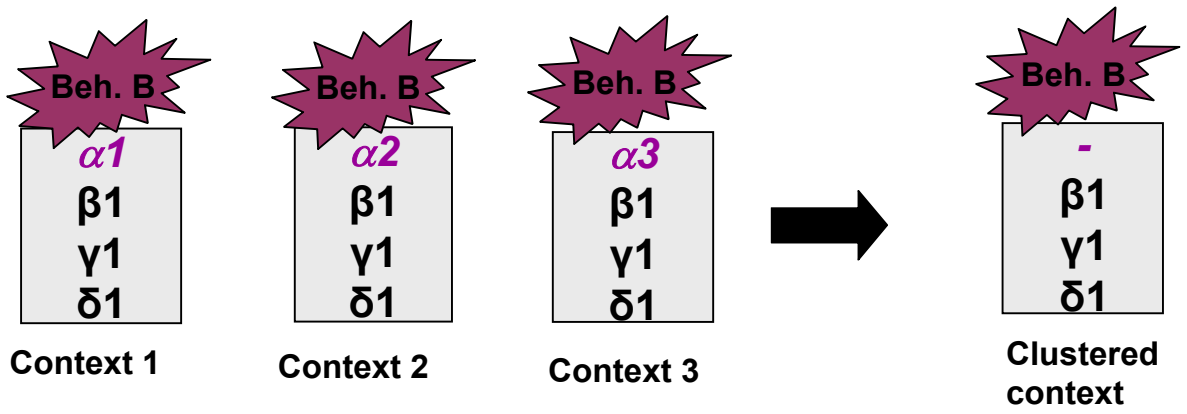


Figure 5-6: Defining a contextual state

Figure 5-6 provides a simple example of a contextual state defined from single contexts. It depicts three problem parts (contexts) of stored cases. All cases have the same solution part (*Beh. B)*, and as such have been classified in the same category in the previous phase. All contexts have the same values for the last three features (respectively β1, γ1, $\delta 1$). However they differ in terms of values for the first feature. If a contextual state is to encompass these three contexts, it can be characterised by last three features (i.e. only these features are deemed relevant to characterise contexts within the contextual state), whereas the first feature is not significant.

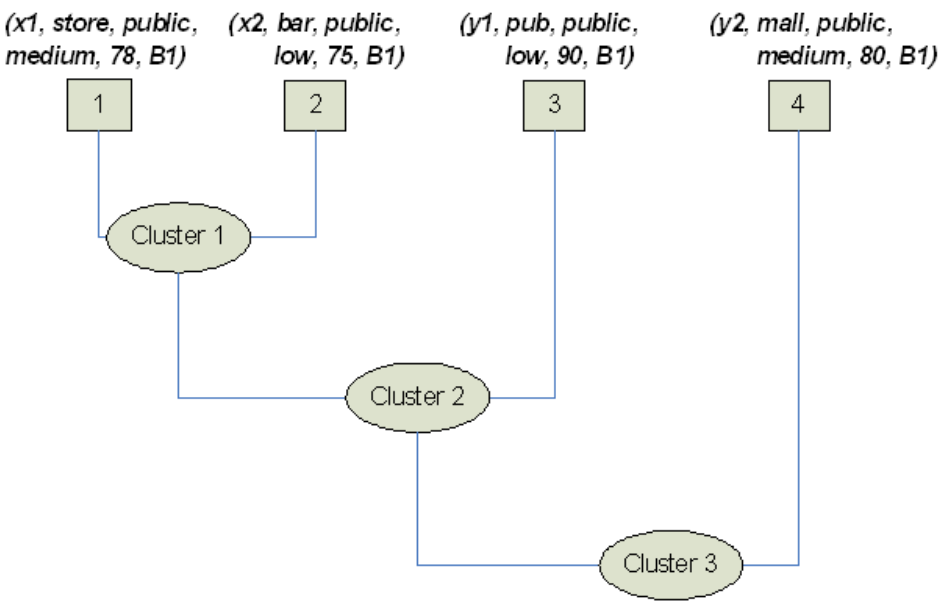


Figure 5-7: Exemplary dendogram obtained for four contexts

An exemplary dendogram for a simple data set of four contexts is represented in Figure 5-7. Initially the four contexts are considered as minimal clusters. The first occurrence of the

The similarity between two clusters C_i and C_j is computed as the average sum of the local similarity metrics for all context features (Equation 5.1).

$$sim(C_i, C_j) = \frac{1}{n} \times \sum_{k=1}^{n} sim_k(f_k^{C_i}, f_k^{C_j}) \quad \text{(E 5.1)}$$

, where:

n, is the number of features, representing any context

$f_k^{C_i}$ and $f_k^{C_j}$ are the values of the k[th] feature for the cluster C_i and C_j respectively

sim_k is the local similarity metric for the k[th] feature.

The Generalized Agglomerative Scheme can be formulated as follows:

- t denotes the current level of hierarchy in the clustering.
- R_t denotes the clustering at level t containing *k* clusters.

1. Initialization
   a. Choose $R_0 = \{C_i = \{x_i\}, i = 1,...,N\}$ as the initial clustering
   b. t=0
2. Repeat until all contexts lie in a single cluster:
   a. t=t+1
   b. among all the possible pairs of clusters (C_r, C_s) in Rt-1

find the one (C_i, C_j) so that:

$$sim(C_i, C_j) = \max_{r,s} g(C_r, C_s)$$

define $C_q = C_i \cup C_j$ and produce the new clustering:

$$R_t = \{R_{t-1}\} - \{C_i, C_j\} \cup \{C_q\}$$

5.3.3 Naming clusters

At this stage, within each case base, dendograms have been defined that provide a structure of the different contexts with regards to their global similarities. In order to define each cluster, values for each feature have to be determined that characterise all the contexts with the cluster. The aim of agglomerating clusters is to define general contexts (i.e. contextual states) from the single defined ones (the cases' problem parts).

Characteristic features' values for each of newly formed clusters are determined as follows:. The value of feature *k* for cluster C*i* is given as follows:

$$f_k^{C_k} = F_k(f_k^{C_{i-1}}, f_k^{C^*_{i-1}}) \text{ if } sim_k(f_k^{C_{i-1}}, f_k^{C^*_{i-1}}) \geq \theta_k \qquad \text{(E 5.2)}$$

$$f_k^{C_k} = null \qquad \text{else}$$

, where: $f_k^{C_i}$ and $f_k^{C_j}$ are the values of the k features for the clusters C_{i-1} and C^*_{i-1} that are merged together into C_i.

θ_k is the threshold for the feature *k*, with value in [0,1]

In fact, as discussed in Chapter 4, features can take nominal, ordinal or interval-scaled values. Ordinal and nominal features take values expressed as ordered or unordered concepts. Interval-scaled values are further distinguished between continuous or discrete range. Hence, according to the types of features characterising representative features for the clusters differs. Thus, $F_K(f_k^{C_{i-1}}, f_k^{C^*_{i-1}})$ is specific to the type of feature k.

$F_k(f_k^{C_{i-1}}, f_k^{C^*_{i-1}}) = f_k^{C_{i-1}}$ for features with nominal and ordinal values (E 5.3)

$$F_k(f_k^{C_{i-1}}, f_k^{C^*_{i-1}}) = \frac{f_k^{C_{i-1}} + f_k^{C^*_{i-1}}}{2} \text{ for features with continuous values}$$

$$F_k(f_k^{C_{i-1}}, f_k^{C^*_{i-1}}) = E\left(\frac{f_k^{C_{i-1}} + f_k^{C^*_{i-1}}}{2}\right) \text{ for features with discrete values}$$

The new feature's value is then the value $f_k^{C_{i-1}}$ of the first cluster C_{i-1}, when the feature takes nominal or ordinal values (e.g. a typical value is "green", "yes", "kitchen").

The new feature's value is the average of $f_k^{C_i}$ and $f_k^{C_j}$, the values of the two clusters C_{i-1} and C_{i-1}^*, when the feature has continuous values (e.g. "1.23").

Finally, when the feature takes only discrete values the *E(.)* function is used over the average of $f_k^{C_i}$ and $f_k^{C_j}$.

5.3.4 Searching for best clusters

The last phase of the retrieval function consists of determining what clusters to which the new context can be best related. Until this point a set of clusters has been determined and for each of them the *F* relevant features have been defined – *F* differing from one cluster to another.

When searching for similar cases (as opposed to exact match) the retrieval phase is performed over the named clusters and not over the single cases. The new context (*K*), expressed as a set of *n* features ($< f1,...fn >$) is compared to all the named clusters. This comparison is performed by computing the similarity between the new context (*K*) and the cluster (C_i), over the *F* non-null features:

$$Sim_{Search}(K, C_i) = \frac{1}{F} \times \sum_{k=1}^{F} sim_k(f_k^K, f_k^{C_i}) \qquad \text{(E 5.4)}$$

The similarity Sim_{Search} is computed as the average sum of the local similarity metrics for the features (f_k), for which the cluster C_i has no null-value. Hence, *F* represents the number of features of the cluster C_i which are non-null.

After having computed all similarity values between the new context and all the clusters (Γ), the new context is assigned to the cluster it presents the greater similarity with (Nearest – neighbour), such as:

$$BestCluster(K) = \max_{C_i \in \Gamma}(Sim_{Search}(K, C_i)) \quad \text{(E 5.5)}$$

The retrieval phase ends with the retrieving of the nearest cluster. In the next step of the adaptation process, the cluster is used to determine the appropriate application behaviour corresponding to the user's preference.

5.4 Conclusion

Retrieving a user's previous experiences having similarity with the user's current context is a critical phase in Contextual Personalisation. These previous experiences will lead to the definition of the solution part for the current problem part (the current user's context). Hence, it is important that the similarity between the experiences or cases is correctly assessed.

The development of the retrieval function is guided by three design principles:

- Similarity assessment between contexts is user-specific

Similarity between contexts can be assessed differently for different users.

- Similarity assessment between contexts is feature-specific

When assessing similarity between different pairs of contexts, the relevant features may differ.

- The results of the retrieval function must be explained to the user

Revealing aspects of context must be displayed to the user to explain the reasons for the previous experiences' selection.

The developed retrieval function addresses these three principles. The method aims to define meaningful and general contexts from several experiences. These general contexts are only characterised by a subset of the features. To do so, the user's previous experiences are first classified according to the application behaviour to which they are related. Second, for each class, experiences are clustered in order to bring out their commonalities in terms of context features' values. The current context is finally compared to all the defined clusters and the most similar cluster is retrieved and presented to the user.

In Chapter 9 a simple evaluation of this retrieval function is presented. The results obtained via a set of simulated user's experiences to the results obtained by using the Nearest Neighbour method are compared.

6 Application-Specific Knowledge

In chapter 4 Copernik was introduced and the principles it calls for to support applications were discussed. One of these principles aims to address the current limitations in terms of user data privacy and deals with the separation between the adaptation decision and applications. Adaptation decision has to be centrally performed based on user contexts and user preferences, preventing the applications from getting knowledge of all user personal data and thereby improving privacy. However, adaptation may not be performed the same way for all contextual personalised applications.

In this chapter, we argue that specific application data drive adaptation for contextual personalised applications. To do so, we review four examples from the literature of applications belonging to the class of Contextual Personalisation. Their differences in the adaptation process are highlighted and are further exploited to characterise three types of data upon which any adaptation process for Contextual Personalisation is based. We refer to these data as application specific knowledge. In order to enable the separation between adaptation decision and applications, each application's specific knowledge has to be delivered to the system implementing the Copernik framework, via templates. The templates for each knowledge type are presented in the subsequent section.

6.1 Adaptation for contextual personalised applications

Users of contextual personalised applications can benefit from systems dealing with context information and adapting the application. However, the adaptation process is not the same for different applications. This becomes evident when considering examples of contextual personalised applications, already available in the literature.

The discussion in this section is organised as follows. Firstly, four examples of contextual personalised applications are presented. For each of these applications, the scenarios they support and their specificities are briefly discussed. Then, a reverse-engineered definition of the application interface is given, assuming that the application is supported by a contextual personalised system. This definition may not correspond to the interface, which has been really implemented. However, it serves to highlight the characteristics required to the adaptation of these applications.

- Context-aware message filtering service [SaGa+05]

One of the applications presented in [SaGa+05] allows a user to specify preferences as to when he wants to see different types of messages based on the context. As discussed in Chapter 3, context encompasses information about the user's activity, the user's location and the weather.

From the described scenario we can, in retrospect, define the following methods in the application interface.

```
// Incoming message is delayed until the user is in a proper context
+ delayMessage()
// Incoming message is delivered to the mobile phone of the user
+ deliverMessage()
```

- CAMS [NaKi+02]

CAMS is a context-aware messaging service. The system selects the most suitable telephone number or email address, and redirects each incoming message or phone call dynamically, according to the user's context. The context is composed of the user's schedule and his location. Adaptation is managed by rules that each user can specify beforehand. The rules specify which means of communication are appropriate under what conditions. The user's schedule is inferred from the user's calendar, while localisation is obtained from the cell phone network.

We can define the following methods in the application interface.

```
// email is redirected to another email address indicated by the
// parameter "Email_Address"
+ redirectEmail(Email_Address)
// call is redirected to the phone indicated by the parameter
// "Phone_Number"
+ redirectPhoneCall(Phone_Number)
```

- The multimedia Infotainer [SaPo+05]

The Infotainer is a mobile application that offers personalised types of news in various forms according to the user's context. For example, the user can get traffic news while driving his car. Since he is focusing on driving, the news is played in a voice format over the car speakers. Alternatively, when the user is at home, sport news can be shown on the television display in the living room. In this application, adaptation is driven by IF-THEN rules, defined by the user or learnt from recorded user behaviour (usage) patterns. The current implementation of the application is sensitive to the user's location only.

Similar to the previous application, one can define the application interface as such:

```
// Application delivers news of type "News_Type" to the user
+ getNewsType(News_Type)
// Application delivers news in a format specified by "News_Format"
+ getNewsIn(News_Format)
```

- Restaurant recommender application [MiKo04]

A restaurant recommender application has been designed for the international airport of Oslo. When the user enters a Bluetooth tag zone “Eating Area”, his context is resolved. Parts of his context (user history) might show that the user has not eaten anything for the last six hours, which might mean he is hungry. The user preferences are further analysed and the recommender application can propose a restaurant in the neighbourhood. In this application, the context is defined by a specific context model, comprising personal, social, but also spatio-temporal and environmental information. The application has been discussed in section 3.2.

The application interface of this application can be represented as:

```
// Application provides restaurant recommendations following the user's
// tastes (Restaurant_Type) and in the specified location (Location)

+  RecommendRestaurant(Restaurant_Type, Location)
```

In some of the current realisations, contextual personalised applications are responsible for handling, i.e. acquiring and processing context and performing user modelling (e.g. [NaKi+02]). In other cases, a system supports diverse applications by providing gathering and processing facilities only (e.g. [MiKo04], [SaPo+05]). However, these systems do not explicitly consider the peculiarities of the applications they support. In fact, this probably results from the fact that both the system and the applications are designed and developed simultaneously by the same people. Moreover, these applications aim to demonstrate the feasibility of the specific system approach which can include, according to the system, a method for handling context, an algorithm for processing context information, the use of ontology-based reasoning, etc. As such, they are not designed to serve any other users than those present in the laboratory. For these reasons, applications are “hard-wired” to these systems.

In the envisaged scenarios of Ubiquitous Computing, however, users may access a large variety of services all over the course of the day. Applications providing these services may be released, on a regular basis by different developer groups just as new services are made available today on the Internet. For a system to support these applications, their characteristics have to be made explicit so as to enable a loose coupling between any application and the system.

It appears that, in order to perform adaptation, i.e. select the right application behaviour, and to provide a context description including the elements to which the application is sensitive, contextual personalised systems must be aware of each application’s characteristics, i.e. application-specific knowledge.

However, the way adaptation is carried out is not identically reproduced in each of them. In

fact, adaptation differs in terms of context information to which the applications are sensitive, adaptation mechanism (e.g. rules vs. case-based reasoning) and the adaptation output, i.e. the behaviour that is triggered by the adaptation process. In the next section, we describe in more detail what types of application-specific knowledge can be defined so that applications are supported by a system implementing the framework's principles.

6.2 Definition of application-specific knowledge

Application-specific knowledge is information that is used by contextual personalised systems to perform the adaptation decision. From the four aforementioned applications, we can define three types of application-specific knowledge.

6.2.1 Behaviour description

The first type of knowledge we identify is the application behaviour description.

In short, adaptation consists of performing an application behaviour that matches the user needs based on some user implicit data. In fact, the adaptation process of an application consists of:

- providing a set of parameters that will govern the reaction of the application;

For example, in the method `RecommendRestaurant(Restaurant_Type, Location)` adaptation requires the provision of a restaurant type (preference) and the user's location (context) parameters. Similarly, in the case of a music player application that has access to a list of songs classified by genre, the system can trigger adaptation by passing as a parameter to the application interface the genre of the songs the user wants to listen to in his current context (e.g. he wants to listen to jazz music at night, but prefers listening to rock music in the morning before going to work).

- and / or selecting one behaviour among a list of offered ones

For example, in [SaGa+05] adaptation is performed by selecting the behaviour that is the most suitable to the user. The selection is made between the operations: `delayMessage(), deliverMessage()`.

Both approaches can be performed jointly, and adaptation can then involve 1) the selection of an operation and 2) the provision of the required parameters. Conversely, the application operations may be performed jointly, thus not requiring a particular operation to be selected. [SaPo+05] illustrates this, as adaptation is enabled by specifying the required type of news (`getNewsType(`*`News_Type`*`)`) and the modality the news provided (`getNewsIn(`*`News_Format`*`)`). If one of these operations is not performed, adaptation does not occur successfully.

The specificities of the four applications with regards to these aspects are presented in Table 6-1. In addition to the selection capability and the parameters that are needed for adaptation

decision, the table presents the relationship between the application behaviours. The relationship is expressed by two logical operations. XOR characterises a logical disjunction between the application operations. Thus, one operation among the set of possible ones is required, which implies a selection process between them. Conversely, AND characterises a logical conjunction between the operations. All of them are required at the same time.

Note that in the course of our review of applications, we have not encountered any application for which a only subset of operations was performed. Hence, in order to limit the scope of the framework to the support of existing contextual personalised applications, this case is not treated here (as indicated in 4.1.3). At this stage, it is worth noting that such a case would not modify the framework presented in Chapter 4; however, it would make the determination of the user preferences in a new context more complex. As any subset of application behaviours could be performed, determining exactly what application behaviours are required and in what order becomes more complex with regards to the growing number of combinations of behaviours. Consider the example of an application with n behaviours. Under the assumption indicated in 4.1.3 and according to the adaptation process of this application (XOR or AND), any user preference is expressed as either one of the n behaviours (XOR) or the set of the n behaviours (AND) with specific parameters (the parameters are determined as in 4.3.6). If a subset of application behaviours is considered, in total $\sum_{i=1}^{n} \frac{n!}{(n-1)!}$ combinations exist for the user preferences (15 if n=3, 64 if n=4, etc.).

Application	Selection	Parameters	Relations between behaviours
CAMS [NaKi+02]	No	Email_Address Phone_Number	-
Multimedia Infotainer [SaPo+05]	No	Type_News News_Format	AND
Context-aware message filtering service [SaGa+05]	Yes	-	XOR
Restaurant recommender [MiKo04]	No	Restaurant_Type Location	-

Table 6-1: Behaviour specificities of the aforementioned applications

It appears then it is compulsory that the system, to carry out adaptation, has prior knowledge of all behaviours and their relationships that the application can provide.

The behaviour description can be expressed in a template in a similar way to the abstract

interface part of a WSDL service description [WSDL]. The template describes the behaviours (operations) supported by the application. To this end, it has to provide the following:

- a way to identify the behaviours, via names;
- the parameters that are required to trigger each behaviour, when needed;
- the event that will cause the behaviour to be triggered, i.e. what event requires adaptation to take place (e.g. incoming call), as well as the produced output(s), i.e. the result(s), of these behaviours (e.g. call is blocked);

Besides, the description has to be completely independent of any communication protocol or concrete data structures. Also, no data type for the inputs / outputs is given in the template because the template is used to infer the adaptation. The call to the interface depends on how the application is implemented (e.g. web service). This is out of scope in this work. The description of a template structure is given in section 6.3.2.

6.2.2 Context features

The second type of knowledge that is specific to any application concerns the context features to which the application reacts.

In contextual personalised systems, information in the user models is stored with respect to the user context in which it is relevant. Then, a description of context provided by the system is used in the adaptation stage to select from a user's model the information governing the adaptation. However, dealing with context is a difficult task, since context has many alternative representations [HeIn+02].

On the one hand, the features that can be part of context are numerous. In Table 6-2, we list features used by the aforementioned applications. Most of them have the feature location in common. But other features vary from one application to another.

Application	Context features	Description[5]
CAMS [NaKi+02]	User's location	Cell phone network (NTT DoCoMo service)
	User's schedule	Calendar application
Multimedia Infotainer [SaPo+05]	User's location	GPS coordinates
Context-aware message filtering service [SaGa+05]	User's location	User's location
	Weather	Weather
	User's activity	Calendar application

[5] Description for [NaKi+02], [SaPo+05] and [SaGa+05] indicates how the information about the features are obtained. In [MiKo04] such information is not available for all features.

Restaurant recommender [MiKo04]	User's task	User's goals and activity
	Social context	people and objects surrounding the user
	Personal context	Preferences
	Time	-
	User's location	Bluetooth cell
	Environmental context	Available applications

Table 6-2: Context features of the four aforementioned applications

On the other hand, the representation of any given context feature varies in terms of types and level of abstraction (granularity).

As shown in the different examples and discussed in Chapter 2, there is no unique representation of context. Thus, a user context can be expressed in different ways, with different context features. Rather context is defined with respect to a specific purpose or utility. For contextual personalised applications, this purpose is strongly related to and characterised by the application-domain. Thus, context features used in the context representation is application-specific knowledge.

To allow contextual personalised systems to properly represent user contexts and permit the selection of information from the user model, the template has to indicate the following:

- the context features the application is sensitive to, via names;
- the type and the level of abstraction of the context features;
- the entity that is related to the context feature. As highlighted in the definition (2.13 – see chapter 2), a context information can be related to any entity and not only the current application user.

Here, the objective is not to determine how the information is obtained from the environment nor is it to list the values the elements can take in the context description at this stage. Rather, the template has to specify in a distinct way what feature is used, including the type, and the level of abstraction (granularity) of the piece of the information. The description of a template structure is given in section 6.3.3.

6.2.3 Adaptation knowledge

The third type of application-specific knowledge is referred to as adaptation knowledge. Adaptation knowledge comprises any piece of information that is used in the retrieval function to infer the adaptation of an application.

In the aforementioned four applications, different reasoning mechanisms are used. [NaKi+02] makes use of user predefined rules, while [SaPo+05] relies on rule learning, and [MiKo04] and [SaGa+05] rely on case-based reasoning. The limitations of rules for adapting to context have been discussed in Chapter 2. This has motivated our proposal for a framework that draws comparisons between previous user experiences based on case-based reasoning

techniques in Chapter 4.

Systems implementing the principles of Copernik must therefore abandon rules to drive the adaptation of contextual personalised applications and rather apply a similarity based approach. In CBR, the similarity assessment is obtained by a similarity metric that compares the new context to the context parts of all stored cases. This metric is application-specific. Following the global/local principle we discussed in section 2.4, such a similarity metric can be decomposed into local ones, each of them assessing commonalities between values of a given feature. These local similarity metrics are then at the basis of the adaptation decision.

Additionally, in the Contextual Personalisation cycle of Chapter 4, we have identified a set of mechanisms different for various applications. These mechanisms must be implemented and deployed by the system supporting the application. Thus, knowledge about what mechanisms are appropriate for the application is part of the application-specific knowledge, as well.

As a consequence, the adaptation template has to indicate:

- the local similarity metrics for each context feature;
- the appropriate feedback mechanism identified by the application developers;
- the appropriate revision mechanism identified by the application developers.

6.3 Templates for expressing application-specific knowledge

In the last section we introduced three types of knowledge required by a contextual personalised system to perform adaptation. As this knowledge is highly application specific and differs from one application to another, it has to be provided to the contextual personalised system so that adaptation can be performed. In this section, we discuss how knowledge is encoded via templates and how the templates are delivered to the contextual personalised systems.

6.3.1 Template mechanism

Application-specific knowledge has to be transferred to contextual personalised systems. To address the requirement, a mechanism has been developed that makes use of templates to transmit application specific knowledge. By utilising this mechanism, knowledge is provided to the contextual personalised system.

- At the first use of the application

The first time the application is used, the application transmits knowledge via templates to the contextual personalised system. The templates for each application-specific knowledge, are further stored locally in the system. The application-specific knowledge can then be retrieved every time the application is accessed.

- As a new template version is available

After the application deployment, new versions of applications can be made available. Currently new versions of applications are motivated by fixing bugs in code and/or making new application features available. As for contextual personalised applications, an update can also involve changes in the adaptation decision. For example, context features the application reacts to can change, new features can enhance the list of available application behaviours, etc. Therefore, when any modification is made by the application developers, one or several templates are updated. Consequently, the contextual personalised system has to be notified of the changes.

6.3.2 Application behaviour template

The application behaviour template provides knowledge about the various behaviours an application can perform. Below is the description of such a template.

```
1     <?xml version="1.0"?>
2     <definitions name="APPLICATION"
3           targetNamespace="http://example.com/APPLICATION.xml"/>
4           <ontology> http://example.com/APPLICATION.owl </ontology>
5           <relation> RELATION_TYPE </relation>
6           <message>
7                 <name="NAME_MESSAGE1"/>
8                 <description="OWL_MESSAGE1"/>
9           </message>
10          <message>
11                <name="NAME_MESSAGE2"/>
12                <description="OWL_MESSAGE2"/>
13          </message>
14          <message>
15                ....
16          </message>
17          <parameter>
18                <name="PARAMETER_NAME1">
19                <description="OWL_NAME1"/>
20          </parameter>
21          < parameter>
22                ....
23          </ parameter>
24          <behaviour>
25                <operation name="BEHAVIOUR1">
26                      <input_message="NAME_MESSAGE1"/>
```

```
27                          <output_message="NAME_MESSAGE2"/>
28                          <parameter ="PARAMETER_NAME1"/>
29                  </operation>
30                  ....
31          </behaviour>
32    </definitions>
```

Line 2: **APPLICATION** indicates the name of the application to which the template refers. The name is determined by the developer.

Line 3: indicates the default target namespace

Line 4: `http://example.com/`**APPLICATION**`.owl` designates the ontology to which this template makes reference. The ontology is discussed in section 6.4.

Line 5: the **relation** element designates the relation between the application behaviours. As illustrated in Table 6.1, the **RELATION_TYPE** takes either the value AND or XOR.

Line 6 to 16: each message of the application is specified with **message** elements. A message is either an event that triggers application's adaptation or a produced output of the adaptation. Each message is characterised by two attributes: a name (Line 7), and an ontology concept (class or attribute) (Line 8), which provides a semantic description of the message (ontology is further discussed in section 6.4).

Line 17 to 20: each parameter required by the different application behaviours is defined by a **parameter** element. A parameter is characterised by its name and an ontology concept (**description**) to indicate the semantic meaning. Such information is helpful to infer the required parameter (i.e. preference) from the list of previous ones. This system capability is, however, not discussed in this work.

Line 24 to 31: the behaviour of an application is defined by the **behaviour** element. Each behaviour is characterised by a name (**operation name**), the event it is triggered by (**input_message**) and the results (**output_message**) it produces. In our approach, the number of events and results is limited to one of each for any application behaviour. Besides, each parameter required by the application behaviour (if any) is defined by a **parameter** element.

6.3.3 Context Features template

The context features template indicates the features an application is sensitive to. As explained in Chapter 2, these features are selected by the application developers.

```
1     <?xml version="1.0"?>
2     <definitions name="CONTEXT_APPLICATION"
3           targetNamespace="http://example.com/CONTEXT_APPLICATION.xml"/>
4           <ontology> http://example.com/CONTEXT.owl </ontology>
5           <context>
6                 <name="FEATURE1"/>
7                 <description="FEATURE1_SEMANTIC"/>
  S.  <target="FEATURE1_TARGET"/>
9           </context>
10          <context>
11                ...
12          </context>
13    </definitions>
```

Line 2: **CONTEXT_APPLICATION** defines the name of the template

Line 3: indicates the default target namespace

Line 4: `http://example.com/`**APPLICATION**`.owl` designates the ontology to which this template makes reference. The ontology is discussed in section 6.4.

Line 5 to 9: a context feature is defined by the **context** element, which has three attributes: the name of the context feature is given by the attribute **name**, a semantic concept the feature refers to is indicated by the attribute **description**, and a **target** attribute specifies the entity to which the context element refers (e.g. location of whom, activity of whom, etc).

6.3.4 Adaptation Knowledge template

Finally, the adaptation template describes the local similarity metrics used in the retrieval phase. This template is closely related to that of the context features, as for each feature relevant to the application, a specific similarity metric has to be defined. The aim of this work is not to define a new way to express metrics for CBR. [CoDo+04] proposes an approach to represent similarity in CBML [CBML]. Currently, values can be expressed as symbolic (enumerated list of possible values), numeric (integer or doubles), Boolean (false and true) values, as well as string and taxonomy (similar to symbolic type except that the possible values are represented with a tree structure). Moreover this approach requires each feature to indicate a relevance weight. However, as discussed in Chapter 5 context features handled in the framework have varying relevance over time. Hence, we impose a restriction when representing each local similarity. The relevance weight for each metric are then set to 1. Finally, note that when expressing such a similarity metric, the list of possible values or the value range has to be defined, either explicitly or implicitly e.g. referring to an existing

taxonomy. This is the reason such values are not indicated in the context features template, as this would be redundant information.

Below, we give an example of how an adaptation template has to be represented.

```
1     <?xml version="1.0"?>
2     <definitions name="SIMILARITY_APPLICATION"
3           targetNamespace="http://example.com/SIMILARITY_APPLICATION.xml"/>
4           <contextTemplate> CONTEXT_APPLICATION </contextTemplate>
5           <similarity name="SIMILARITY_FEATURE1">
6                 <input="FEATURE1"/>
7                 <location="http://example.com/FEATURE1.xml"/>
8           </similarity>
9           ...
10          <feedback> FEEDBACK_MODE </feedback>
11          <revision> REVISION_MODE </revision>
12    </definitions>
```

Line 2: "**SIMILARITY_APPLICATION**" is the name of the template.

Line 3: indicates the default target namespace

Line 4: c**ONTEXT_APPLICATION** is the name of the context template, which is related to the adaptation knowledge template.

Line 5-9: the similarity metric used for any context feature is introduced by a **similarity** element. It is characterised by attributes including, the name of the context feature it refers to (**input**) – as specified in the context feature template - and the description of the similarity metric itself, which is given in an additional file, the location of which is indicated by a URL in the attribute **location**.

Line 10: the feedback element (**feedback**) indicates what feedback mode is the best for the application and has been selected by the developers. As discussed in Chapter 4, there are three possible modes and thus three possible values: *explicit, reward and interruption.*

Line 11: similarly, the revision mode is indicated by a revision element (**revision**), taking three possible values: *user, system, user-supported*.

6.4 Semantic support

When discussing the templates in the previous sections, we have introduced references to two ontologies that provide description of semantic concepts.

An ontology is defined in [DICT] (cited in [Flan05b]) as an explicit formal specification of how

to represent the objects, concepts and other entities that are assumed to exist in some areas of interest and the relationships that hold among them. For AI systems, what “exists” is that which can be represented. When the knowledge about a domain is represented in a declarative language, the set of objects that can be represented is called the *universe of discourse*. We can describe an ontology by defining a set of representational terms. Definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory.

Reasoning and making inferences by means of an ontology and using description logic (DL) can be performed via Tboxes (Terminology) and Aboxes (Assertions) (Figure 6-1), which are part of a knowledge base. Aboxes and Tboxes permit the differentiation between the actors of the domain of interest and the concepts defined in this domain. For example, in a given domain, `Parent` and `Child` are concepts which are formally defined in a Tbox, where as, `John` and `Mary` are items of the domain and are defined in an Abox. As a consequence, reasoning in the domain implies the items of the Abox are mapped into the concepts of the Tbox. In this example, mapping `Mary` into the concept `Parent` and `John` into the concept `Child`, it may be possible to infer additional relations between the two people of the domain.

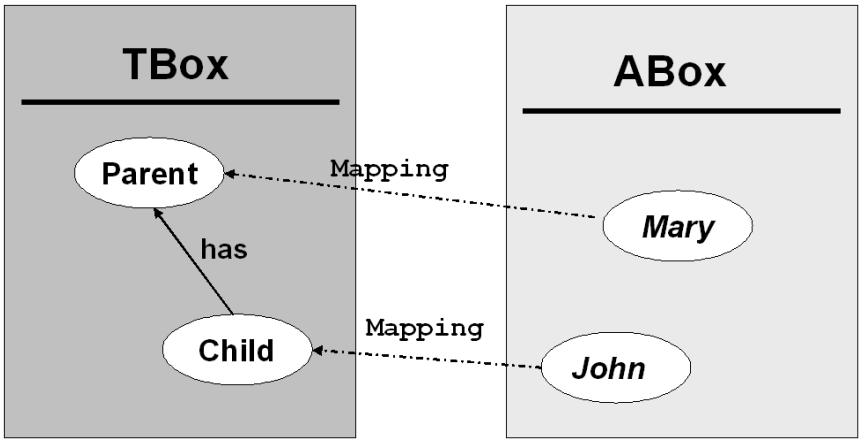


Figure 6-1: Tbox and Abox

An ontology can then be beneficial in a domain comprised of numerous concepts to infer relationships between them. In addition, an ontology can also be helpful in a domain in which multiple entities are involved that deal with the same concepts. Indeed, the overall definitions provided by an ontology can facilitate knowledge comparison and reuse, ensuring that these different entities handle the same concepts. This is surely the main reason for making use of ontologies in the aforementioned application specific knowledge type.

In the templates two different ontologies are required. They are the following:

- Ontology for the application behaviour knowledge

Expressing the behaviour knowledge in a template requires the support of an application

ontology. In any behaviour template, all messages (i.e. events and results) as well as the parameters are characterised by an ontology concept. In order to perform correctly, contextual personalised systems must “know” to which events they are required to react. Since these systems are expected to support various applications, each of them defining one or several event(s) and result(s), a semantic support is needed for the system to monitor these events. Moreover, as discussed in Chapter 4, the central user model is advantageous not only in terms of user privacy, but also because it enables new user preferences to be inferred from previous ones, thus enabling more system autonomy and further preventing users from explicitly keying their preferences. As a consequence, a semantic description for each application parameter is required in order for the system to determine what parameter must be passed over the application interface to trigger the selected behaviour and/or to determine parameters for new applications.

The support provided by this ontology is, however, not further developed in this work, as the algorithms and mechanisms to infer new user preferences are beyond the scope of this study.

- Ontology for the context features knowledge

As already discussed, context information varies in its form and granularity. To identify each feature required by an application, we propose relying on a context ontology. In this ontology, the different ways to consider a piece of context information, i.e. the different features and their relationships are described. For example, location is expressed as a parent concept in the ontology with child-concepts depicting different features for characterising a location, including an exact location (inhabited by a unique entity), a building (in which several entities can be located), a country, etc. The ontology does not, however, specify the technique and the sensors that must be utilised to consequently gather context in the environment. However, a system providing gathering capabilities can take advantage of this information and deploy mechanisms and policies to capture the required context feature’s value (for example, a set of predefined policies could be implemented in the system that indicates for different features what sensors are needed). Though such an issue appears extremely interesting, we have decided not to treat it and have rather concentrated our research on the adaptation process.

As well, the template requires giving information about the target of the context feature. Similarly, as for the feature itself, the context ontology can represent as concepts a set of entities that can characterise the target element. Typical concepts include, device owner (e.g. specified as a subclass of the concept “user”), user’s neighbours, room, or an object.

6.5 Conclusion

Contextual personalised systems traditionally support applications by providing a representation of the user's context. Subsequently, context-aware applications use context to determine how its behaviour has to be adapted.

However, performing adaptation in contextual personalised systems and independent of the applications offers advantages. User data privacy is ensured and a central user model where information is stored in a non-redundant manner can be maintained that can serve several applications.

But, adaptation arises differently for various applications. Applications have their own characteristics that influence adaptation. We refer to these characteristics as application-specific knowledge.

We have reviewed some existing contextual-personalised applications and discussed three types of application-specific knowledge that enable contextual-personalised systems to perform adaptation of applications. We distinguished between: 1) the behaviour description, detailing all the behaviours the application can display, 2) the context features, describing the features along which the application can be adapted and finally 3) the adaptation knowledge, which defines the local similarity metrics used by the case-based reasoning techniques in charge of retrieving the previous cases. We express the knowledge with templates that are communicated to the contextual-personalised systems by the applications.

7 Developing contextual personalised applications

After having introduced our approach to personalising context-aware applications in Chapter 4, detailed a method for retrieving similar past experiences in Chapter 5, and presented the application-specific knowledge in Chapter 6, this chapter deals with the applications in practice. First we discuss the requirements on the applications so that they can take advantage of our framework. Then we present an application, the Call Profile Manager, and we detail its application-specific knowledge.

7.1 Applying the framework

7.1.1 Requirements for applications

Copernik as described in Chapter 4 aims to serve contextual personalised applications. Such applications can be very diverse in nature, being developed for different application domains and dedicated to the mobile or the desktop environment. However, for an application to be supported, it has to address some requirements.

- Context-dependent user profile

The most important requirement is that the application has to be a contextual personalised application, i.e. its personalisation is governed by an individual context-dependent user profile. To illustrate this, we can consider three scenarios of contextual personalised applications:

- Desktop manager

As a user starts his laptop, the personalized context-aware system detects the current context, e.g. where he is, whether in his office or at home, and what activity he is currently performing. Consequently, applications used by the user are launched, e.g. email applications, previous documents, etc. As the context changes, e.g. a colleague enters the office, the applications available on the desktop are updated.

- Music player

As a user starts his music player, a playlist is composed of the songs the user likes the most with respect to his current context. For example, when preparing in the morning he likes to listen to rock music, whereas when getting back home at night he rather prefers jazz music.

- Call Profile Manager

As a call comes in on the user's mobile device, the context is analysed. The call is either blocked, when the user is in such a situation that he does not want to be disturbed, or notified to the user via different ring tones or vibration, with regards to the user's likings for the current context.

- XOR or AND relations between application behaviours

As discussed in Chapter 6, an application can be adapted based on changing user preferences. For example, a restaurant recommender service provides recommendation to its user according to the user's tastes and location. However, adaptation for some applications can also be carried out by determining and performing a relevant behaviour from a list of available ones. In this work, we consider – as underlined in Chapter 4 – either applications the behaviours of which are performed concurrently (characterised by AND in their behaviour templates) or applications for which only one single behaviour is selected and triggered (XOR). We do not consider e.g. applications for which a subset of behaviours would be triggered.

- Development process

Finally, as adaptation requires some application-specific knowledge it is necessary for the application to be developed following a contextual personalised development process. By going through the different stages of the development process, an application designer ensures that all required information has been defined and is being made available to the adaptation system. This information includes the description of the application behaviours, the feedback and revision modes, the context features, etc. The development process is discussed in the next section.

7.1.2 Contextual personalisation development process

We discuss the different phases that are taken for the development of any contextual personalised application. It is worth noting that such a process differs mainly from the one of traditional applications in that additional information driving adaptation, i.e. application specific knowledge, is required. The whole process is depicted in Figure 7.1 in a way similar to the Waterfall process [Royc70]. The different phases that constitute the contextual personalised process are described as follows:

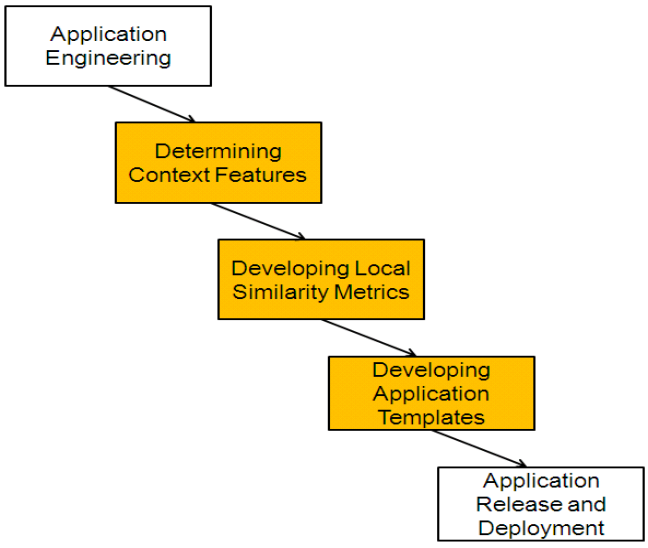


Figure 7-1 Development process for contextual personalised applications

The process highlights the different steps to be taken to develop a contextual personalised application. The coloured blocks designate the steps specifically dealing with application-specific knowledge.

- Application engineering

The application engineering phase is the first step to be taken when developing a contextual personalised application. It corresponds to the first traditional activities and associated results that participate in the production of any traditional software product. Traditionally, one distinguishes for all software processes between three fundamental process activities: [Somm07] 1) software specification, where customers and engineers define the software to be produced and the constraints on its operation, 2) software development, where a structure is defined, and the software is designed and programmed, 3) software validation, where tests on the software are performed to ensure that it fits the customer requirements.

A fourth activity is usually defined as part of the software process: software evolution, which concerns the modifications on the software to e.g. adapt to changing customer and market requirements. This activity occurs throughout the lifetime of the application. As such, it is not specifically part of the application engineering phase we present here, but rather can be seen as a parallel activity to the subsequent phases. Indeed, software evolution, e.g. update, can occur at the same time as the determination of context features or the local similarity metrics, the application template development or the application deployment, when new requirements are identified.

Different types of software systems need different types of development processes [Somm07], involving a refinement of the specification, development and validation activities in some finer-grained tasks. Besides, these generic activities may be organised in different ways. This is, however, specific to the application itself and therefore a more detailed process is not discussed here.

This application engineering phase ends with an application having been designed and programmed following the guidelines and principles in use in the software engineering field. As a consequence of the application engineering, the application behaviours are defined. Indeed, as the software product is designed, several functionalities that it offers to its users are investigated and documented (e.g. via UML use cases). Thus the first application specific knowledge type (i.e. the application behaviours) is already known at this stage of the process.

- Determining context features

The next phase in the context personalization process consists of identifying the context features for the application. This set of context features governs the application adaptation. As discussed in Chapter 2, it seems impossible to list all types of context features. Thus, the application designer faces a great and possibly infinite number of possibilities for defining the set of context features. Thus, a methodological approach is required at this stage to come up

with a selection of features.

The selection of a set of context features – possibly ranging from 1 or 2 to several tenths or more – can be performed via several methods:

- Scenario driven and requirements analysis

The set of context features is determined from the analysis of use cases or typical scenarios covered by the applications. By investigating the application scenarios, the development team can define what features are relevant for the adaptation process.

- User tests in laboratory settings

In order to define a set of features, user tests can be performed in laboratory settings. The application can be run in many different test scenarios by a pool of test users. In these tests, the application reactions (e.g. behaviours displayed automatically) are compared with the user expectations obtained from user interviews. This way, one can, for example, narrow down a very large set of candidate features to a list of meaningful ones, in order to have the two set of results match each other as much as possible.

- Online learning techniques

Alternatively, measurements can be performed in real situations by users, instead of by performing user test in laboratory settings. This permits the personalisation of context features, as the set of features may differ from one user to another. However, this requires learning techniques to be developed to determine the set of features from these situations, e.g. narrowing down a large set of candidate features again. The selection could be governed by considering a trade-off between the accuracy obtained with the set of selected features and the "cost" of gathering values for the features. Indeed, it is not equally easy to collect values for all features in the context-aware environment. The cost of gathering values for a feature may largely depend on several aspects such as the availability of required sensors or software in the environment, the available processing power on the platform (e.g. mobile phone), etc. For example, one can expect that the cost of gathering a room temperature value is lower when the mobile device is equipped with a thermometer than when it is not. Thus, if it appears unlikely that values for some features are not able to be read from the environment, a wise decision would be not to select those features, resulting in a potentially less accurate adaptation of the application. Such an issue is a research question on its own and falls out of the scope of this research work.

- Developing local similarity metrics

At this stage of the process, both the application behaviours and the context features have been identified and defined. The remaining knowledge type is the knowledge related to the personalisation process.

The definition of a similarity metric is mostly a creative process requiring a certain understanding of the application domain. Hence, usually only domain experts (e.g.

application developers) are able to provide the knowledge to be encoded into the similarity measure [Stah03]. This means the domain experts using their experience and comprehension of the domain generally write down the metrics as such. The metrics thus do not result from a theoretical process or proved methodology. Although, some approaches to "learn similarity metrics" have been developed. However, these approaches do not permit the modelling of the knowledge represented by the metric. Rather, they aim to learn feature weights (e.g. [WeAh95], [JaCr+00], [Stah03]) to assess the relative importance of each feature.

- Developing Application Templates

Once each type of application specific knowledge has been defined for the contextual personalised application, it is time to express them in templates following the basic forms discussed in Chapter 6. An example of templates is given in section 7.2.5. Templates must be further made available to any adaptation system by the application. Besides, ontologies, which are referred to by the templates, as well as the local similarity metrics, must also be made available. This is, however, not discussed further in this work.

- Application release and deployment

In the next phase of the process, the newly created contextual personalised application can be released. According to some design decisions, the application can run on a server and be accessed remotely (e.g. Web services) or be deployed on a local computing platform (e.g. mobile device, desktop). This phase completes the context personalisation development process. Of course, several iterations of the process may be required to address and cope with the further evolution of the application.

7.2 The Call Profile Manager application

In the previous section, the specificities of applications supported by the Copernik framework have been presented and the different development stages such applications have to go through have been discussed.

In order to demonstrate the feasibility and the benefits of the work in Chapter 8 and 9, an exemplary contextual personalised application has been developed: the Call Profile Manager application. This section describes the application. In addition, it presents the specific knowledge of the Call Profile Manager and introduces its related templates.

7.2.1 Scenario

The Call Profile Manager application aims to support the following scenario.

> *Paul is a researcher and he is currently involved in many projects. Therefore he receives many phone calls every day from different people on his mobile device. Mobile phones have facilitated people's communication means, enabling people to be reached easily. But this also has a serious drawback: a phone call can interrupt and disturb the phone's owner at any time. To avoid such an inconvenience, Paul has started using the Call Profile Manager application.*
>
> *Paul has been using the Call Profile Manager for some weeks now and this afternoon he has an important meeting at the university. As he arrives in the morning, he decides to meet up with a colleague to put the final touches on the presentation he will give in the afternoon. He joins his colleague in his office and together they work on the presentation. When someone tries to reach him few minutes later, the call is block and switched to Paul's mailbox. Indeed, Paul does not wish to accept calls when he is in a colleague's office. On the way back to his office, in the corridor of the department, he is notified about the blocked call. In the corridor, Paul wants to be reached again. During the day, the Call Profile Manager application continues working. At noon he receives another call in the cafeteria. While Paul is in the meeting room, the two callers that tried to reach him received a message asking them to try again later.*
>
> *In the evening Paul goes out to town and continues using the service. In the tram, he receives a call from a friend to fix an appointment. His phone has been switched to the vibration mode, since Paul usually does not want to disturb people around him when using public transportation. Later, while Paul is having a drink with friends in a pub, he is notified of two incoming calls by a louder ring tone. As the evening ends, a friend offers Paul to drive him back home. In the car, Sarah, Paul's friend, attempts to join him and Paul is made aware of the call as his phone vibrates.*
>
> *Conversely the next day, an incoming call is blocked while Paul is driving. Indeed, Paul does not want to be disturbed in this very situation.*

The Call Profile Manager aims to alleviate the limitations of such scenarios in managing on behalf of the user the mobile phone reactions to an incoming call, based on the user's contexts and his related preferences.

7.2.2 Application behaviours

From the above scenario, we can determine the behaviours of the Call Profile Manager application. They are summarised in Table 7-1.

Behaviour name	Description
Normal ring tone	Notifying any incoming call through a user pre-selected ring tone
Loud ring tone	Notifying any incoming call through a user pre-selected loud ring tone.
Call Block	Blocking any incoming call and redirection to the mailbox.
Vibration mode	Notifying any incoming call through vibrations.

Table 7-1: Behaviours of the Call Profile Manager application

In the Call Profile Manager application, the behaviours are exclusive disjunctive, i.e. only one of the application behaviours is triggered at a time, unlike in other applications such as in e.g. [NaKi+02]. Moreover, no behaviour requires any additional specific parameter. Therefore, the aim of the adaptation process for the Call Profile Manager consists of determining based on the user's preferences, the application behaviour that is currently required from the list of available ones.

7.2.3 Context features

The aforementioned scenario indicates that the application primarily reacts to changes in the user's location. Indeed, the user's preferences with regards to the application differ when the user is in a pub, in a meeting room, etc.

As suggested in [Dobs05], when we speak of location we typically mean determining where some person or artefact is located in the real world. The question of determining a location primarily seems trivial. But when looking closely at it, it can appear very subtle, as there are a huge number of possible answers, each of them revealing something about the way we conceptualise location and any application based upon it. For example, Dobson [Dobs05] presents several ways to reckon with location:

- Absolute location

Location can be characterised by coordinates within a location system. Location models are required in order to provide notions about distances and ranges.

- Structured naming

Locations can also be characterised by names assigned to them, when they are perceived to have a unique identity. For example, a room number in a building (e.g. room 1234) or a house number in a street can provide clues to identify a location. In a more general way, names are often associated to locations, e.g. kitchen, conference room, etc. It permits to

identify a class of spaces rather than a space itself, but can be completed by additional information, e.g. building name to alleviate the uncertainty of the location.

- Relative naming

Relative naming serves to identify a space by its relationship to some object or space, such as in his office, in his car, in Mike's office, next to the pillar, with Mary, etc.

- Approximate answers

Sometimes imprecise information can serve to identify some locations. Though this introduces uncertainty when being used in localisation system, such information can still be helpful. Examples of imprecise information are: future or past locations (at 10am he will be in…, at 8 am she was at …), vicinity (he is within a range of …), on path (between Liverpool and Manchester), etc.

- Task-based answers

Another form of location is task-based. Someone's location is defined by capturing the task in which he is engaged. Tasks are often related to locations. For example, the kitchen is dedicated to activities such as cooking or eating.

- Negative answers

To complete the list, one can also mention locations that are indicated by a narrowed down list where some of them are removed, such as in the statement, "Mike is neither at home nor in his office."

Thus, information from different types can be given to characterise a location. Hence, there is no single canonical representation of location that will be uniformly useful for all possible location-based applications. Therefore, when dealing with location information for a contextual personalised application, the kind of location information to provide to the computing environment is important.

In the following we present the context features that have been selected for the Call Profile Manager application. These features are mainly related to the user's location. In section 7.1.2, we discuss several approaches for determining context features. In order to ease the application development, we rely on the analysis of scenarios and requirements to come up with a list of acceptable features.

For each feature, we discuss what it corresponds to and we describe how it is represented and handled by the application. In order to introduce each feature, a use case is presented that depicts different situations of Paul, an imaginary user, in which he requires a specific application behaviour. Each of these situations is described by the single feature only (e.g. *Paul in an office* as opposed to *Paul in the corridor, next door*, depict two situations where Paul's situation is characterised by Paul's geographical position in the space).

Position feature

Paul's situation	Paul's preferred application behaviour
Paul in an office	Block call
Paul in the corridor, next door	Notification with loud ring tone
Paul in the secretary next to the corridor	Notification with vibration

Table 7-2: Use case for the position feature

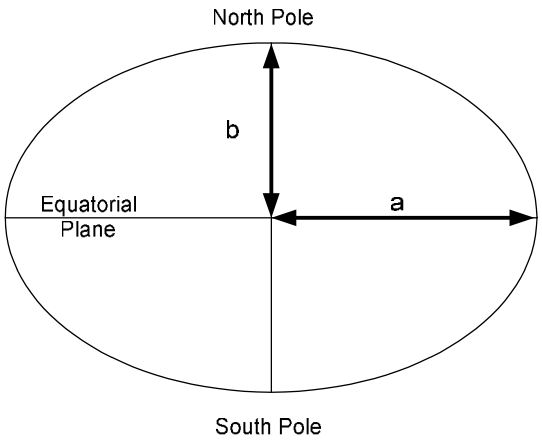
The position of the location indicates the geographical situation (place) of the location. It is characterised by the location's coordinates defined in a coordinate system. A location model is then required in order to provide notions about ranges and distances. These notions can be presented in different formats. [BeDu05] identifies two basic classes of coordinates:

- Geometric coordinates define positions in the form of coordinate tuples relative to a reference coordinate system. A further distinction can be made between global geometric coordinates, defining positions anywhere on the planet, and local geometric coordinates, defining positions locally, e.g. in a room or a building. Through geometric operations, it is then possible to calculate distances between geometrically defined positions.
- Symbolic coordinates define positions in the form of abstract symbols, e.g. room and street names. In contrast to geometric coordinates, the distance between two symbolic coordinates is not defined.

The objective in selecting the position feature is to determine locations that are geographically proximate, though not sharing any commonalities for the other remaining features (detailed hereafter). Then the selection of geometric coordinates to define positions seems the most appropriate, since it enables the measurement of the distance between two locations. Moreover, the envisioned application scenario does not limit the use of the application to one single or a set of locations. Rather, the Call Profile Manager can be used anywhere. This calls for global geometric coordinates to characterise positions.

Today, one well-accepted and widespread coordinate system, because of its utilisation in Global Positioning Systems (GPS), is the World Geodetic System 1984 [WGS84]. It defines a reference frame for earth that models earth as an ellipsoid, with its origin at the earth's centre of mass. Since it was launched in 1984, the reference model was enhanced at different occasions.

The WGS84 uses a reference ellipsoid with the characteristics depicted in Figure 7.2.



a = Semi Major Axis,
b = Semi Minor Axis

	Distance (m)
a	6 378 137
b	≈ 6 356 752

Figure 7-2: WGS84 Reference Ellipsoid [WGS84]

Coordinates in the WGS84 are expressed as triples containing the geographic longitude, latitude and the elevation above sea level. Thus a position feature (of a user *A*) is formally expressed as:

$$f_1^A = \{lat^A, long^A, elev^A\}$$

, where:

lat^A is the latitude, i.e. the north-south geographic coordinate. Latitude is given as an angular measurement ranging from 0° at the Equator (low latitude) to 90° at the poles (high latitude) (90° N for the North Pole or 90° S for the South Pole).

$long^A$ is the longitude, i.e. the east-west geographic coordinate. Longitude is given as an angular measurement ranging from 0° at the Prime Meridian to +180° eastward (E) and −180° westward (W). The zero point longitude is the Greenwich meridian.

$elev^A$ is the elevation, i.e. the height above the mean sea level. Elevation is measured in meter.

Latitudes and longitudes can be expressed in different unit types, e.g.:

- Degrees (°), Minutes (‘), Seconds (‘’) (e.g. W130°30’5’’)
- Decimal degrees (39.909°)
- GPS coordinates (39° 54.33’)

In this work, latitudes and longitudes are expressed in decimal degrees. The position feature does not consider the elevation. Indeed, the contexts considered in Chapter 9 do not require a distinction by their altitude (i.e. all simulated proximate contexts are at the altitude). This is also motivated by the fact that the internet service [GPSV] used to gather the coordinates of the different addresses does not provide such an indication.

As an example, a position feature is expressed hereafter as:

$$f_1^A = \{51.313020°, \quad 9.459164° \quad \}$$

It is worth noting that at this stage, a format for expressing the position feature is selected. However, it does not impose any specific positioning system technology.

Function feature

We refer to the next selected feature as function. Table 7-3 lists for the imaginary user, Paul, some exemplary situations characterised by this feature (and this feature only), as well as his preferred application behaviour.

Paul's situation	**Paul's preferred application behaviour**
Paul in a meeting room	Block call
Paul in a living room	Notification with normal ring tone
Paul in an office	Notification with vibration

Table 7-3: Use case for the function feature

The function feature refers to the purpose for which the location exists and is used by the user. In a nutshell, it answers the question: "What is the location for?" As such, the function is described by the name of the location. For example, the site where I work is my *office*, while the site where I sleep is my *bedroom*.

The function feature belongs to the nominal category, since it takes values from different states for which no meaningful ordering can be found. For example, a function feature can have values such as: *bedroom, corridor, building*, etc. Any term designating a location can be used, provided it is an English term.

Ownership feature

Table 7-4 depicts situations characterised by the feature *ownership* and the associated preferred application behaviours.

Paul's situation	Paul's preferred application behaviour
Paul in his boss's office	Block call
Paul at home	Notification with normal ring tone
Paul at a friend's home	Notification with vibration

Table 7-4: Use case for the ownership feature

The ownership feature characterises the individual that is used to occupying the location. For example, John's office is "owned" by John since it is the office to which he has been assigned. The ownership feature can have a great influence on the choice of application behaviour. Indeed, the user may act differently whether he is in his office, the office of one of his colleagues, or the office of someone with whom he has no social relationships. This feature is a personal indication. Indeed, a customer can consider a restaurant as a public place, while the waiter considers it as private (the place where he works).

Though no study on users' perception of ownership could be found in the literature on context awareness, several studies have investigated individuals' willingness to share personal information.

In [OlGr+05], a survey was performed on thirty people in order to determine what information they are willing to share and with whom. In total, 40 types of information were selected, including current location, cell phone number, social security number, health status, etc. The authors found that people differ in terms of their willingness to share and with whom they are sharing the information. In addition, most participants tend to cluster people they wish to share information with into a similar set of categories. These categories were labelled by the authors as follows: *public, co-workers, manager and trusted co-worker, family, spouse*.

Another survey is detailed in [KhCo06] and investigates privacy preferences and sharing patterns of context information, thus seeming particularly relevant to the Call Profile Manager. The study was conducted on 20 participants who were asked what kind of context information they were willing to disclose to potential callers from six different types of social relations. Among other findings, three main clusters of social relations were found. The first cluster comprises *friends, family members as well as significant others*; the second cluster contains *colleagues and boss*, whereas the third cluster contains *unknown people*.

These two surveys give hints about users' perception on and categorisation of social relations. We interpret these results as follows: The more social interactions an individual has with others, the more social trust he has towards them, thus being willing to share personal information (preference or context).

From these results, we determine a categorisation for the ownership feature. We define the following classes: public (or unknown people), colleague (or co-workers), manager (or boss), friend, and family. In addition, we add a class, private, which was not present in the two studies, since their purpose was on the sharing of personal preferences and context information.

The list of values for the ownership is given in Table 7-5.

Ownership classes	**Description**
Public	Default class
Manager	Any individual working with the user and ranked higher
Colleague	Any individual working with the user, with the same rank or below
Friend	Any individual
Family	Family members including parents, children, cousins, aunts, and uncles.
Private	User (him-/herself) and partner

Table 7-5: Classes for Ownership Feature

Attention feature

Table 7-6 depicts situations characterised by the feature: attention and the associated preferred application behaviours.

Paul's situation	**Paul's preferred application behaviour**
Paul driving a car	Block call
Paul sitting in the car	Notify with normal ring tone

Table 7-6: Use case for the attention feature

Attention determines how much attention the user can give to an application in the current location. This dimension is particularly relevant for situations where interactions with an application must be restricted, since it may e.g. endanger the user.

In this work, we treat attention as an ordinal feature and we define three categories: *low, medium and high*.

Noise level

Finally, Table 7-7 depicts situations characterised by the feature: noise level and the associated preferred application behaviours.

Paul's situation	**Paul's preferred application behaviour**
Paul in a bar	Notify with loud ring tone
Paul in the library	Notify through vibration

Table 7-7: Use case for the noise level feature

The noise level feature characterises the acoustic noise level measured in the close vicinity (i.e. current location) of the user.

Table 7-8 gives some typical values of noise level.

Noise source	**Noise level**	**Noise in dB**
Silence	Threshold of hearing	0
Human breath	Quietest audible sound for persons under normal conditions	10
Whisper, rustling leaves	-	20
Quiet home	-	40
Quiet office, average home	Moderate	50
Voice conversation (1m), average office, hair dryer	Loud	60
Noisy office, inside automobile	Loud	70
Loud conversation	Fatigue and disturbance level	65
Horn	Long exposure can cause hearing loss	85
Discotheque	-	95
Siren at 30 m	Deafening, human pain limit	120
Military jet taking-off	Short exposure can cause hearing loss	150

Table 7-8: Some typical noise level values in dB

The noise level for the location corresponds to the sound level which can be measured in the location. As any noise, it is measured in decibels (dB), which expresses the magnitude of the

sound power relative to a specified or implied reference level[6].

It is therefore convenient to express the sound pressure as a logarithmic decibel scale related to this lowest human hearable sound – 2 10^{-5} Pa.

The Sound Pressure Level:

$$SP = 10 \cdot \log(\frac{P^2}{P_{\mathrm{Re}f}^2}) = 20 \cdot \log(\frac{P}{P_{\mathrm{Re}f}}) \qquad \text{E 7.1}$$

where,

SP is the sound pressure level in dB;

P is the sound pressure in Pa;

and P_{Ref}= 2 10^{-5} is the reference sound pressure in Pascal (Pa).

Note that when the sound pressure doubles, the sound pressure level is increased with 6 dB – i.e. 20 log (2).

Table 7-9 summarises the context features selected for the Call Profile Manager application, and their description.

Context Feature	Description	Location class as defined in [Dobs05]
Position	Geographical position in the WGS84 coordinate system	Absolute location
Function	Purpose of the location	Relative naming and task-based answers
Ownership	Individual associated to the location	Relative naming
Attention	Attention level required by the application user when being in the location	*Not discussed by [Dobs05]*
Noise level	Noise level in the location	*Not discussed by [Dobs05]*

Table 7-9: Context features selected for the Call Profile Manager Application

7.2.4 Local similarity metric

At this stage, the context features relevant for the Call Profile Manager application have been defined. The next step consists of designing similarity metrics for each of them. As discussed in Chapter 2 the features' metrics are called local similarity metrics, as opposed to the global

[6] The Sound Pressure is the force (N) of sound on a surface area (m^2) perpendicular to the direction of the sound. The SI-units for the Sound Pressure are N/m^2 or Pa. The lowest sound pressure possible to hear is approximately 2 10^{-5} Pa.

similarity metrics, which assess similarity between two context instances.

The local similarity metric for the i^{th} feature (sim_i) is a function from D_i x D_i to [0, 1]. Formally:

$$sim_i : D_i \ x \ D_i \rightarrow [0,1]$$

In the previous section, we have described for each feature the domain of definition D_i. In the following the local similarity functions for each of the features are formally expressed.

Position feature (f_1)

The similarity for the feature position is computed from the physical distance between two locations. Computing this similarity is particularly relevant when locations are relatively close to each other. The increase of the distance beyond a certain value does not make locations more different from each other. Therefore we define that beyond a distance D_{MAX}, the similarity is set to 0.

The physical distance between two locations is traditionally computed with Euclidian distance function. Thus, the similarity metric for the position dimension is formally expressed as:

$$sim_1(f_1^A, f_1^B) = \begin{cases} 1 - \dfrac{d(f_1^A, f_1^B)}{D_{Max}} & d(f_1^A, f_1^B) \le D_{Max} \\ 0 & d(f_1^A, f_1^B) > D_{Max} \end{cases} \qquad \text{E7.2}$$

, where: f_1^A and f_1^B are the position features of location A and location B (each feature has a domain D_1 defined as the range of the pair *{lat, long}* defined page 153), $d(f_1^A, f_1^B)$ is the Euclidian distance between the two features and D_{MAX} the distance beyond which the similarity is set to 0.

This function is specially designed to provide similarity values between [0, 1], as depicted in Figure 7-3.

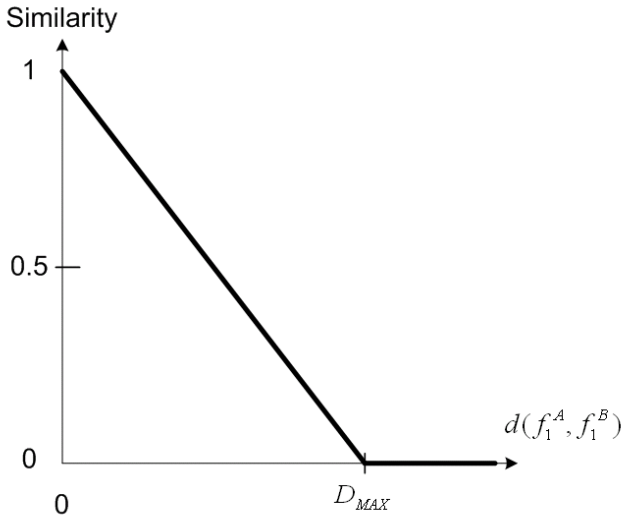


Figure 7-3: Similarity for the feature position

Since the position features are expressed in an ellipsoidal referential, the Euclidian distance is not as straightforward to compute as in an Euclidian referential. Formally the Euclidian distance in an ellipsoidal referential is obtained as:

$$d(f_1^A, f_1^B) = r \cdot \left[2 \cdot A \cdot \tan 2 \cdot (\sqrt{A}, \sqrt{1-A})\right] \qquad \text{E 7.3}$$

with: r, the radius of earth

with $A = \frac{1}{2}\left[\sin^2(\frac{\Delta lat}{2}) + \cos(lat^A) \cdot \cos(lat^B) \cdot \sin^2(\frac{\Delta long}{2})\right]$

with $\begin{cases} \Delta lat = lat^A - lat^B \\ \Delta long = long^A - long^B \end{cases}$

Function feature (f_2)

The similarity between the function features depends on how closely related and interchangeable the concepts are. For example, we can intuitively assess a greater measure of similarity between the functions *office* and *secretariat* than between the functions *office* and *corridor*. It appears then that this similarity is strongly related to the semantic of the concepts.

As a consequence, in order to measure the similarity between the function features, we need to assess how these two features are semantically related to each other. Evaluating semantic relatedness is a problem with a long history in artificial intelligence and psychology [Quil68], [CoLo75]. In particular, semantic similarity represents a special case of semantic relatedness: for example, *car* and *gasoline* would seem to be more closely related than *bicycles* and *cars* but the latter pair is certainly more similar [Resn99].

One can distinguish between different approaches to assess semantic similarities. These

approaches rely on the hierarchical structure of ontology and taxonomy. As discussed in Chapter 6, a taxonomy provides a classification of terms (referred to as concepts) in a tree structure. At the top of the structure is a single concept, the root node that is applied to all concepts in the taxonomy. The concepts are organised and displayed via parent-child relationships. Thus the only link between two concepts in a taxonomy is the link IS-A. Some taxonomies can include single children with multi-parents, for example, “car” might appear with both parents, “vehicle” and “steel mechanisms”. Conversely, the term ontology designates a somehow broader structure. Originally, ontology seeks to describe the basic categories and relationships or relationships of concepts within a given domain. Concepts in an ontology are thus arranged not only with IS-A link, but other relationships such as IS-PART-OF, IS-SIMILAR-TO can be defined.

[BeKa+05] gives an overview of the different approaches to similarity measures based on ontologies. In the following section we present the directions the most relevant to our work.

The most intuitive similarity measure of two objects in an ontology is their distance within this ontology. For example, *eagles* are more similar to *geese* than to *whales*. They would hence reside closer in a typical biological ontology. The calculation of the distance is here based on the specialisation graph of objects in an ontology. The distance is defined as the shortest path going through a common ancestor. The problem of such a distance is that it is highly dependent on the construction of the ontology and therefore on the decisions made when building the ontology. Besides, this approach relies on the notion that links in the ontology represent uniform distance, i.e. all links in an ontology are equally important to measure similarity. In real taxonomies, there is a wide variability of the “distance” covered by a single link [Resn99].

Another approach relies on information theory [Resn99]. The approach exploits the intuitive notion that *similarity for two objects is characterised by the extent to which they share information*. This can be highlighted in the following example. Consider an exemplary ontology (Figure 7-4) where the concepts *nickel* and *dime* are both directly subsumed by *coin*. Also the most specific subclass that *nickel* and *credit card* share is *medium of exchange* ranked higher in the ontology. Then, the similarity of the former pair (nickel, dime) will be greater than the one of the latter (nickel, credit card).

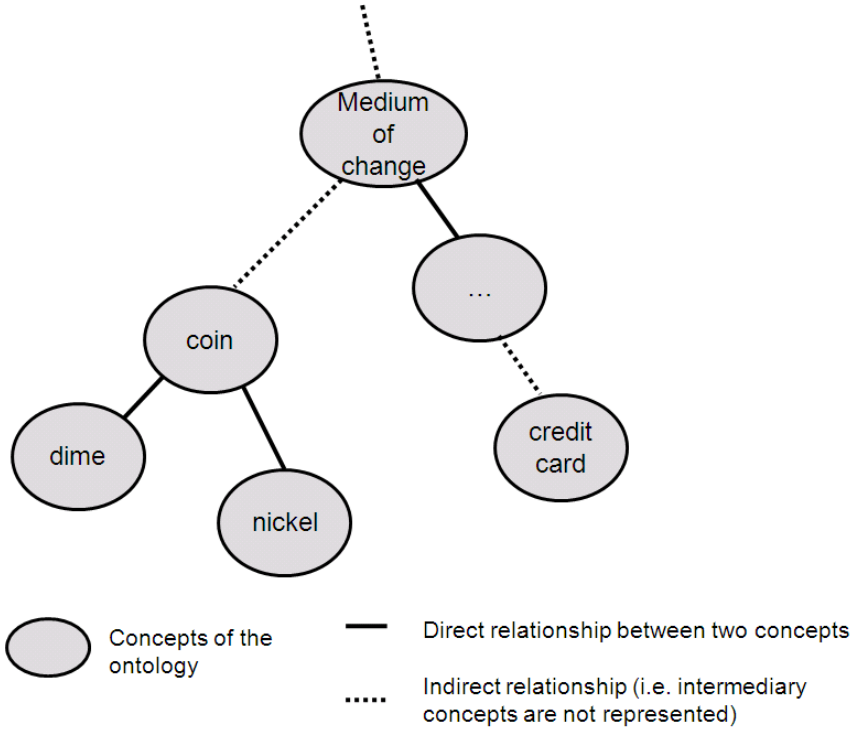


Figure 7-4: Extract of an exemplary ontology

This notion also grounded the link-counting method detailed above. But to avoid the unreliability of link distances, [Resn99] proposes to consider the probability of encountering instances of objects. Thus, semantic similarity measure can be characterised by the following:

The more *information* two concepts share, the more similar they are. Here, *information* shared by two concepts is indicated by the information content (IC) of the concepts that subsume them in the taxonomy.

Following the standard augmentation of information theory [Ross76] (cited by [Resn99]), the information content of a concept (or object) c, denoted (ic_{res} (c)), can be quantified as the negative logarithm of the likelihood (p) of encountering an instance of the concept, such as:

$$ic_{res}(c) = -\log p(c) \qquad \text{E 7.4}$$

The function p is monotonically increasing as one moves up the taxonomy: if c1 IS-A c2 (c1 is a child concept of the parent c2), p(c1)<p(c2). Then, if the taxonomy has a unique top node, its probability is 1. As the probability increases, its information content or "informativeness" decreases; so the more abstract a concept, the lower its information content. If there is a unique top node, its information content is 0.

Formally, Resnik [Resn99] defines the similarity between two concepts c1 and c2 of the taxonomy as:

$$sim(c1,c2) = \max_{K \in S(c1,c2)} \left[-\log(p(K)) \right] \qquad \text{E7.5}$$

, where S(c1,c2) is the set of concepts that subsume both c1 and c2.

Following Resnik's definition, other information theory based similarity metrics using the same notion of information content ($ic_{res}I$) have been developed. Here, information content is obtained through statistical analysis of corpora (e.g. texts), from which probabilities of concepts occurring are inferred. In the considered application domain it is however difficult to identify appropriate corpora. Indeed, what corpora would properly characterise a location's semantic?

Rather in [SeVe+04], an approach relying on taxonomies' hierarchical structure was solely presented. The approach is based on WordNet [WORD]. WordNet is a semantic network database for the English language that provides a description of the semantic relations between words in the English language. Its design, inspired by psycho-linguistic theories of human lexical memory, and the fact that it is the largest semantic network database, make WordNet relevant for our purpose of similarity computation. In its latest version WordNet contains nearly 81500 noun-synsets. A synset is a basic building block, which represents a set of synonyms denoting the same concept, paired with a description. Synsets are interconnected with different relational links, but the IS-A link only (taxonomy) is used in the approach.

The method for obtaining IC values relies on the assumption that the taxonomy structure of WordNet is organised in such a way that concepts with child nodes (hyponyms) convey less information that concepts that are leaves (i.e. concepts at the bottom of the taxonomy that do not have any child). The authors argue that the more hyponyms related to a concept, the less information the concept expresses, otherwise there would be no need to further differentiate it. Likewise the concepts that are leaves are the most specified in the taxonomy so the information they express is maximal. For example, "a vehicle" is less informative than "a motorised vehicle" or "a car". The information content of a concept in WordNet (*wn*) is formally expressed as:

$$ic_{wn}(c) = \frac{\log(\frac{hypo(c)+1}{\max_{wn}})}{\log(\frac{1}{\max_{wn}})} = 1 - \frac{\log(hypo(c)+1)}{\log(\max_{wn})} \qquad \text{E 7.6}$$

, where the function *hypo* designates the number of hyponyms of concept c and $\max_{wn}$ is the maximum number of concepts in the taxonomy.

In [SeVe+04], E7.5 was evaluated in substituting Resnik's E7.4 with E7.6. Results were obtained by correlating the similarity scores with those of human judgements provided in [MiCh91]. To make comparisons, other existing metrics were evaluated as well. According to [SeVe+04] results were found to outperform other metrics, suggesting that the initial assumption concerning the taxonomy structure of WordNet is correct.

This approach seems very appropriate and was selected in our work. Indeed, it does not impose links as a uniform distance. Moreover, in comparison with Resnik's one, it avoids the utilisation of data corpora to compute the information content. Such corpora are difficult to define and obtain for the current task. Finally, results mentioned by the paper's authors correlate well with human judgements, which ultimately govern the similarity assessment that the similarity metric intends to approximate.

Formally, the similarity metric for the Function feature is expressed as:

$$sim_2(f_2^A, f_2^B) = \max_{K \in S(f_2^A, f_2^B)} \left| 1 - \frac{\log(hypo(K)+1)}{\log(\max_{w_n})} \right| \qquad \text{E7.7}$$

where:

f_2^A and f_2^B are the function features of location A and location B respectively, whose domain is the set of concepts in WordNet designating a location.

$S(f_2^A, f_2^B)$ is the set of concepts that subsume both f_2^A and f_2^B.

Hypo designates the number of hyponyms of the concept *K* ;

$\max_{wn}$ is the maximum number of concepts in the taxonomy;

Table 7-10 indicates some example values for some possible location semantic.

	Office	**Living-room**	**Bar**	**Hall**	**Corridor**
Office	1.0	0.534	0.549	0.563	0.355
Living-room	0.534	1.0	0.669	0.689	0.303
Bar	0.549	0.669	1.0	0.699	0.435
Hall	0.563	0.689	0.699	1.0	0.967
Corridor	0.355	0.303	0.435	0.967	1.0

Table 7-10: Similarities for some values of the Function feature

Attention feature (f_3)

As discussed in the previous section, the attention feature can take three different values. To define the similarity between features we utilise the following measure function:

$$sim_3(f_3^A, f_3^B) = 1 - \frac{\left|\theta(f_3^A) - \theta(f_3^B)\right|}{N_{attention} - 1} \qquad \text{E7.8}$$

, where *θ(•)* is a function that assigns an integer to each feature's value, as shown in the table below. $N_{attention}$ is the number of distinct values a feature can take (here set to 3).

In this function the denominator, which is equivalent to the number intervals between the

feature classes serves as a normalising factor, in that it ensures that the similarity value is in [0,1].

Attention level (f_3)	**Value - $\theta(f_3)$**
Low	0
Medium	1
High	2

Table 7-11: Attention level vs. values

Figure 7-5 depicts the different values the similarity function for the attention feature can take.

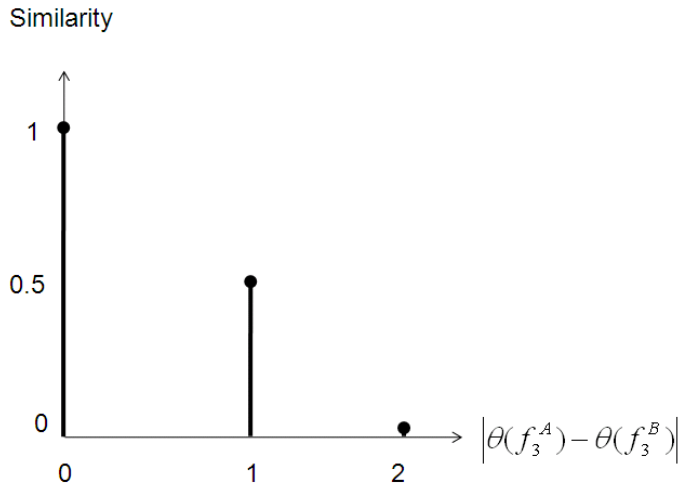


Figure 7-5: Similarity for the Attention feature

Ownership feature (f_4)

In section 7.2 we have defined a set of values for the feature ownership. The next step consists of selecting a metric to assess similarity for this feature.

The commonality between two categorical values is characterised by the interval delimited between them. To design the similarity metric for the ownership feature we postulate the assumption that the interval for any two consecutive values is the same. Moreover, we arrange the defined categories into the following order:

{private, family, friend, colleague, manager, public}

It is ranged from the most private to the less private category. The ordering corresponds to the ones proposed in [OlGr+05], except that the categories Boss/Manager and Co-Worker/colleague are inverted. The motivation for this inversion is so that it seems the pieces of information labelled as private (e.g. marital status, access to my computer, social security

number) are shared with the managers due to their higher rank in the company. It rather seems that from a social perspective a colleague directly follows the friend's circle, as most people tend to have a more informal relation with colleagues than with their managers.

As a consequence, we formally express the local similarity metric for ownership feature as follows:

$$sim_4(f_4^A, f_4^B) = 1 - \frac{\left|\eta(f_4^A) - \eta(f_4^B)\right|}{N_{ownership} - 1} \quad \text{E 7.9}$$

, where *η(•)* is a function that assigns an integer value to each distinct ownership feature's value. The denominator, which is equivalent to the number intervals between the feature classes ($N_{Ownership}$, being the number of these classes and is here set to 6), serves as a normalizing factor, in that it ensures that the similarity value is in [0, 1].

The table below describes the ownership classes and indicates the value that is assigned to each of them.

Ownership classes	**Description**	$\eta(f_4)$
Public	Default class	5
Manager	Any individual working with the user and ranked higher	4
Colleague	Any individual working with the user, with the same rank or below	3
Friend	Any individual identifies as such	2
Family	2. level family members (cousins, aunts, uncles) and partners	1
Private	User and partner	0

Table 7-12: Ownership classes vs. values

Figure 7-6 depicts the similarity value for the various distances between the features.

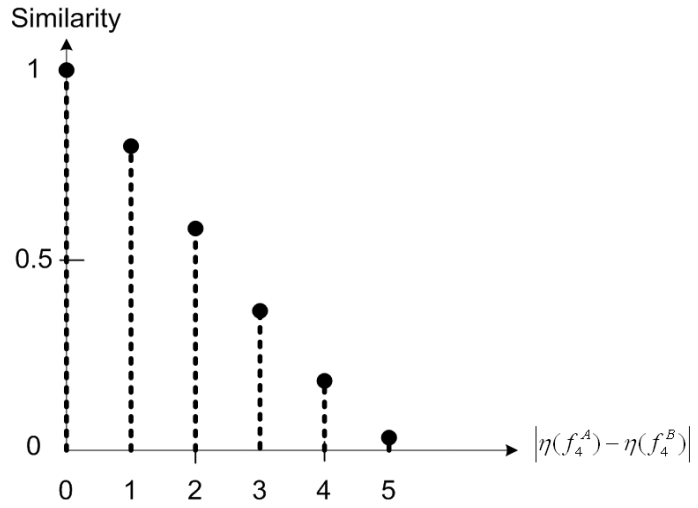


Figure 7-6: Similarity for the Ownership feature

Noise level feature (f_5)

The noise level feature characterises the acoustic noise level measured in the location.

As discussed in section 7.2.3, acoustic noise level is defined on a continuous domain of positive real number ($\Re^+$). However, two values are particularly relevant: the quietest audible sound for persons under normal conditions is 10 dB, whereas the human pain limit is set to 120 dB. These two values are defined as the lower and upper boundaries for noise level measurements. Indeed, below and beyond these two values, measurements are of low significance and do not bring any further information for the assessment of noise level similarity (e.g. two locations with noise level of 10 dB and 5 dB respectively are not very dissimilar in terms of noise level, since it is imperceptible by human beings – both levels are quiet).

We opted for a monotonic decreasing function to implement the feature's similarity metric, on the continuous range [10, 120]. Formally the metric is then expressed as:

$$sim_5(f_5^A, f_5^B) = 1 - \frac{\left|f_5^A - f_5^B\right|}{Noise_{Max} - Noise_{Min}} \quad \text{if } Noise_{Max} < f_5^{A,B} < Noise_{Min} \quad \text{E 7.10}$$

$$sim_5(f_5^A, f_5^B) = 0 \quad \text{else}$$

The similarity metric function for the Noise level feature is depicted in the Figure 7.7. The similarity decreases proportionally to the increase in the noise level's distance.

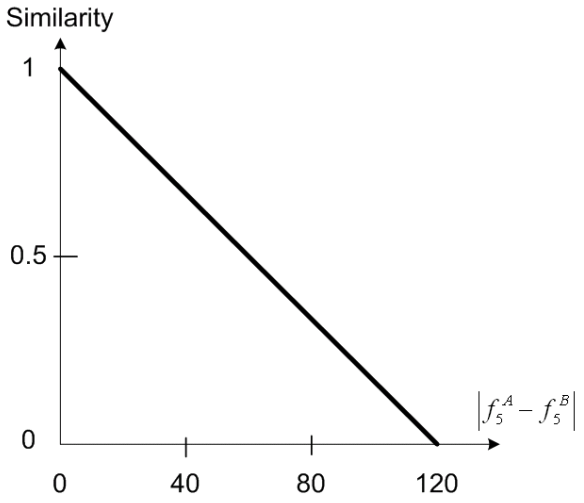


Figure 7-7: Similarity for the Noise Level feature

Revision and Feedback mechanisms for the Call Profile Manager

Besides the similarity metrics, adaptation knowledge also encompasses the indication characterising the revision mode and the feedback mode. We selected *explicit user feedback* as a feedback mode. The revision mode is set to *user-supported*. Indeed, in the case of the Call Profile Manager application, it makes little sense to ask the user for providing correction. How should a user specify he would have preferred to have been notified through notification instead of a loud ring tone? The call already happened. Rather, the GUI displays a list of the 3 remaining behaviours, ranked in a decreasing order according to the computed similarity assessments. The user can then select from the list, what he would have preferred.

7.2.5 Expressing templates

The last step of the application development process consists of developing the application templates. The three types of templates have been discussed in Chapter 6. In what follows, the templates of the Call Profile Manager are presented. They are expressed from the knowledge defined in the previous steps.

Behaviour template

The behaviours offered by the Call Profile Manager are presented in section 7.2.2. The application can enable the blocking of incoming calls. Alternatively the application passes the call and notifies the user with a specific (loud or normal) ring tone or with vibration.

```
1     <?xml version="1.0"?>
2     <definitions name="BehaviorCallProfileManager"
3           targetNamespace=http://example.com/BehaviorCallProfileManager.xml/>
4           <ontology> http://example.com/CallProfileManager.owl</ontology>
5           <relation> XOR </relation>
```

```
6           <message>
7                 <name=”RingToneHighInput”/>
8                 <description=”RingToneHigh”/>
9           </message>
10          <message>
11                <name=”RingToneLowInput”/>
12                <description=”RingToneLow”/>
13          </message>
14          <message>
15                <name=”VibrationInput”/>
16                <description=”Vibration”/>
17          </message>
18          <message>
19                <name=”CallInput”/>
20                <description=”IncomingCall”/>
21          </message>
22          <message>
23                <name=”BlockInput”/>
24                <description=”BlockCall”/>
25          </message>
26          <behaviour>
27                <operation name=”BlockBehaviour”>
28                      <input_message=”CallInput”/>
29                      <output-message=”BlockInput”/>
30                </operation>
31                <operation name=”PassCallHighBehaviour”>
32                      <input_message=”CallInput”/>
33                      <output_message=”RingToneHighInput”/>
34                </operation>
35                <operation name=”PassCallLowBehaviour”>
36                      <input_message=”CallInput”/>
37                      <output_message=”RingToneLowInput “/>
38                </operation>
39                      <operation name=”VibrationBehaviour”>
40                      <input_message=”CallInput”/>
41                      <output_message=”VibrationInput”/>
42                </operation>
43          </behaviour>
44    </definitions>
```

The template describes the four possible behaviours of the application, referred to as: `BlockBehaviour`, `PassCallHighBehaviour`, `PassCallLowBehaviour`, and `VibrationBehaviour`. For each of them, we indicate the behaviour's input message, i.e. the event triggering the behaviour and its output message, i.e. the result it produces. It

should be noted that the template does not give any indication about how the application further reacts to the behaviour, e.g. when the call is blocked the caller can leave a message.

As discussed before, the behaviours of the application are exclusive disjunctive, i.e. one of them is performed at a time. This is indicated by the XOR relation tag. Finally, the ontology to which the template makes reference is described within a file referred to as `CallProfileManager.owl`.

Context features template

The context features template specifies the context features to which the application reacts. We have presented them in the previous section: *Function, Position, Ownership, Attention and Noise level*.

```
1     <?xml version=”1.0”?>
2     <definitions name=”ContextCallProfileManager”
3           targetNamespace=”http://example.com/ContextCallProfileManager.xml”
4           <ontology>http://example.com/ContextCallProfileManager.owl</ontology>
5           <context>
6                 <name=”Position”/>
7                 <description=”location#GPS_coordinates”/>
8                 <target=”user#device_owner”/>
9           </context>
10          <context>
11                <name=”Function”/>
12                <description=”location#semantic”/>
13                <target=”user#device_owner/>”
14          </context>
15          <context>
16                <name=”Ownership”/>
17                <description=”location#ownership”/>
18                <target=”user#device_owner”/>
19          </context>
20          <context>
21                <name=”Attention”/>
22                <description=”attentionLevel”/>
23                <target=”user#device_owner”/>
24          </context>
25          <context>
26                <name=”Noise Level”/>
27                <description=”noiseEnvironment”/>
28                <target=”user#device_owner”/>
29          </context>
```

```
30    </definitions>
```

Each context feature is described by a name and is associated to an ontology concept and a target. The ontology to which the template makes reference is indicated by the `ontology` element. In this example, the current ontology differs from the that of the behaviour template. However, one single ontology could be used for both templates. A context feature can be either described by a concept of its own (such as `attentionLevel`) or by the property (`ownership`) of a concept (`location`). Finally, the template indicates that all features refer to the device owner. Indeed, no context information related to other entities, e.g. persons in the environment is required.

Adaptation knowledge template

Finally, the adaptation knowledge template of the Call Profile Manager application is depicted below. It gives indications about the local similarity metrics used and the Degree of Freedom selected by the application developer.

```
1     < ?xml version= ”1.0” ?>
2     <definitions name= »SimilarityCallProfileManager »
3
      targetNamespace= »http ://example.com/SimilarityCallProfileManager.xml »
4           <contextTemplate>http ://example.com/ContextCallProfileManager.xml
            </contextTemplate>
5           <similarity name=”SimPosition”>
6                 <input=”Position”/>
7                 <location = »http ://example.com/PositionFunction.xml »/>
8           </similarity>
9           <similarity name=”SimAttention”>
10                <input=”Attention”/>
11                <location = »http ://example.com/AttentionFunction.xml »/>
12          </similarity>
13          <similarity name=”SimNoiseLevel”>
14                <input=”NoiseLevel”/>
15                <location = »http ://example.com/NoiseLevelFunction.xml »/>
16          </similarity>
17          <similarity name=”SimOwnership”>
18                <input=”Ownership”/>
19                <location=”http://example.com/OwnershipFunction”/>
20          </similarity>
21          <similarity name=”SimSemantic”>
22                <input=”Semantic”/>
23                <location = »http ://example.com/semanticFunction »/>
24          </similarity>
25          <feedback>explicit</feedback>
```

```
26q          <revision>user-supported</revision>
25     </definitions>
```

The system makes use of a local similarity metric for each context feature presented in the context features template. These similarity metrics, characterised by a name (e.g. `SimPosition`), are related to a type of feature (e.g. `Position`). The location of the similarity metric is indicated by the `location` element. Besides, the template indicates that the selected feedback mode is explicit feedback (`explicit`) and the correction mode is user-correction (`user-supported`), i.e. the user himself selects the proper behaviour.

7.3 Conclusion

The framework presented in Chapter 4 and the mechanisms described in Chapters 5 and 6 provide support to contextual personalised applications. However, this class of applications presents some constraints that need to be addressed by the developers to take full advantage of the framework's principles. In this chapter we discussed the steps any such application has to go through at design time and we exemplify them by means of a prototype application, the Call Profile Manager application.

To make use of the support system which is designed from the framework's guidelines described in Chapter 4, an application has to address the following requirements: the application has to be contextual personalised so that its personalisation is governed by context-dependent preferences. The behaviours of the application have to be either exclusive disjunctive or performed concurrently. Finally, when being designed, the application has to go through an extended development process. That is, besides the traditional operations including e.g. the development of a software blueprint and the implementation of the various modules, application-specific knowledge has to be defined and expressed in templates, as discussed in Chapter 6.

To demonstrate the support provided by the Copernik framework, we have developed an application, the Call Profile Manager application, which permits the management of reception of incoming calls according to the user's context. In the second part of this chapter we described the development of specific knowledge for this application. In particular, we defined five different types of context information to which the application is designed to react: user's position, location's function, level of required attention in the location, ownership and noise level in the location. The choice of these features has been made by considering an application scenario. However, as underlined in the discussion of the development process, additional or different set of features could be defined by other developers. Furthermore, the respective templates of the three types of knowledge are described.

In this chapter, we did not discuss how values for the different features are gathered from the user's environment. As mentioned before the context gathering mechanisms do not fall into the scope of this work.

8 Implementing the framework

This chapter describes a prototypical implementation of a system following the guidelines of the framework and using the cluster-based retrieval function. The objective of the implementation is twofold. Firstly, the prototype can be seen as a proof of concept. In this way it shows the feasibility of the framework's realisation. Secondly, the prototype serves as a test-software to evaluate the approach. The experimental evaluation and the corresponding results are discussed and presented in Chapter 9.

In this chapter the main requirements to the prototype and the design decisions are first discussed. The architecture of the system implementing the framework is derived from the requirements and is further described. The components of the architecture are presented and some of the main general operations between them are depicted in a set of message sequence diagrams.

8.1 Requirements and design decisions

The implementation of any new piece of software is governed by a set of requirements. One typically distinguishes between functional requirements that define and characterise the main functionalities expected from the software and non-functional requirements governing aspects of the system structure. The requirements that are presented in the following are derived from the concepts detailed in Chapter 4 and 6.

Non-functional requirement

Non functional requirements, unlike their functional counterparts do not impose any system functionality. Rather they concern the system structure.

- The system must centrally process user information.

The separation of concern principle introduced in 4.3 calls for system-based contextual personalisation and as a consequence the development of software systems to support contextual personalised applications. These systems, referred to as contextual personalised systems, must serve as a central processing unit to the user information and carry out the operations required to personalise the applications. This provides more privacy of user data. However, it does not impose any requirement on the distribution of the unit. Indeed, parts of the system can be distributed over a network or run together locally.

Functional requirements

Functional requirements characterise the functionalities that the system is expected to fulfil. The following requirements are derived from the principles given in Chapter 4 and the template description in Chapter 6.

The system must:

- manage application-specific knowledge

Application-specific knowledge is provided by a contextual personalised application to the system via a set of templates. The knowledge describes the application behaviours, the context features to which the application reacts and the adaptation information. For any application supported by the system, application-specific knowledge must be managed. This imposes that up-to-date templates have to be stored in the system, so that currently relevant knowledge is used in the adaptation process. Thus, the system must be able to collect new templates, and store them. Besides, these xml-based templates have to be parsed to determine the application-specific data and provide them to the system software entities that need them.

- monitor applications

The system must monitor the applications. Indeed, personalisation is triggered by a set of events occurring in the applications (e.g. incoming call for the Call Profile Manager). Thus, the system must be able to catch these events to consequently perform the adaptation process. Besides, when the system has triggered the application behaviour(s), user's actions (via the application's GUI) must be monitored, e.g. when the application is started or stopped, what behaviours are manually selected by the user, what information is typed in by the user, etc.

- handle appropriate context description with regards to the application anytime required

To perform personalisation when requested, the system must get a description of the current user's context. As context can take different forms (various context features and/or granularity level can be needed), the system must rely on context descriptions that are made up of the only context features to which the application reacts. Hence, the system must be able to handle different context descriptions for the different applications it supports.

As discussed in 4.1.3, we assume in this work that complete and correct context descriptions are made available in an underlying system anytime it is requested. As a consequence, the contextual personalised system does not specifically implement the operations to gather sensor information from the user' environment and process them.

- determine the required application's behaviour

Based on the set of previous user's experiences, the system must determine the applications' action that currently best match the user needs.

- trigger the needed application behaviour

The system must trigger the determined application behaviour by calling the application interface.

- provide the user with explanation

When adapting a contextual personalised application, the system must provide an explanation to the user about the triggered application behaviour.

- request and gather user feedback

Following any adaptation decision, the system must ask for and gather the user feedback to the adaptation. Also, upon a negative feedback, which indicates the user's disagreement towards the presented behaviour, the system must correct the behaviour, presenting a new behaviour that will generate positive feedback.

- collect and maintain the user's previous experiences with the application

The system must manage the case base, where the user's experiences are stored. Managing the case base consists of storing new experiences, and possibly updating or deleting previous ones.

8.2 System Architecture

The aforementioned requirements lead to the development of the contextual personalised system, whose architecture is depicted in Figure 8.1.

The required functionalities for supporting contextual personalised applications are provided by Contextual Personalised System (CP-System). Additional functionalities to gather and interpret context are provided by the GCI (Context Gathering and Interpretation) System and are not further discussed in this section.

The CP-System is structured around 10 components that cooperatively support contextual personalised applications. Below we give a description of the operations and responsibilities of these components.

Interaction Manager

The Interaction Manager component is responsible for interacting with the contextual personalised applications. On the one hand, the component triggers the application behaviour that best matches the user's current need as determined by the adaptation process. On the other hand, the component implements the observer pattern ([GaHe+95] pp.293) and listens to the events occurring at the application side, such as application interruption (the application is terminated by the user).

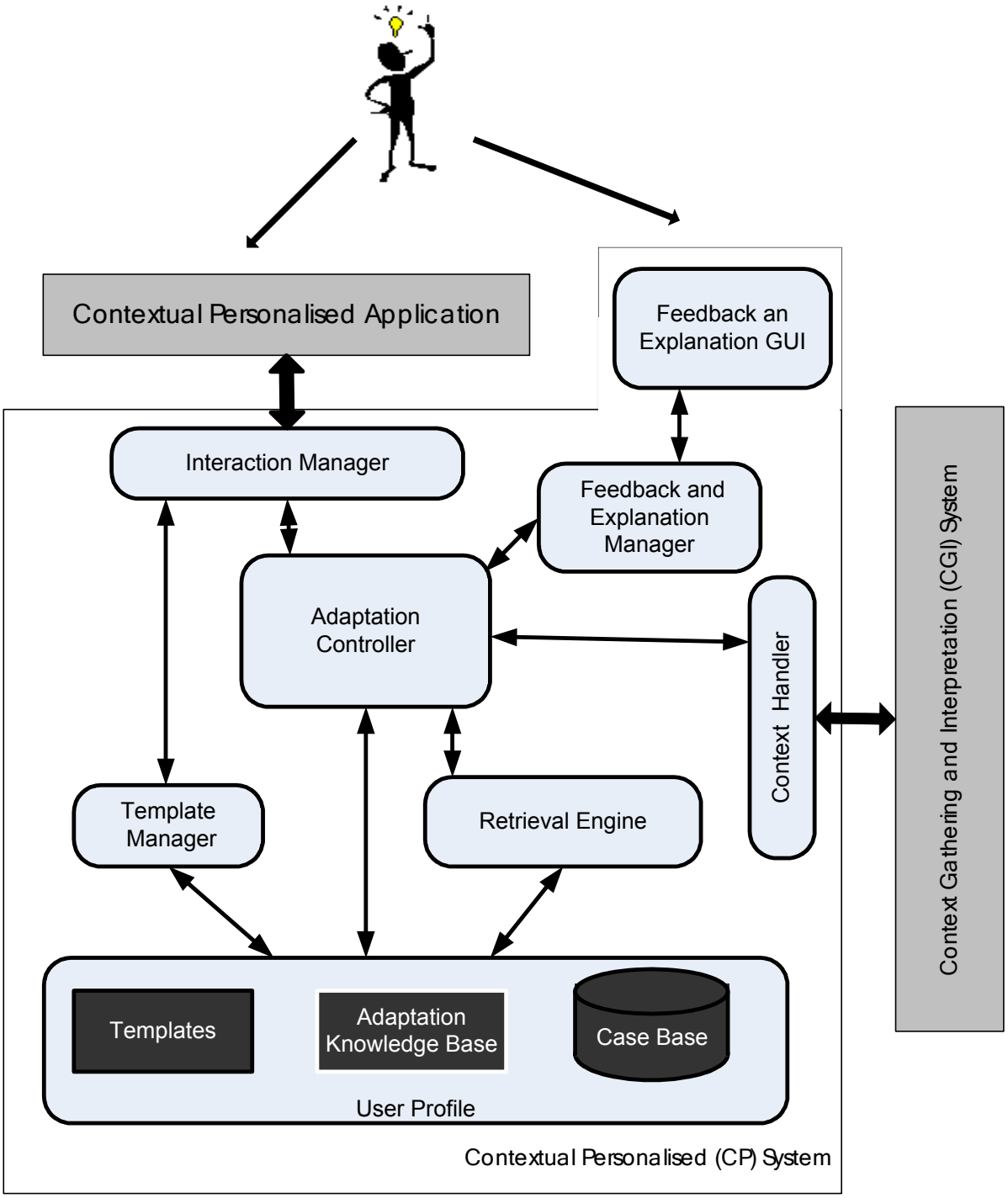


Figure 8-1: Architecture of the implemented CP System

Adaptation controller

This component is the central component of the Contextual Personalised system. It administers and dispatches the calls to the components of the system to perform the actions specified by the Framework in Chapter 4. Besides, the component is responsible for managing the user's previous experiences. As such, it composes a new case when a successful adaptation has occurred and can delete cases from the Case Base, e.g. when the application does not react to the same context features anymore.

Context Handler

This component monitors changes in the user location context. It subscribes to some context providers (e.g. sensors, actuators) to gather context information about the user. It further builds a description of the user location as described in the context model. When the component has detected changes in the user location (when needed by the application, i.e. when the application reacts to context changes and repeatedly displays a behaviour as opposed to when the application displays a behaviour only once), it triggers the determination of the application behaviour at the Behaviour Manager component.

In addition the architecture features another building block: the Context Gatherer and Interpreter. It is no architecture component, but it is used by the Context Handler to retrieve context information that is required to describe the current user location.

User Profile

In the User Profile component information required for the adaptation process is stored and maintained. We distinguish between three types of information:

- User's previous experiences with the applications

  The user's previous experiences with the application are stored in the case base. Each previous experience is composed, as discussed in Chapter 4, of a context description (encompassing values for the context features to which the application reacts) and the application behaviours and parameters that have been selected. In addition, each case is complemented with a reference to the application and the template's version to which it refers. This enables the retrieval of the application's specific cases among all cases from (possibly) different applications.

- Adaptation knowledge

  In the adaptation knowledge base, the set of data used for the adaptation of each application is stored. It includes parsed data from the templates, as well as additional data such as the similarity metrics or application ontologies. The similarity metrics defined in the adaptation knowledge template are stored in the User Profile. The similarity metrics serve to drive the retrieval of similar user's previous experiences carried out by the Retrieval Engine.

- Templates

  For each application, up-to-date versions of the templates are stored locally. These templates are deleted when new versions are made available. Besides, the application-specific knowledge that has been parsed from the XML-templates is stored in the form of an attribute-value pair. Thus, every time knowledge is required during the process, templates are not processed again.

Feedback and Explanation Manager

The component's role consists of constructing the Graphical User Interface to provide explanations to users and gather user feedback. The component is also responsible for processing the user feedbacks (e.g. semi-explicit feedbacks, see Chapter 4) and to determine the adaptation parameters when input by users in the User-supported correction mode.

Feedback and Explanation GUI

The Feedback and Explanation GUI displays to the user an explanation about the behaviour, which has been triggered, and asks for user's feedback, following the application feedback's mode (see Chapter 4 for a discussion).

Template Manager

The Template Manager component deals with the application templates. When templates are received by the system, they are processed to parse the application-specific knowledge. This knowledge is further stored in the User Profile. The Template Manager component also gathers additional requested data related to the templates, e.g. local similarity metrics, application ontologies.

Retrieval Engine

The Retrieval Engine component implements the retrieval strategy. Based on the description of the user's current context, the component searches among the cases characterising the user's previous experiences for 1) obvious similar case, or 2) proximate match. When no such case is found, the component applies the retrieval function, which is described in Chapter 5. As such, it clusters the user's previous cases and determines the set of most similar cases. Subsequently, the Retrieval Engine component computes the explanation information that is to be displayed to the user, based on the retrieved case(s).

8.3 System operations

In this section we describe the interactions between the architecture components for some of the operations to be carried out to support contextual personalised applications. The operations are depicted in a set of sequence diagrams that feature the system's components. In addition, the flows of information between these components are depicted. These diagrams do then not intend to represent implemented classes or to indicate typed data.

The following three operations are depicted:

- How the new case's solution part is determined

  An operation, which has been previously accessed by the user (user's previous experiences have been gathered), is about to be adapted. The diagram depicts the sequence of actions from the event triggering the adaptation to the determination of the solution part for the new case.

- How the user interacts with the system after adaptation occurs

  Following the sequence of actions presented above, the diagram depicts the components and the flow of information that completes the adaptation process.

- Application registration, template update

  The diagram depicts what happens when a new application is accessed (i.e. to be used for the first time by the user) or when new templates are made available for the application. This diagram focuses on the sequence of actions before the adaptation process itself gets started.

User preferences' determination

The sequence diagram (Figure 8-2) represents the sequence of actions required to determine the user preferences and hence adapt a contextual personalised application.

Adaptation starts when a predefined event occurs at the application's side and is notified to the system. The Interaction Manager component is responsible for listening to applications' events implementing the observer pattern. Capturing the event notification, the Interaction Manager notifies the Adaptation Controller that initiates the subsequent operations of the adaptation process. Firstly, a description of the user's current context is needed. As different applications may react to different contexts, the Adaptation Controller retrieves from the User Profile the list and the description of the application-specific context features. They are further passed to the Context Handler component. This component's role consists of interfacing with the underlying CGI (Context Gathering and Interpretation) system. As it is considered an external system to the CP (Contextual Personalised System), it is not depicted on the diagram. Secondly, the context description is provided to the Retrieval Engine component. This component is in charge of 1) retrieving similar cases and 2) adapting the solution part of the retrieved cases. To this end, all the user's previous cases (with the application only) are retrieved from the User Profile. Then, the retrieval phase is performed: a) searching for obvious similar case b) searching for proximate match and finally c) searching for similar cases (as described in Chapter 5). Subsequently, the solution of the selected cases is adapted. Information composing the solution part of the new case

(adaptation parameters) is finally sent to the Adaptation Controller component.

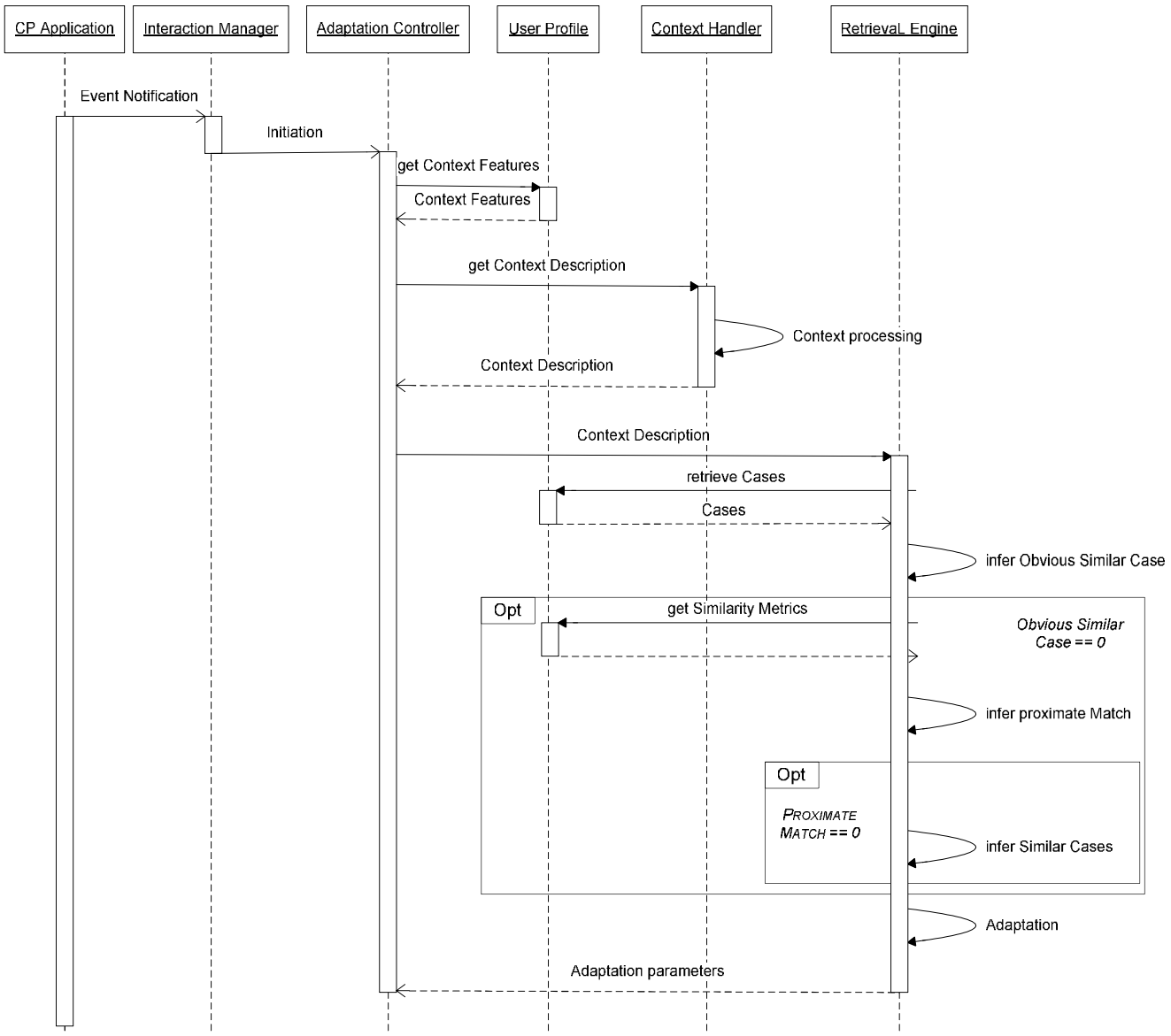


Figure 8-2: Determining the solution part of the new case

Performing adaptation

In the second diagram, Figure 8-3, the operations following the determination of the new case's solution part are depicted.

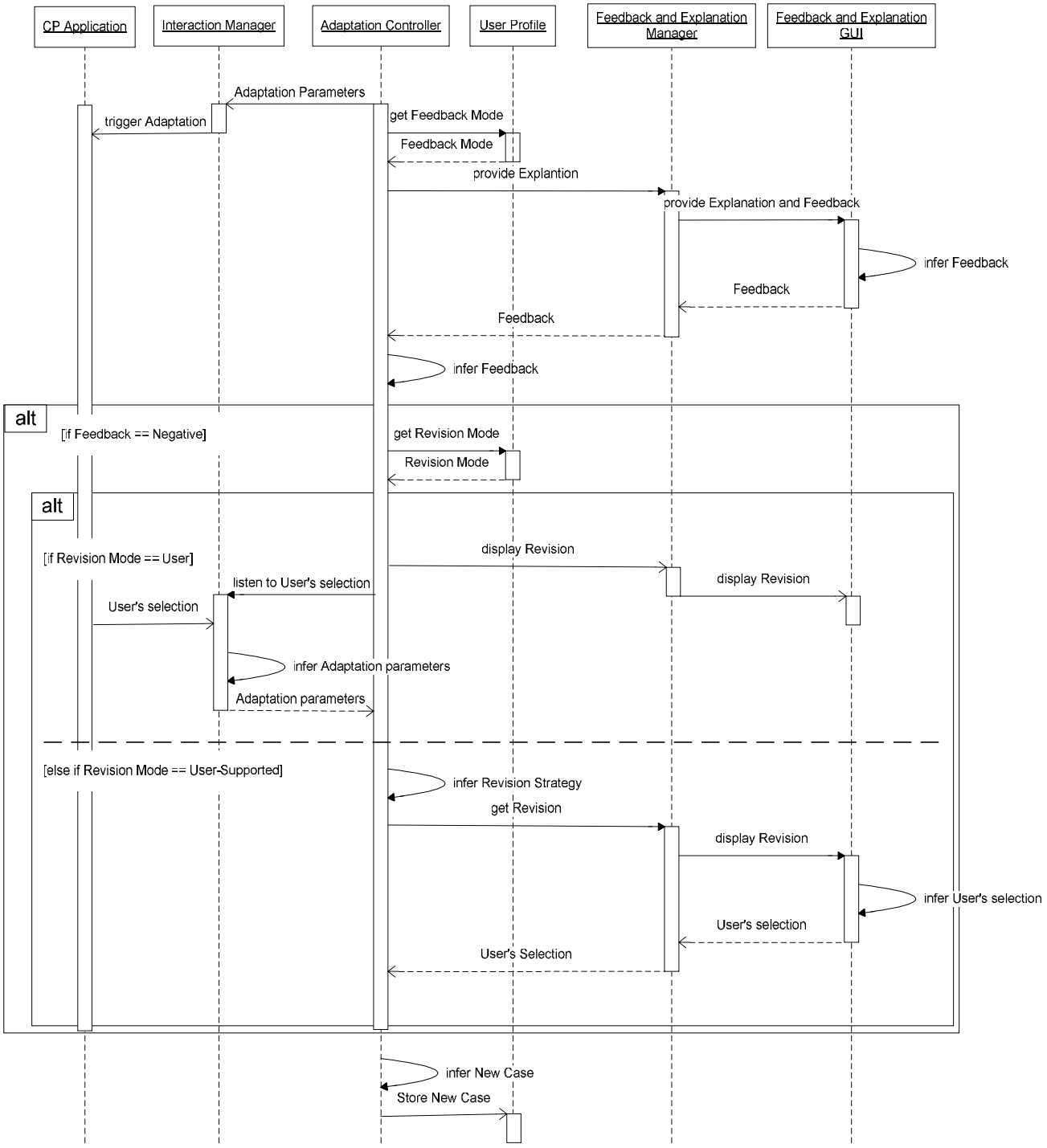


Figure 8-3: User's interactions in the adaptation process

The sequence begins with the adaptation parameters being known by the Adaptation Controller component. They are then transmitted to the Interaction Manager component, in charge of triggering the right behaviour(s) of the application. Once this has been made, the application has been adapted. Indeed, a behaviour of the application has been triggered based on the user's context. However, as discussed in Chapter 4, the system has to make sure that the user needs have been correctly identified, so that the new case can be faultlessly learnt.

Hence, following the application triggering, an explanation has to be given to the user and a feedback has to be recorded. The framework described in Chapter 4 presents several modes for gathering feedback. For each application, the required feedback mode is indicated in the application-specific template. Thus, the Adaptation Controller must once again access the User Profile and get the piece of information specifying how feedback has to be collected for the current application. Subsequently, the explanation about the application's adaptation is displayed to the user and the user's feedback is asked via the Feedback and Explanation GUI. The GUI is composed of the Feedback and Explanation Manager component.

According to the type of feedback given by the user, the sequence of actions differs slightly. When the user agrees with the proposed application behaviour, the new case is then built and stored in the case base of the User Profile. However, when the user disagrees with the adaptation, a correction has to be made. Again, the correction mode is specific to the application. The Adaptation Controller retrieves then the current application's correction mode from the User Profile. When the correction is up to the user (User Mode), the Feedback and Explanation GUI displays a short message, asking the user to select the behaviour by e.g. pressing keys from the application GUI. The Interaction Manager component listens to the user inputs and infers the adaptation parameters from them. These parameters are passed to the Adaptation Controller to build the solution part of the new case. Alternatively, when the user mode is User-Supported, the possible application behaviours are displayed to the user (see Chapter 4). The user's selection is passed back to the Adaptation Controller. Finally, the new case is built and stored in the case base of the User Profile. Correction can also be made by the system, without the user's intervention at first. This is, however, not explicitly depicted in the diagram, as it implies the same sequence of actions than the one determining the initial adaptation proposal.

Template update

The last diagram, Figure 8.4, represents the sequence of actions taking place when templates are provided to the system. These templates have been discussed in Chapter 6.

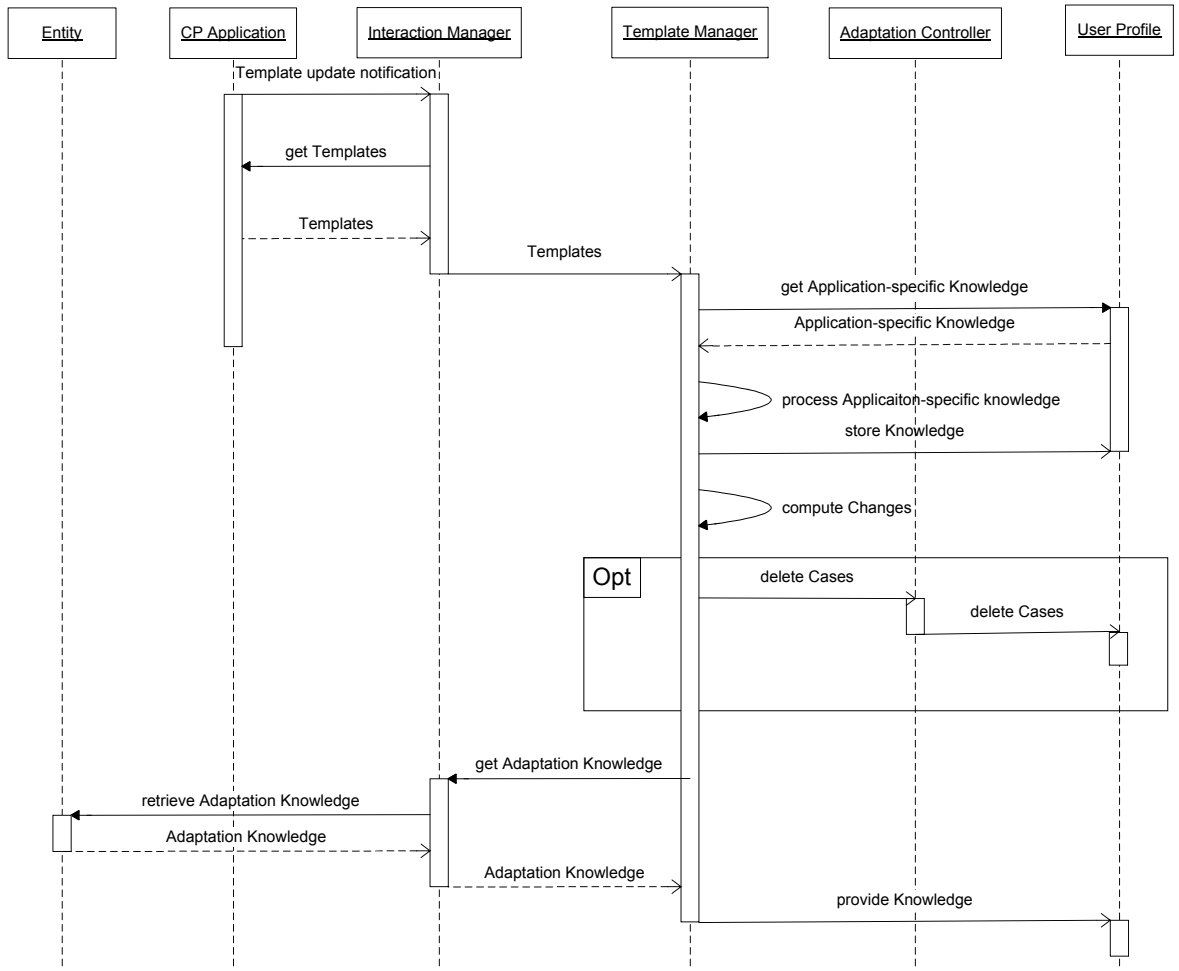


Figure 8-4: Template update

Templates are provided to the system on two occasions. On the one hand, application-specific knowledge has to be indicated when the application is about to be used for the first time by the user. On the other hand, new templates are communicated to the system when they are updated, i.e. a change in the application-specific knowledge occurs.

In Figure 8-4, we represent only the latter case, as it also depicts the actions of the former. The phase is initiated as the Interaction Manager component receives the notification of a template update. The component then retrieves the templates that have been modified from the application's side and forwards the templates to the Template Manager component. This component further processes the templates and determines the new application-specific knowledge, which is further stored in the User Profile. Subsequently, the Template Manager component determines what the template changes imply for the system. Indeed, when an application behaviour is no longer available, cases related to this behaviour are no longer needed in the Case Base. Similarly, when context features to which the application reacts have changed, all cases can be deleted as it will be of no avail to further compute the similarity between previous contexts and the next user's current context. When such changes occur and request cases to be deleted from the Case Base, the Adaptation

Controller component is notified. Alternatively, new templates can introduce new similarity metrics for the similarity computation or new application ontologies. The Template Manager component subsequently retrieves these data and stores them into the User Profile component.

8.4 Conclusion

In this chapter, the implementation of a system dedicated to the support of Contextual Personalised applications was described. Both functional and non-functional requirements that drove the development of the system have been derived from the framework's principles presented in Chapter 4. The components that constitute the system have been described and the sequences of actions for key operations have been depicted.

The system was implemented in the Java programming language with the version 1.5 of the Java Platform Standard Edition (J2SE) [J2SE].

The core functionalities for this system for adapting contextual personalised applications were discussed in this chapter. However, the functionality in charge of gathering and interpreting context information is not detailed in our framework. To this end, it can make use of two additional frameworks developed at the Department for Communication Technologies at the University of Kassel, by other researchers (see Appendix for a brief description)

Finally, it is worth adding that the Call Profile Manager has been realised for demonstration purposes It displays a simple interface to users and indicates the behaviour being triggered. Moreover, incoming calls can be simulated for the demonstration by pressing a button of the interface.

9 Evaluation

In this chapter, an evaluation of the cluster-based retrieval function is presented. This evaluation is performed by means of the Call Profile Manager application, which has been introduced in Chapter 7.
The chapter is structured as follows. Firstly, we discuss the objectives of the evaluation. In particular, we consider the limitations and the specificities of the evaluation constraining its feasibility. Secondly, we give a description of how the evaluation is performed and provide the results of the evaluation.

9.1 Evaluation objectives

In research work, the evaluation is a crucial part, since it can assess the validity, correctness, and meaningfulness of the assumptions made in the research . In the context of this work, a first evaluation can be performed experimentally.
However, before starting with any evaluation, it is paramount to clearly identify its objectives, i.e. what goals are being pursued in the evaluation and what questions the evaluation is expected to answer.

The work presented in this thesis is comprised of three parts, each of them addressing the requirements brought by contextual personalised systems.

- Copernik, the Contextual Personalised Application Framework, which defines a set of principles to support contextual personalisation applications.
- Adaptation-specific knowledge and the associated templates, which are required for a contextual personalised system (following the guidelines and principles of Copernik) to support contextual personalised applications.
- The retrieval function that is suggested for the retrieval phase and that retrieves from the user's previous experiences, the ones that are the most similar to the current user's context.

Ideally, the evaluation of this work must target all its aspects, and as such investigate in-depth the three aforementioned parts.

In Chapter 3 we have discussed the limitations of current realisations targeting contextual personalisation and presented the framework in Chapter 4. The framework has been governed by a set of requirements and principles that cooperatively participate in addressing the limitations. A contextual personalised system following the framework principles has been developed and is described in Chapter 8. As such, the software implementation can serve to demonstrate the feasibility of the approach as well as the advantages it can bring to users, which result from the realisation of the analysed requirements.

Similarly, the templates discussed in Chapter 6 are defined as a consequence of the separation principle between contextual personalised system and contextual personalised applications. As adaptation is required to be performed in a central system, and since applications rely on specific information to be personalised, templates are developed to enable this information to be communicated to the system. The software implementation also provides a means to assess the approach.

Thus, the evaluation presented hereafter concentrates on the retrieval function that is presented in Chapter 5. Applying this approach, we aim to facilitate the retrieval of previous user's experiences. In this way, we aim to trigger, in the user's context at hand, an application behaviour consistent with the behaviour that would be selected by the user, *as often as possible*.
The difficulty of performing an evaluation in the context of this work is, however, threefold:

Estimating the quality of the retrieval task
As discussed in Chapter 2, Contextual Personalisation is at the crossroads of the context-awareness and adaptive systems research fields. As an adaptive system, a contextual personalised system adapts to its user's preferences and uses an interference mechanism to provide the adaptation features. Here, the interference mechanism consists of the retrieval function, responsible for retrieving similar user's previous experiences, and the adaptation of the retrieved solution parts. An exact evaluation of the retrieval function would focus on investigating how correct this interference mechanism performs, i.e. how *good* the retrieved user's experiences are. This is, however, an arduous task, since it is difficult even for human beings to assess this criterion. Even the notion of *good retrieved experience* might be complex to handle. Is it only the most similar experience? What if its solution part does not lead to a correct adaptation?
In order to approximate this notion, one can investigate whether the retrieval function and the adaptation features lead cooperatively to a correct adaptation, i.e. whether the triggered application behaviour corresponds to what the user expects. However, even when the adaptation happens correctly to the user, this does not imply that the best experiences have been retrieved.

Constraining the application domains
The primary objective of the work is to develop an approach that can be applied to support applications from various domains and address different scenarios. Of course, carrying out an exhaustive evaluation covering all these application domains is not feasible, as this would require every single one of them to be considered distinctly. As this work does not encompass any assumption on the domains, this can be large, i.e. greater than 10 (4 distinct scenarios are already listed in section 7.1). For each application domain, specific data (as presented in 7.2 and 7.3 for the Call Profile Manager) have to be specified and modelled.

Hence, the evaluation of the retrieval function is only feasible (considering the limited resources in terms of persons and time available for this thesis) for a limited subset of these domains.
The experimental scenario which has been selected is based on the Call Profile Manager application. The motivation for selecting the Call Profile Manager is that this application can be utilised by users, i.e. offering contextual personalised behaviours, in a large variety of contexts. These contexts can be easily distinguished from one another, as they can be related to the user's proximate environment, i.e. the location of the user and the characteristics of this location.

Acquiring user data
Contextual personalised systems act on behalf of their users. The evaluation of these systems requires user data that serve as a teacher to assess the correctness of the system's adaptation. Basically, these data specify the user preference(s) in a given context. Such data can be obtained in two ways. Either they acquire from persons, or they are simulated by means of software. Simulating data by software is difficult, as the user preferences are not randomly determined but rather they are the result of a complex cognitive process. It is the aim of the retrieval function to approximate this process. On the other hand, relevant and useful data from accurate test users may be difficult to acquire in research environments, as remarked by [Stah03]. The method employed to collect and determine user data is developed in Section 9.2.3.

The experiment described in the following section aims to perform the evaluation of the retrieval function in comparing this function with the Nearest Neighbour method. This method is used in the existing approaches relying on Case-based reasoning that are presented in 2.3.1. In short, the Nearest Neighbour approach retrieves the user's experience, whose problem part is the most similar to the new context.
A complete evaluation of the retrieval function would also require an evaluation of various aspects related to the application-specific knowledge of the application (e.g. impact of the features or of the similarity functions) or parameters discussed for the retrieval function. The investigation of these aspects and parameters would require a large number of experiments, requiring hardware and time resources.
Hence, we have restricted the evaluation to the most interesting aspect only.

- Relative comparison of both approaches according to the amount of training data required.

First of all, one objective of the evaluation is to acquire knowledge about how well the approach we proposed behaves in comparison to the existing approach. The focus of the evaluation consists then of investigating if the application adaptation can benefit from our

approach in terms of *correctness of adaptation*, i.e. in a new context the application is adapted as the user expects it. In that regard, we are interested in comparing the approaches as the number of training data, i.e. user experiences which are available in the user profile, increases.

9.2 Evaluation of the Call Profile Manager

In the following section we describe the evaluation of the retrieval function based on the Call Profile Manager application. Firstly, we introduce Adaptation correctness and explain how it is computed. We further describe the steps to be taken to perform the evaluation. In addition, we present the settings of the experiment as well as the generation of the test data. Finally, we discuss the results of the experiment.

9.2.1 Adaptation correctness (*AC*)

The evaluation of the retrieval function is performed using user experiences, which are expressed following the general case representation, (Section 4.3.2) as:
$\{< f1, f2, f3, f4, f5 >; Application\ behaviour\}$

- $< f1, f2, f3, f4, f5 >$: represents the context and is characterised by the features defined in Chapter 7 for the Call Profile Manager (*Position, Semantic, Ownership, Attention, Noise Level*).
- $Application\ behaviour$ designates the application behaviour which conforms the user preferences. In Chapter 7, four application behaviours are defined for the Call Profile Manager (*Block, Loud Ring Tone, Normal Ring Tone and Vibration*)

The general idea of the procedure is as follows:

For the user experience *A*, expressed as:
$\{< f1^A, f2^A, f3^A, f4^A, f5^A >; Application\ behaviour^{A1}\}$
adaptation is performed based on the context expressed as
$$< f1^A, f2^A, f3^A, f4^A, f5^A > .$$
Hence, the adaptation (based on the previous user's experiences available in the user profile) determines an application behaviour designated as $Application\ behaviour^{A2}$

We evaluate how *good* the adaptation process for the user experience *A* is in comparing the user preference and the result of the adaptation.
Hence, if the computed application behaviour for the new context ($Application\ behaviour^{A2}$) matches the user preference ($Application\ behaviour^{A1}$) the result can be assessed as *correct*. If not, the result is considered as *erroneous*.

We designate by (*Adapt(A^{CB})*) the result of the adaptation performed on the user experience *A* and rely on a set of previous user experiences available in the user profile (or case base) *CB*. Formally it can be expressed as:

$$Adapt(A^{CB}) = \delta(Application\ behaviour^{A1}, Application\ behaviour^{A2}) \qquad \text{E9.1}$$

where:
$\delta(X,Y) = 0$ if $X \neq Y$
$\delta(X,Y) = 1$ if $X = Y$

Clearly this function can take two values: 0 (if adaptation erroneous) or 1 (adaptation is correct). In order to obtain meaningful evaluation data, this operation is required to be performed on a larger set of user experiences.

We consider then the *adaptation correctness (AC)* performed by the system S, which is computed as the arithmetic average of the adaptations (*Adapt()*) performed over a set of user experiences Ω (containing exactly *N* experiences) with a user profile CB. Note that CB is identical for the *N* calculations of *Adapt()*.

Formally:

$$Adaptation\ Correctness: AC(\Omega, CB) = \frac{1}{N} \times \sum_{i=1}^{N} Adapt(\theta_i^{\Omega,CB}) \quad \text{(E 9.2)}$$

Where: $\theta_i^{\Omega,CB}$ is the i*th* user experience in the set Ω. The adaptation is performed based on the previous experiences stored user profile CB.

9.2.2 Evaluation steps

The general procedure of a test experiment is discussed in this section. It is performed via a series of steps, which are illustrated in Figure 9-1.

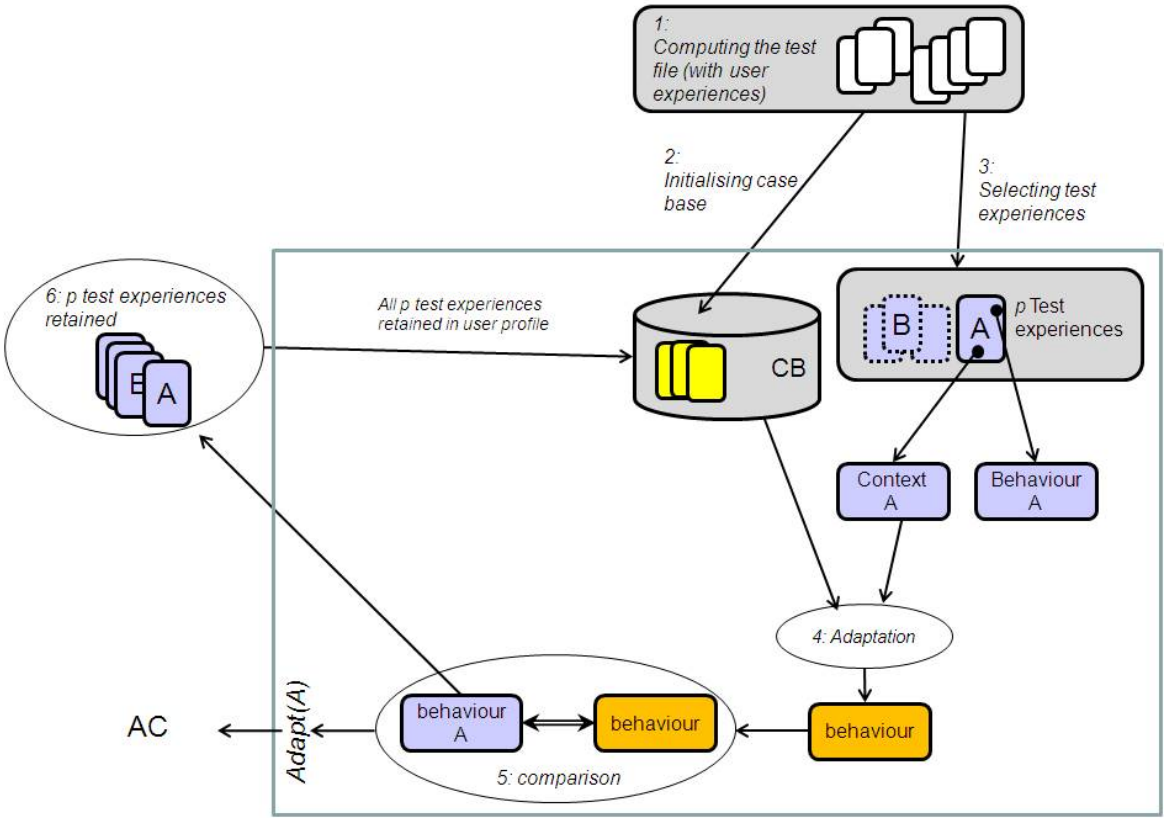


Figure 9-1: General evaluation procedure

Before the procedure starts, however, it is required to build a data set, referred to as *test data set*, where each piece of data relates a context to a user preference with regards to the behaviour of the Call Profile Manager. How these data are obtained is discussed in the next section.

The steps of the evaluation procedure are as follows:

1. Computing the test file

The first step of the procedure consists of computing the test file. The test file contains a sequence of experiences. In this phase, *all test data* are placed into a file and randomly ranked. Hence, all test file contains exactly the same test data. However, they are ranked differently.

2. Constructing the initial user profile with the first *n* user experiences

In the second step, the initial case base is constructed, in selecting the first *n* user experiences from the test data file. This step is mandatory, as the adaptation relies on the set of previous user's experiences. Without any experiences available in the case base, no adaptation can be performed initially.

3. Selecting the next *p* user experiences

Thirdly *the test experiences* are selected. To this end, the next *p* user experiences available in the data file are selected.

4. For each of these test experiences, performing the adaptation process

For each *test experience*, the adaptation process is performed. Hence, the problem part of the experience (context) is separated from the solution part (application behaviour). The adaptation process consists either of applying the cluster-based retrieval function or the Nearest Neighbour approach. This step ends with an application behaviour having been computed (marked in orange in Figure 9-1).

5. Comparing the result of the adaptation with the test experience's solution part

In the fifth step, the result of the *Adapt()* function for the test experience is computed, in comparing the results of the adaptation with the test experience's solution part. The result is either 1 (match) or 0 (no match).

6. Storing the test cases into the case base

Steps 4 and 5 are repeated for each *p* test experiences selected in step 3. When an Adapt() value for all *p test experiences* has been computed, Adaptation Correctness is in turn calculated (E 9.2). This result provides an indication of how good the adaptation is according to the current number of experiences in the user profile. Before completing this phase, the test experiences are added to the user profile.

Step 6 provides a result for the Adaptation Correctness obtained from the first *p test experiences*. At this stage, the user profile CB no longer contains *n* initial experiences but *n+p* initial experiences. In order to investigate the evolution of the Adaptation Correctness as the number of experiences in the user profile increases, the next p experiences in the test files are selected and the steps 4 to 6 are repeated.

This ends when the test file does not contain any further *p test experiences*. For example, for a data file containing 100 test experiences data, 8 iterations of the evaluation phase takes place. The first iteration is performed having only 5 experiences in the user profile, whereas for the last iteration, the user profile contains 85 experiences.

However, at this stage, the general procedure is performed on a single test file only. In order to gather a larger number of data, the general procedure is repeated k times, with changing test files. Thus, for all iterations we gather k values, characterising the Adaptation Correctness of the k files.

Considering the relative complexity of gathering user data (Section 9.1), it is difficult to prepare test files with different user data. Thus, new test files are obtained by randomly changing the order of the user data. As such, all files contain the exact same user data, but their order with the files differ. As the number of iterations grows, the user experiences available in the user profile differs, as do the test experiences that are employed at any iteration.

9.2.3 Generating the test data

In this section, the acquisition of the test data is detailed.

Test data are fundamental to the evaluation. Usually, data about users are obtained either by simulation or from test persons. In the context of this work, simulating the user preferences is, however, difficult. One could select a set of contexts and randomly assign user preferences to each of them. However, such data would not cope with the assumption we postulated in Chapter 4: “In similar contexts, users have similar needs with regards to the application”. An alternative is to rely on test persons to provide the data. However, as remarked in [Stah03], accurate human test persons are difficult to acquire in pure research environments. In addition, even the feature’s values for a set of contexts might be difficult to gather. The Call Profile Manager provides an example of the complexity of such a task. Even for an application, whose behaviours are relatively straightforward, the context elements may be difficult to capture. In a real-world environment, such features are not easily accessible, i.e. cannot be gathered without deploying a specific infrastructure.

To remedy these difficulties, we adopt the following approach to generate the test data. We manually determine a set of 100 real-world locations. Among others, locations corresponding to the author’s office environment, home environment, or public places in the vicinity were defined. Each single location was further *transformed* into contexts. This transformation implies that for each of these human perceived locations (e.g. my office) relevant and accurate values were defined for all context features of the Call Profile Manager application. The list of defined context is given in the Appendix. Table 9-1 indicates the possible values (or value range) the context elements defined as part of the Call Profile Manager specific-knowledge can take.

Feature	Values / value range	Comments
Position	Latitude Longitude	Value range
Function	e.g. Office, bar, hall, corridor,	Set of values available in the Wordnet taxonomy [WORD]
Ownership	public, manager, colleague, friend, family, private,	Set of values
Attention	Low, medium, high	Set of values
Noise level	From 20 to 120 (dB)	Value range

Table 9-1: Values and value ranges for the user experiences

For example, context nr 56 in the Appendix is written as:

56	51.314049, 9.491139	Theatre	Public	High	80	BLOCK

This context corresponds to the user being in a theatre hall in Kassel, Germany. The columns can be read as:

- **51.314049, 9.491139**: is the position (expressed in degrees) of the location
- **Theatre** is the name assigned to the location. Since the theatre room is specifically designated here, the term “theatre” suffices. However, e.g. a corridor in the close vicinity is meant, the best term to apply is no longer “theatre” but “corridor”.
- **Public**: indicates that the location can be accessed by unrelated persons.
- **High**, characterises the user’s level of attention (e.g. the user might be watching a movie)
- **80**, the level of noise expressed in dB.

At this stage, it might be interesting to point out again that the context features are not only meant to characterise spatial aspects of the user’s location (as position does). Noise level and attention level are not spatial information; rather, these two features are related to the location.

- The last column **Block** indicates that the user, when in this context, expects any incoming call to be blocked.

There is no “one correct way of behaving” for the application for all users. For example, one cannot say that the Call Profile Manager must block all incoming calls when the user is at the office. Rather, this depends exclusively on the user’s preference in the considered context. To determine, for each context, the user preference (i.e. the required application behaviour), an analytical process was pursued in considering the previous experiences and the contexts to come in the list. This analytical process has enabled the determination of some overall user behaviours for different contexts, thus leading to the determination of some identifiable patterns among the contexts. For example, in locations related to the user’s home environment, the user preference was mostly set to normal. Such patterns are clearly not unrealistic. But according to the considered user, they can be more or less distinguishable.

9.2.4 Results of the experiment

In this section, the results of the cluster-based retrieval function and those of the Nearest Neighbour methods are given.

Experiment settings

Following the objectives of the evaluation the experiment is performed that utilises the test data and follows the general evaluation procedure.

Numerous parameters have been defined in the different domains of this work: the Call Profile Manager application-specific knowledge, the cluster-based retrieval function and the evaluation procedure. In the following, we present their values of the parameters.

For the position similarity metric (see Section 7.2.4), we have selected

- D_{MAX} = 100 m

This way, the similarity of contexts, which are physically proximate, is increased.

Concerning important parameters of the evaluation, we have used constant values during all experiments.

- n = 5, the user profile is initially populated with 5 user experiences
- p= 10, a set of 10 test experiences are utilised to compute the Adaptation Correctness.
- k= 10, the general procedure is reproduced 10 times, using 10 different test file.

The cluster-based retrieval function, presented in Chapter 5 also relies on a set of parameters, the features' thresholds (5.3.3). These parameters are kept constant in the experiments.

- For the Position feature, $\theta_1 = 0.8$
- For the Semantic feature, $\theta_2 = 0.67$
- For the Ownership feature $\theta_3 = 1$
- For the Attention feature $\theta_4 = 1$
- For the Noise level feature $\theta_5 = 0.95$

Experiment

The objective of this experiment is to compare the Nearest Neighbour approach, employed in the approaches detailed in Chapter 3, and the cluster-based retrieval function, described in Chapter 5. Figure 9-2 illustrates the results obtained, with the set of parameters indicated in 9.3.2.

The x-axis corresponds to the number of user experiences available in the user profile, on which the adaptation relies. The x-axis ranges from 5 to 85. When the evaluation starts, 5 user experiences are stored within the user profile (which is empty at this stage). After each iteration, the test experiences are stored in the user profile. The last iteration with the last 10-experiences set is performed using 85 experiences in the user profile. The y-axis denotes the Adaptation Correctness. The notion of Adaptation Correctness is discussed in 9.2.1. The function ranges from 0 (no correct adaptation performed for any of the test experience) to 1

(all adaptations have been performed correctly).

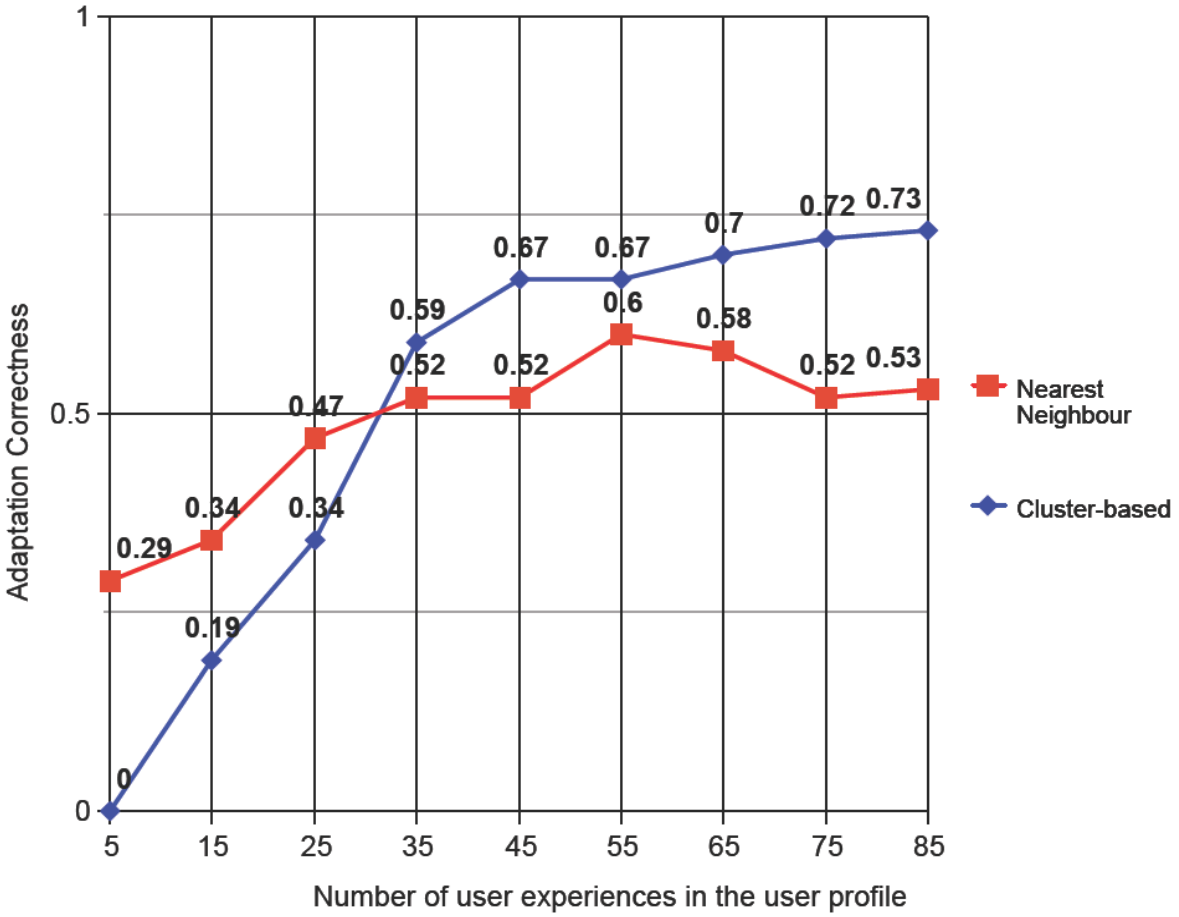


Figure 9-2: Evaluation results for experiment

In Figure 9-2, the results indicate that the cluster-based retrieval function performs relatively better than the Nearest Neighbour approach when the number of user experiences becomes greater than 30. In fact, one can distinguish between two distinct parts.

Initially the number of experiences available to perform adaptation is very limited (<35) and the Nearest Neighbour approach appears here to outperform the cluster-based retrieval function. In the first iteration, where only 5 user experiences are available, the retrieval function does not rely on enough user experiences to be able to compute any cluster. Even when several contexts are related to the same user preference, they appear to be too dissimilar to lead to the definition of a cluster. In the second iteration, as more user experiences are available, clusters can be defined. However, these clusters are not meaningful, i.e. they do not depict relevant context states. Hence, adaptations happening at this stage are in great part erroneous.

When user experiences become more numerous in the user profile, the behaviour of the cluster-based retrieval function is improved and constantly outperforms the Nearest Neighbour approach. This results from the fact that the more clusters are formed, thus

covering more numerous contextual states and increasing as such the probability for the new context to be related to a relevant contextual state. We must notice though that the Adaptation Correctness for the retrieval function increases in a relatively steady way.

The performance of the Nearest Neighbour is also interesting to consider. Surprisingly, the maximum value obtained by the Nearest Neighbour approach does not occur as the number of experiences is maximal in the user profile. In fact, it appears that the increase over a certain threshold (number of experience > 65) does not further improve the Adaptation Correctness. On the contrary, it is reduced. Besides, the performance of the cluster-based retrieval function should be primarily considered with regards to the one of the Nearest Neighbour approach. As such it is clear that this former approach constitutes an interesting candidate to be deployed in the retrievial phase. However, considering the results (not relative to another approach, but in an absolute way), it is doubtful that the Call Profile Manager application could be available as such out of the laboratory. Indeed, the best approach does “only” provide an Adaptation Correctness of 0.73, which means that – though the data are no real user data and may reflect more discernable patterns than those gathered in a real-world environment - almost 3 times out of 10 adaptation performs wrongly. Such an “error rate” might be too large for the application to become a product and garner user acceptance.

We see a common reason to the decrease of the Adaptation Correctness for the Nearest Neighbour method and the mitigated absolute performance of both approaches. They both rely very heavily on the specific-knowledge of the Call Profile Manager application, specifically the context features and similarity metrics. Hence, these results may show that the set of selected context features is either not complete (additional features must be added) and/or some are irrelevant (some features do not help distinguish among contexts). Also, the design of the similarity metrics might necessitate a review in order to better capture similar contexts. This provides an alternative and interesting demonstration of the importance of specific-application knowledge. As the primary objective of this work is not to develop any product and due to the limited resources available, we have not further improved this knowledge.

10 Conclusion

This chapter concludes this thesis by recapitulating the main results and pointing out some open questions. These open questions can be seen as interesting issues for future research and would contribute nicely to the development of product-mature contextual personalised applications.

10.1 Objectives and achieved results

In this thesis, we have presented a novel approach for handling contextual personalised applications. Context-awareness offers new opportunities for personalising applications in both the mobile and the desktop environments. Personalisation aims to provide users with applications that behave as expected by users, thus matching their preferences. In the area of context-awareness, some applications can be adapted to provide the behaviours the user expects with regards to his current context. Contextual personalised applications are thus defined as the subclass of context-aware applications that make use of the users' contexts to provide personalised behaviours. Using contextual personalised applications can increase the user acceptance of such applications, as the user's interactions (e.g. key strokes typed by the user) with the system required to govern the applications are reduced. Ultimately, this may result in a more effective use of the user's attention and concentration.

The main achievement of this work is the development of a framework, named Copernik (COntextual PERsoNalized applicatIon frameworK), which aims to support contextual personalised applications by providing guidelines to support contextual personalised applications.

10.1.1 Problem summary

The vision of pervasive computing [Saty01] calls for new types of applications providing (semi-) autonomous and personalised services. An approach to offer these new types of applications is context-awareness, a concept that promotes the reaction of applications to a user's context.

As defined in [Dey00] context is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves. Traditionally, information such as the user's location, the user's activity, people and objects surrounding the user and temporal information (time) is considered.

Context-aware applications can make use of context to adapt aspects of their structure, functionality or interface. In this work we specifically consider context-aware applications that are being personalised. We refer to these applications as contextual personalised

applications. Here, adaptation is governed by user models that contain information about the user's preferences, interests, etc, which are context-dependent. In various contexts a user can have different preferences. For example a user might be interested in cultural news when browsing the web at home, while he might prefer having scientific news when browsing the web from his office.

Current contextual-personalised applications suffer from two limitations:

- Adaptation in context mostly relies on a set of rules

  For a contextual personalised application to be adapted the following types of information are required: 1) user preferences, or what tasks the user expects to be performed – 2) the related context, or the situation in which these preferences are relevant. However, the traditional way to address these types of information consists of defining, before the system runtime, user preferences and their associated contexts and combining them into rules. This requires the user to think about his preferences in all the contexts in which he might find himself in the future. Furthermore, rules are not extendable. When a context occurs which has not been specified in any rule, no specific adaptation can be performed. Other relevant approaches relying on case-based reasoning do not address all the requirements of Contextual Personalisation.

- Applications have knowledge of user preferences and context descriptions

  Contextual personalised applications are governed by information such as the user's preferences and interests, which are contained in context-dependent user models. In the current realisations, and following the separation of concerns from [DeSa01], context-aware applications are further provided with context descriptions. Applications use them to trigger adaptation (e.g. modifying the output mode - text or audio - or the types of information it displays) based on the context-dependent user models. However, these two types of data, i.e. context and user preferences, are highly sensitive. Should this data become available to malicious parties, it could be very detrimental to the users.

10.1.2 Contributions

To overcome these limitations, an approach to Contextual Personalisation is then required, that 1) would reduce user interactions required to govern the adaptation while still enabling users to keep control over the applications and 2) prevent applications from obtaining knowledge of both the user's context and his preferences. The approach encompasses the following main contributions.

Definition of contextual personalised applications and development

The first novelty in this work concerns a new categorisation of context-aware applications, which highlights the peculiarities of contextual personalised applications. The categorisation

is discussed in Chapter 2. It distinguishes between two types of applications. The *Contextual Selection* category concerns applications that provide a set of operations independent of the user needs, which are triggered when some predefined context-based events occur. They do not rely on or make use of user preferences. *Contextual Personalisation* characterises applications which provide functionalities adapted to their users' needs. Though several categorisations were presented by distinct users in the literature before, the explicit definition of a class of application derived from personalisation and context-awareness fields alike, to the best of our knowledge, has never been made.

Moreover, in Chapter 7 we state what the application experts should take into consideration and what specific tasks they should undertake when developing a contextual personalised application.

Copernik: Framework for Contextual Personalisation applications

In this work a framework is introduced, the aim of which is to support contextual personalised applications.

The framework provides a basic structure defining policies and methods that govern the adaptation of contextual personalised applications. It describes a set of operations that are successfully performed to lead to the application's adaptation. The framework is governed by a set of principles.

The framework is founded on case-based reasoning, a technique of machine learning that allows the adaptation of application behaviours to the user's context. Employing case-based reasoning provides several advantages.

- Preferences are learnt during all user experiences with the application. Thus user changes in preferences can be acquired and used to further perform adaptation.
- A minimal set of interactions from the user with the application is required. Instead of indicating beforehand a set of preferences, the user can make use of the system directly, without inputting any information.
- Contexts that are handled by the system are not limited to a set of fixed, predefined ones. Instead, all incoming contexts, even those which were not foreseen, can be considered for performing adaptation of applications.

The framework mandates that users stay in control of the adaptation process. This means that ultimately users decide whether adaptation is acceptable to them as such or requires modification. User control is enabled through the following mechanisms:

- Providing explanations to the users about the application's adaptation.

  In order for the user to understand why the system has performed the adaptation, an explanation is provided to the user. The explanation consists of a human understandable

description of the current user's context that underlies the adaptation.

- Allowing users to give an opinion on the adaptation mode following any suggestion.

To do so, users are asked to provide feedback in several possible modes. The feedback is then assessed as either positive or negative.

Cluster-based retrieval function

The framework defines a set of phases to be performed in order to support contextual personalised applications. Perhaps, the most critical one includes the retrieving phase, which retrieves meaningful cases from the user's case base corresponding to similar previous user experiences.

In Chapter 5 we have identified the design principles that guide the development of a metric for retrieving cases which are similar to the current user's context. These include 1) similarity between contexts can be assessed differently for different users; 2) when assessing similarity between different pairs of contexts, the relevant features may differ; 3) the results of the retrieval function must be explained to the user. Considering these principles, we have developed a retrieval function.

This cluster-based retrieval function has been compared to the nearest neighbour approach. An evaluation was performed based on a data set relating contexts and user's preferences. The cluster-based retrieval function has been found to generally outperform the nearest neighbour approach.

Application-specific knowledge

Performing adaptation in context-aware systems and independent of application offers certain advantages. User data privacy is ensured and a central user model where information is stored in a non-redundant manner can be maintained to serve several applications.

However, adaptation arises differently for various applications. Applications have their own characteristics that influence adaptation. We refer to these characteristics as application-specific knowledge.

We reviewed some existing context-aware applications and discussed three types of application-specific knowledge that enable context-aware systems to perform adaptation of applications. These include 1) the behaviour description, detailing all the behaviours the application can display, 2) the context elements, describing the context elements the application can be adapted along and finally 3) the adaptation knowledge required by the reasoning mechanism that triggers the adaptation. We expressed the knowledge with templates that are communicated to the contextual personalised systems by the applications.

10.2 Outlook

Before contextual personalisation becomes part of our everyday life, research should be further performed to turn what is today a research work into the reality of products. In particular, work remains to be done. We briefly outlined those we found most pertinent to this work as the following:

- Improving the adaptation of contextual personalised application

We have observed in Chapter 9 the benefits brought by the retrieval function over the nearest neighbour approach. *A possible improvement direction could consist of computing context feature weights* in order to mitigate the impact of features on clustering, whose value range is narrow and therefore not as meaningful as large value-ranged features to determine contextual state.

- Supporting the application domain expert when developing the application-specific knowledge

Application domain experts are responsible for the development of contextual personalised applications. Their role, detailed in chapter 7, is to determine the application-specific knowledge and to design and implement applications. As discussed in section 7.1, determining application-specific knowledge implies the determination of application behaviours, context features and similarity metrics. While application behaviours are relatively simple to find from the application scenario and use cases, selecting the context features and developing local similarity metrics are more complex tasks. In this work, the selection of context features is performed based on the analysis of scenarios and use cases. However, *the benefits of methods relying on sets of laboratory measurements and user tests and online learning techniques*, which are discussed in 7.1.2, *merit further investigation*. In addition, the definition of a similarity is a creative process and is currently performed by domain experts having an understanding of the application domains. *An interesting research direction would be to investigate methods in order to learn these metrics, thus minimising the use of expert knowledge*.

- Deploying a “ubiquitous computing environment”

Contextual personalised systems, which are implemented following the principles and guidelines described in Chapter 4, rely on an infrastructure to handle context information. This infrastructure is responsible for providing means for gathering context information via sensors and processing these pieces of information to represent context in a usable form for the contextual personalised system. In addition, processing information has to be performed in such a way that the obtained context is complete – no feature information is missing – and correct – no feature information is false or uncertain. In the scope of this work, we have expressed these requirements as assumptions. However, they should be addressed. Thus, *methods are required to cope with noisy and missing sensor data*. Also

new approaches should be researched to compute values from sensor information for "untraditional" features. For example, in this work, we have defined a function feature, which refers to the purpose for which the location exists and is used.

- Performing more extensive experimental evaluations

The experimental evaluation performed in this work, and presented in Chapter 9, is suited to investigate the implemented algorithm of the retrieval function applied to users' past experiences. However, this evaluation is constrained by the difficulty we have faced to acquire evaluation data. The performed experiments were based on a set of simulated contexts and user preferences. In order to refine the experiment results, *it would be preferable to perform an evaluation on a larger user scale, i.e. with a large panel of potential users and in real situations*. In order to investigate how results are reproducible and ensure that the designed framework could cope well with various application specifications, *additional scenarios should be considered and realised*. Some concepts of contextual personalized applications have already been suggested in Chapter 7.

Hopefully, we will witness the results of fruitful research in these areas as an everyday user takes advantage of this technology.

The work in this thesis presents a general approach for supporting contextual personalised applications. As the path towards the realisation of Ubiquitous Computing scenarios progresses, it seems that Contextual Personalisation possesses huge potential for future users. Having information and services custom-designed to their preferences and needs of the moment will facilitate the use of these services and ultimately reach the aim of having computing structures "vanished into the environment" as presented in [Weis91]. The gap is large between laboratory demonstrators supported by paper-based scenarios and products available on a large scale. Only time will tell how this will evolve.

Appendix

Data set: User experiences

No	Position	Function	Ownership	Attention	Noise level	Behaviour
1	51.312980, 9.459104	Bedroom	Private	Low	38	VIBRATION
2	51.312980, 9.459154	Kitchen	Private	Low	45	NORMAL
3	51.312830, 9.459104	Living room	Private	Medium	65	NORMAL
4	51.313020, 9.459108	Corridor	Private	Low	40	NORMAL
5	51.313204, 9.459154	Garden	Private	Low	50	LOUD
6	51.313070, 9.459104	Bathroom	Private	Low	41	BLOCK
7	51.313000, 9.459164	Bedroom	Private	Low	32	VIBRATION
8	51.311677, 9.475257	Office	Manager	High	62	BLOCK
9	51.312204, 9.474858	Office	Private	Medium	54	VIBRATION
10	51.312154, 9.474858	Office	Colleague	High	58	VIBRATION
11	51.312104, 9.474860	Office	Colleague	High	57	VIBRATION
12	51.312054, 9.474858	Office	Colleague	Medium	62	VIBRATION
13	51.312054, 9.474958	Office	Colleague	Medium	61	VIBRATION
14	51.312156, 9.474961	Office	Colleague	Medium	65	VIBRATION
15	51.311577, 9.475258	Office	Manager	High	64	BLOCK

16	51.311467, 9.475357	Secretariat	Colleague	Low	63	NORMAL
17	51.311407, 9.475290	Meeting-room	Colleague	High	63	VIBRATION
18	51.311547, 9.475385	Corridor	Colleague	Low	46	NORMAL
19	51.311367, 9.475302	Kitchen	Colleague	Low	56	NORMAL
20	51.311300, 9.475287	Office	Colleague	Medium	63	VIBRATION
21	51.311561, 9.475297	Corridor	Colleague	Low	47	NORMAL
22	51.311727, 9.475457	Toilet	Colleague	Low	45	BLOCK
23	51.312107, 9.475287	Entrance Hall	Colleague	Low	56	NORMAL
24	51.312477, 9.475257	Canteen	Colleague	Low	75	LOUD
25	51.311300, 9.475657	Library	Colleague	High	40	BLOCK
26	51.311677, 9.475757	Classroom	Colleague	High	63	BLOCK
27	51.311642, 9.475657	Toilet	Colleague	Low	46	BLOCK
28	51.311999, 9.475457	Elevator	Colleague	Low	45	NORMAL
29	51.315363, 9.497047	Supermarket	Public	Low	75	LOUD
30	51.315663, 9.497047	Store	Public	Low	64	LOUD
31	51.315964, 9.497053	Store	Public	Medium	68	LOUD
32	51.315801, 9.497037	Store	Public	Low	67	LOUD
33	51.315653, 9.497847	Store	Public	Medium	66	LOUD

34	51.315700, 9.497167	toilet	Public	Low	45	BLOCK
35	51.315653, 9.497067	Corridor	Public	Low	75	LOUD
36	51.315473, 9.496987	Corridor	Public	Low	73	LOUD
37	50.942585, 6.955360	Reception	Public	High	72	VIBRATION
38	50.942885, 6.955430	Bedroom	Public	Low	48	VIBRATION
39	50.942885, 6.955480	Bathroom	Public	Low	43	BLOCK
40	52.503011, 13.327261	Bedroom	Public	Low	46	VIBRATION
41	52.503041, 13.327261	Corridor	Public	Low	47	VIBRATION
42	52.503011, 13.327301	Bathroom	Public	Low	44	BLOCK
43	52.503511, 13.327283	Barroom	Public	Medium	73	NORMAL
44	52.503395, 13.327321	Lobby	Public	Medium	72	VIBRATION
45	52.503384, 13.327221	Restaurant	Public	Medium	70	VIBRATION
46	48.869801, 2.307586	Car	Public	Medium	68	NORMAL
47	52.400097, 9.703048	Car	Public	Low	70	NORMAL
48	50.107445, 8.664747	Wagon	Public	Low	66	VIBRATION
49	49.892453, 8.657702	Wagon	Public	Medium	65	VIBRATION
50	50.897968, 6.971020	Car	Friend	Medium	70	NORMAL
51	51.529437, 9.893592	Car	Manager	Medium	71	VIBRATION

52	47.101539, -1.812382	Car	Family	Medium	69	NORMAL
53	50.808427, 8.772526	Car	Friend	High	71	BLOCK
54	50.897860, 6.951600	Car	Private	High	73	BLOCK
55	51.314532, 9.472960	Theatre	Public	High	82	BLOCK
56	51.314049, 9.491139	Theatre	Public	High	80	BLOCK
57	51.314349, 9.491115	Hall	Public	Low	74	LOUD
58	51.314179, 9.491129	Corridor	Public	Low	70	NORMAL
59	51.311780, 9.491395	Street	Public	Low	65	LOUD
60	51.314847, 9.472943	Street	Public	Low	67	LOUD
61	52.517344, 13.395743	Street	Public	Medium	68	LOUD
62	48.860106, 2.339628	Hall	Public	Medium	58	VIBRATION
63	48.860106, 2.339628	Museum/gallery	Public	Medium	63	VIBRATION
64	48.860106, 2.339628	Museum/gallery	Public	High	62	VIBRATION
65	48.860106, 2.339628	Museum/gallery	Public	High	59	VIBRATION
66	51.322357, 9.502040	Bar	Public	Medium	77	LOUD
67	47.215765, -1.558994	Bar	Public	High	82	LOUD
68	51.312885, 9.466583	Restaurant	Public	Medium	71	VIBRATION
69	51.316635, 9.500682	Restaurant	Public	High	73	VIBRATION

70	51.316593, 9.501357	Church	Public	Medium	58	BLOCK
71	48.853291, 2.348970	Cathedral	Public	Medium	65	BLOCK
72	41.009659, 28.967480	Mosque	Public	Medium	62	BLOCK
73	51.312980, 9.459104	garage	Private	Low	58	NORMAL
74	51.312143, 9.459009	Garage	Friend	Low	59	VIBRATION
75	51.312143, 9.459009	Living room	Friend	Medium	74	VIBRATION
76	51.312223, 9.459017	Hall	Friend	Low	65	VIBRATION
77	51.312643, 9.459169	Garden	Friend	Low	58	VIBRATION
78	52.525791, 13.314281	Hall	Colleague	Low	65	NORMAL
79	52.525791, 13.314281	Hall	Colleague	Low	64	NORMAL
80	52.525578, 13.314345	Canteen	Colleague	Low	77	LOUD
81	52.526099, 13.314279	Office	Colleague	High	72	VIBRATION
82	52.526087, 13.314481	Office	Colleague	High	70	VIBRATION
83	52.526179, 13.314281	Office	Colleague	Medium	70	VIBRATION
84	52.526379, 13.314318	Boardroom	Colleague	High	73	BLOCK
85	52.525981, 13.314211	Boardroom	Colleague	High	77	BLOCK
86	52.526501, 13.314310	Boardroom	Colleague	High	50	BLOCK
87	48.876141, 2.358283	Station	Public	Low	73	LOUD

88	48.144906, 11.581568	Station	Public	Low	67	LOUD
89	46.800979, -1.900794	Living room	Family	High	73	NORMAL
90	46.800872, -1.900792	Kitchen	Family	Low	42	NORMAL
91	46.801019, -1.900824	Bedroom	Family	Low	36	VIBRATION
92	46.800939, -1.900894	Bedroom	Family	Medium	44	VIBRATION
93	47.218901, -1.593732	Living room	Family	Medium	68	NORMAL
94	47.218951, -1.593812	Kitchen	Family	Low	58	NORMAL
95	51.312493, 9.492777	Hall	Public	Medium	65	NORMAL
96	51.304642, 9.500779	Park	Public	Low	60	LOUD
97	50.050777, 8.564129	Airport	Public	Low	68	LOUD
98	48.831932, 2.355539	Wagon	Public	Low	72	VIBRATION
99	52.522548, 13.415008	Corridor	Public	Low	68	LOUD
100	50.050777, 8.564129	Store	Public	Medium	70	LOUD

Underlying Frameworks: FAME2 and Foxtrot

As discussed in this thesis, the framework which is presented, aims to support contextual personalised applications. However, the functionality in charge of gathering and interpreting context information is not addressed in this work. Rather, it makes use of two additional frameworks developed at the Department for Communication Technologies at the University of Kassel, by other researchers[7].

[7] FAME 2 was developed by Björn Wüst as part of his dissertational work. Foxtrot was developed by Tino Löffler and Stefan Sigg, in the context of the bmb+f MIK21 Project [BMBP].

In this section the fundamentals of these frameworks are presented.

The Foxtrot and FAME2 Frameworks

Context-aware systems such as [Dey00] and [Henr03] have been developed to support context-aware applications. Applications can subscribe to predefined context sources, which can be plain sources, aggregators and interpreters. Foxtrot (Framework fOr ConteXT Model cOmpuTing) [Sigg08] extends this concept in allowing the distribution of support modules to various mobile computing devices in a ubiquitous computing environment. Indeed, the framework aims to serve for a variety of computing settings and scenarios, thus being subject to constant changes in its structure. By enabling the addition, removal or update of modules at runtime, based on a service oriented architecture approach, Foxtrot meets this fundamental requirement to context-awareness.

This is accomplished by adapting the architecture to the Framework for Applications in Mobile Environment (FAME2) [Wüst06]. FAME2 enables applications and services to be distributed between various devices and to be added, updated or removed at runtime.

Basically the context information flow is depicted in Figure 1.

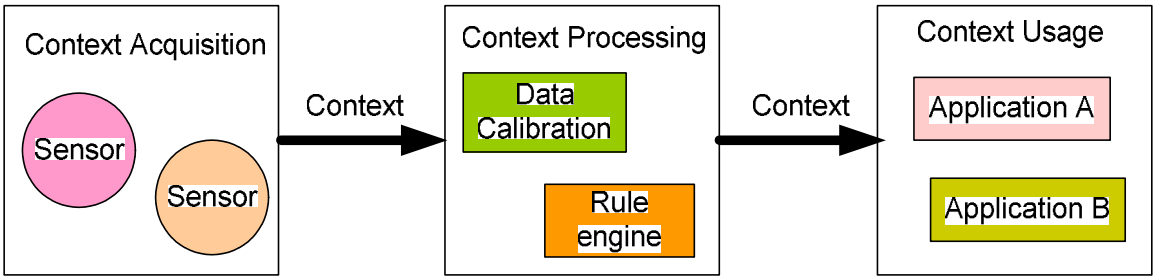


Figure A: Context information flow between the logical entities of Foxtrot (from [Sigg08])

The logical operations of context-aware systems can be divided into two distinct parts: the context acquisition part, the context processing part. Applications further make use of the information from the context processing part. In what follows, the two operations are briefly discussed, for information purpose only.

- In the context acquisition part, sensor information is obtained from physical or logical sensors which are represented as acquisition service.

An acquisition service is responsible for data acquisition in the architecture. Consequently it has output but no input ports. It has to implement at least one method: acquire(). In this method, measurements from a physical or logical sensor are obtained by a polling mechanism. The method acquire() is called periodically by the management unit. The result of this acquisition procedure is transformed to a context object. The context object containing the measured sensor information is afterwards stored in the buffers of all data sources of the acquisition service. A data source constitutes the output interface of the acquisition service. It is accompanied with buffer that holds all recently measured context information.

- This sensor information is, as context information, forwarded to the context processing part where the main processing of the architecture is disposed. In the context processing part of the architecture several different processing operations might be contended one after another. The context processing part constitutes the major contribution of Foxtrot.

Each data source is associated with a processing service. This processing service reads out the contents of the context buffer in regular intervals. The buffer is flushed afterwards.

Services that are responsible for context processing operations consist of three parts. Input ports, data sources and a processing step. Processing services receive context objects from other Foxtrot services at their input ports. From there, the received context elements are then further processed in the main part of the processing service, the processing step. A processing step processes the data contained in the context objects input to the processing step.

Also, data sources and input ports of different services can be connected.

- Context usage:

The high-level context obtained after the context processing step is transformed into an event and delivered to an event handler. Context aware applications can subscribe to the event handler and will be informed if a specific context occurs.

Foxtrot services are built using the FAME2 [Wüst06] framework. FAME2 enables adding, removing and updating of services at runtime, which eases the maintenance of the system. Additionally it allows us to integrate several service discovery technologies and communication protocols to access arbitrary services from any other device reachable.

FAME² is specifically designed for developing middleware that is able to be executed on mobile devices. It includes a development process to implement middleware services, and service execution environments (SEEs) that host (i.e. execute) the middleware services. The objective of the development process is to separate the concerns of developers of middleware services, and of developers of SEEs, which is often mixed today, resulting in unnecessary interdependencies between middleware services and their SEEs. FAME² is a a solution to implement middleware for distributed computing systems in ubiquitous computing with the following features:

- minimal resource use
- flexibility by enabling reconfiguration of middleware and its services at runtime
- a security concept on three different levels of access control: middleware, service, and operation
- an open interface to utilise virtually any existing service discovery that is used in middleware today
- online-update functionality of middleware services, including services that are in uses

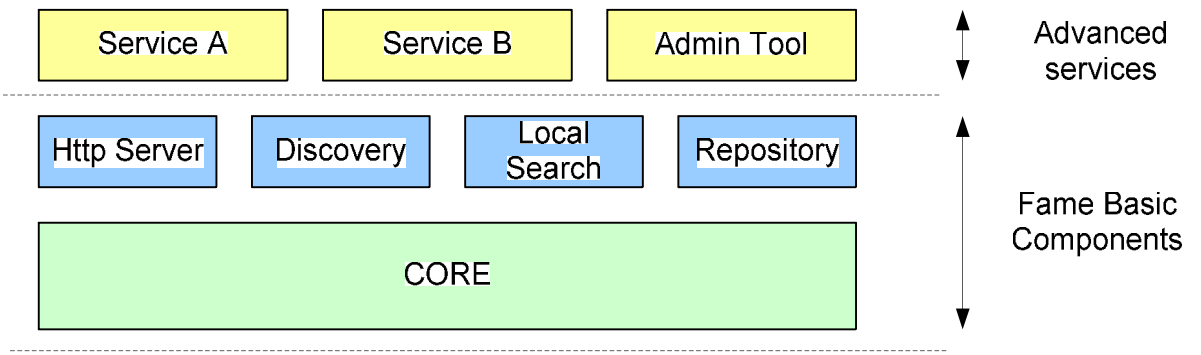


Figure B: Schematised representation of the FAME2 Framework

Integration of Contextual personalised systems

The system, whose architecture was presented in Chapter 8, can take advantage of these two frameworks as follows:

- The architecture can be realised as a context processing component. Thus it takes advantage of the context acquisition layer for gathering context. Besides it can also make use of additional implemented processing components in different scenarios.
- Foxtrot itself is integrated in the FAME 2 platform. Any context processing component can take advantage of the lifecycle management provided by the platform. Particularly interesting is the listening mechanism supported by the platform (i.e. an “eventing” mechanism) that has to be used by the User interaction component to monitor application behaviours.

References

[AaPl94] A. Aamodt, E. Plaza: Case-Based Reasoning: Foundational Issues Methodological Variations, and System Approaches. In Artificial Intelligence Communications, Vol. 7, Issue 1, pp. 39-52, 1994.

[AaGö+04] L. Aalto, N. Göthlin, J. Korhonen, T. Ojala: Bluetooth and WAP Push based location-aware mobile advertising system. In Proceedings of the 2nd International Conference on Mobile Systems, Applications and Services, pp 49–58, 2004.

[AbAt+97] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper: Cyberguide: A Mobile Context-Aware Tour Guide. In Wireless Networks, Vol. 3, pp. 421-433, 1997.

[Alle79] J. F. Allen: A plan-based approach to speech act recognition, Doctoral Thesis, University of Toronto, Toronto, Canada, 1979.

[AMAZ] www.amazon.com (last visited 24.02.2008)

[ArCo+01] L. Ardissono, L. Console, I. Torre: An adaptive system for the personalized access to news. In AI Communications, Vol. 14, Issue 3, pp. 129-147, 2001.

[BaDe03] L. Barkhuus, A. K. Dey: Is Context-Aware Computing taking Control away from the User? Three levels of interactivity examined. Technote in Proceedings of UbiComp 2003, 2003.

[BaDu07] M. Baldauf, S. Dustdar, F. Rosenberg: A Survey on Context Aware Systems. In International Journal of Ad Hoc and Ubiquitous Computing 2007, Vol. 2, Nr.4, pp. 263-277, 2007.

[BaFl05] A. Battestini, J.A. Flanagan: Modelling and simulating context data in a mobile environment. In Proceedings of the 1st Workshop on Context Awareness for Proactive Systems (CAPS), pp. 127–136, 2005.

[BeDu05] C. Becker, F. Duerr: On location models for ubiquitous computing. In Personal and Ubiquitous Computing, Vol. 9, Issue 1, pp. 20-31, 2005.

[BeIn+87] D. Benyon, P. Innocent, D. M. Murray: System Adaptivity and the Modelling of Stereotypes. In Proceedings of INTERACT 87 – 2nd IFIP International Conference on Human-Computer Interaction, pp. 245-253, 1987.

[BeKa+05] A. Bernstein, E. Kaufmann, C. Bürki, M. Klein: How Similar Is It? Towards Personalized Similarity Measures in Ontologies. In Proceedings of the 7. Internationale Tagung Wirtschaftsinformatik, pp. 1347-1366, 2005.

[BeRi+01] R. Bergmann, M. M. Richter, S. Schmitt, A. Stahl, I. Vollrath: Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. In Proceedings of the 9th German Workshop on Case-Based Reasoning, 2001.

[BeSt03] A. R. Beresford, F. Stajano: Location Privacy in Pervasive Computing. In IEEE Pervasive Computing, Vol. 2, Issue 1, pp. 46-55, 2003.

[BeTr+02] R. Bergmann, R. Traphöner, S. Schmitt, P. Cunningham, B. Smyth: Knowledge-intensive product search and customization in electronic commerce. In E-Business Applications, Advanced Information Systems, pp. 89-103, 2002.

[BiCa04] G. Biegel, V. Cahill: A framework for developing mobile, context-aware applications. In Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communication, 2004.

[BMBP] http://www.mik21.uni-kassel.de/ (last retrieved on 28.02.2008)

[BrBo97] P. J. Brown, J. D. Bovey, X. Chen: Context-aware Applications: from the Laboratory to the Marketplace. In IEEE Personal Communications, pp. 58-64, 1997.

[Brez99] P. Brézillion, Context in problem solving: a survey. In The Knowledge Engineering Review, Vol. 14, Issue 1, pp. 47-80, 1999.

[BrJo01] P. J. Brown, G. J. F. Jones: Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering. In Personal and Ubiquitous Computing, Vol. 5, Issue 4, pp. 253-263, 2001.

[Brus01] P. Brusilovsky: Adaptive Hypermedia. In User Modeling and User-Adapted Interaction, Volume 11, Number 1-2, pp.87-110, 2001.

[BuBa+83] B.G. Buchanan, D. Barstow, R. Bechtal, J. Bennett, W. Clancey, C. Kulikowski, T. Mitchell, D. A. Waterman: Constructing an Expert System. In Building Expert Systems. Technowledge Series in Knowledge Engineering, Addison-Wesley Publishing Company, 1983.

[Burk02] R. D. Burke: Hybrid Recommender Systems: Survey and Experiments. In User Modeling and User-Adapted Interaction, Vol. 12, Nr. 4, pp. 331-370, 2002.

[Cass04] J. Cassens: Knowing What to Explain and When. In Proceedings of the ECCBR 2004 Workshops, pp. 97-104, 2004.

[CaWo01] L. N. Cassel, U. Wolz: Client Side Personalization. In Proceedings of the Second DELOS Network of Excellent Workshop: Personalisation and Recommender Systems in Digital Libraries 2001.

[CBML] https://www.cs.tcd.ie/research_groups/mlg/CBML/index.php (last retrieved on 24.02.2008).

[ChDa+00] K. Cheverst, N. Davies, K. Mitchell, A. Friday, C. Efstratiou: Developing a Context-Aware Electronic Tourist Guide: Some Issues and Experiences. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pp. 17-24, 2000.

[Chen04] H. Chen: An Intelligent Broker Architecture for Pervasive Context-Aware Systems. Phd Thesis, University of Maryland, Baltimore County, 2004.

[Chen05] A. Chen: Context-Aware Collaborative Filtering System. In: Proceedings of the International Workshop on Location- and Context-Awareness 2005, pp. 244-253, 2005.

[ChKo00] G. Chen, D. Kotz: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

Available at: http://www.cs.dartmouth.edu/reports/abstracts/TR2000-381/ (last retrieved 24.02.2008)

[ChMi+01] K. N. Cheverst, K. Mitchell, N. Davies: Investigating Context-aware Information Push vs. Information Pull to Tourists. In Proceedings of Mobile HCI 2001: Third International Workshop on Human Computer Interaction with Mobile Devices, pp. 1-6, 2001.

[ChMi+02] K. N. Cheverst, K. Davies, N. Mitchell: Exploring Context-aware Information Push. In ACM Personal and Ubiquitous Computing. Vol. 6, Number 4, pp. 276-281, 2002.

[CoDo+04] L. Coyle, D. Doyle, P. Cunningham: Representing Similarity for CBR in XML. In Advances in Case-Based Reasoning, Proceedings of the 7th European Conference, ECCBR 2004, pp. 119-127, 2004.

[CoLa06] V. Coroama, M. Langheinrich: Personalized Vehicle Insurance Rates: A Case for Client-Side Personalization in Ubiquitous Computing. In Proceedings of the Workshop on Privacy-Enhanced Personalization, in CHI 2006 Conference on Human Factors in Computing Systems, pp. 56-59, 2006.

[CoLa+06] O. Coutand, S. L. Lau, M. Sutterer, K. David, O. Droegehorn: User Profile Management for Personalizing Services in Pervasive Computing. In Proceedings of the 6th International Workshop on Applications and Services in

Wireless Networks (ASWN), pp. 3-11, 2006.

[CoLo75] A. Collins, E. Loftus: A spreading activation theory of semantic memory. In Psychological Review, Vol. 82, pp. 407-428, 1975.

[CoSm+05] S. Consolvo, I. E. Smith, T. Matthews, A. LaMarca, J. Tabert, P. Powledge: Location disclosure to social relations: why, when, & what people want to share. In Proceedings of CHI 2005: Conference on Human Factors in Computing Systems, 2005

[CuDo+03] P. Cunningham D. Doyle, J. Loughrey: An Evaluation of the Usefulness of Case-Based Explanation. In Proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR 2005, pp.122-130, 2003.

[DaLö+05] W. Dargie, T. Löffler, O. Droegehorn, K. David: Composition of Reusable Higher-level Contexts. In Proceedings of the 14th IST Mobile & Wireless Communication Summit, pp. 19-23, 2005.

[DeAb00] A. K. Dey, G. D. Abowd: Towards a Better Understanding of Context and Context-Awareness. In the Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000), 2000.

[DeHa+04] A. K. Dey, R. Hamid, C. Beckmann, I. Li, D. Hsu: a CAPpella: Programming by Demonstration of Context-Aware Applications. In CHI 2004, ACM Conference on Human Factors in Computing Systems, CHI Letters Vol. 6, Issue 1, 2004.

[DeSa01] A. K. Dey, D. Salber, G. D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In the Human-Computer Interaction (HCI) Journal, Vol. 16, Issue 2-4, pp. 97-166, 2001.

[Dey00] A. K. Dey: Providing Architectural Support for Building Context-Aware Applications. PhD Thesis, College of Computing, Georgia Institute of Technology, 2000.

[Dey01] A. K. Dey: Understanding and Using Context. In: Personal and Ubiquitous Computing Journal, Vol. 5, Issue 1, pp. 4-7, 2001.

[DICT] http://dictionary.reference.com/ (last visited 24.02.2008).

[Dobs05] S. Dobson: Leveraging the subtleties of location. In Proceedings of Smart Objects and Ambient Intelligence, pp. 175–179, 2005.

[DoNi04] S. Dobson, P. Nixon: More Principled Design of Pervasive Computing Systems. In Engineering for Human-Computer Interaction and Design, Engineering Human Computer Interaction and Interactive Systems, Joint Working Conferences EHCI-DSVIS 2004, Hamburg, Germany, pp. 292-305, 2004.

[Dour04] P. Dourish: What We Talk About When We Talk About Context. In Personal Ubiquitous Computing, Vol. 8, Issue 1, pp. 19-3, 2004.

[ElNa00] R. Elmasri, S.B. Navathe: Fundamentals of database systems, Third Edition, Addison-Wesley, 2000, ISBN 0-201-54263-3.

[FaCl04] P. Fahy, S. Clarke: CASS – a middleware for mobile context-aware applications. In Proceedings of the Workshop on Context Awareness, MobiSys 2004.

[Fait03] L. Faith Cranor: I Didn't Buy it for Myself': Privacy and Ecommerce Personalization. In Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society, pp. 111-117, 2003.

[FiKo00] J. Fink, A. Kobsa: A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. In User Modeling and User-Adapted Interaction, Vol. 10, Nr. 2-3, pp. 209-249, 2000.

[FiKo02] J. Fink, A. Kobsa: User Modeling in Personalized City Tours. In Artificial Intelligence Review, Vol. 18 Issue 1, pp. 33-74, 2002.

[Fish01] G. Fisher: User Modeling in Human-Computer Interaction. In User Modeling and User-Adapted Interaction, Vol. 11, Nr. 1-2, pp. 65-86, 2001.

[Flan05a] J. A. Flanagan: Unsupervised clustering of context data and learning user requirements for a mobile device. In Proceedings of the 5th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05), pp. 155-168, 2005.

[Flan05b] J. Flanagan: Context awareness in a mobile device: Ontologies versus unsupervised/supervised learning. In Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, pp. 167-170, 2005.

[FlHi+03] J.A. Flanagan, J. Himberg, and J. Mäntyjärvi: A hierarchical approach to learning context and facilitating user interaction in mobile devices. In Artificial Intelligence in Mobile System 2003 (AIMS 2003) (in conjunction with Ubicomp 2003), pp. 66–73, 2003.

[FoDu92] P. W. Foltz, S. T. Dumais: Personalized Information Delivery: An Analysis of information Delivery Methods. In Communication of the ACM, Vol. 35, Issue 12, pp. 51-60, 1992.

[FrMa+05] E. Frias-Martinez, G. D. Magoulas, S. Y. Chen, R. D. Macredie: Modeling Human Behavior in User-Adaptive Systems: Recent Advances Using Soft Computing Techniques. In Expert Systems with Applications Vol. 29, Issue 5, pp 320-329, 2005

[GaHe+95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series, 1995. ISBN: 0201633612.

[GiHo80] M. L. Gick, K. J. Holyoak: Analogical problem solving. In Cognitive Psychology, Vol. 12, pp. 306-355, 1980.

[GoAm05] D. Godoy, A. Amandi: User Profing for Web Page Filtering. In IEEE Internet Computing, pp. 56-64, 2005.

[GöTh+06] M. H. Göker, C. A. Thompson, S. Arajärvi, K. Hua: The PwC Connection Machine: An Adaptive Expertise Provider. In Proceedings of the 8th European Conference, ECCBR 2006, pp. 549-563, 2006.

[GPSV] http://www.gpsvisualizer.com/geocode (last retrieved on 28.02.2008)

[Gree01] S. Greenberg: Context as a Dynamic Construct. In Human-Computer Interaction, Vol. 16, Issue 2-4, pp. 257-268, 2001.

[GuPu+04] T. Gu, H. K. Pung, D.Q. Zhang, “A middleware for building context-aware mobile services”, In Proceedings of IEEE Vehicular Technology Conference (VTC), Milan, Italy, 2004

[HaBl+05] G. de Haan, O. Blanson Henkemans, A. Aluwalia: Personal assistants for healthcare treatment at home. In Proceedings of the 2005 annual conference on European association of cognitive ergonomics, pp. 225 – 231, 2005.

[HaIn06] B. Hardian, J. Indulska, K. Henricksen: Balancing Autonomy and User Control in Context-Aware Systems – A Survey. In: 3rd International Workshop on Context Modelling and Reasoning (CoMoRea), PerCom’06 Workshop Proceedings, pp 51-56, 2006.

[Hase05] S. Haseloff: Context Awareness in Information Logistics. PhD Dissertation, Technical University of Berlin, 2005.

[HaSh+01] U. Hanani, B. Shapira, P. Shoval: Information Filtering: Overview of Issues, Research and Systems. In User Modeling and User-Adapted Interaction, Vol. 11, Nr. 3, pp. 203-259, 2001.

[HeAb06] M. Hefke, A. Abecker: A CBR-Based Approach for Supporting Consulting Agencies in Successfully Accompanying a Customer's Introduction of Knowledge Management. In Proceedings of the 8th European Conference, ECCBR 2006, pp. 534-548, 2006.

[HeIn+02] K. Henricksen, J. Indulska, A. Rakotonirainy: Modeling Context Information in Pervasive Computing Systems. In Proceedings of the 1st International Conference on Pervasive Computing (Pervasive), pp. 167-180, 2002

[HeIn05] K. Henricksen, J. Indulska: Personalising Context-Aware Applications. In OTM Workshop on Context-Aware Mobile Systems (CAMS), pp. 122-131, 2005.

[Henr03] K. Henricksen: A Framework for Context-Aware Pervasive Computing Applications. PhD thesis, School of Information Technology and Electrical Engineering, The University of Queensland, 2003.

[HiFl+03] J. Himberg, J. A. Flanagan, J. Mäntyjärvi. Towards Context Awareness Using Symbol Clustering Map. In Proceedings of the Workshop for Self-Organizing Maps 2003 (WSOM2003), pp. 249-254, 2003.

[HiKa+05] M. Hitchens, J. Kay, B. Kummerfeld, A. Brar: Secure Identity Management for Pseudo-Anonymous Service Access. In Proceedings of the 2nd International Conference on Security in Pervasive Computing, SPC 2005, pp. 48-55, 2005

[HiMä+01] J. Himberg, J. Mäntyjä rvi, P. Korpipää: Using PCA and ICA for exploratory data analysis in situation awareness. In Proceedings of the IEEE Conference on Multisensor Fusion and Integration in Intelligent Systems, pp. 127–131, 2001

[HoBr+98] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse: The Lumière project: Bayesian user modeling for inferring the goals and needs of software users. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 256–265, 1998

[HoLa04] J. Hong, J. Landay: An Architecture for Privacy-Sensitive Ubiquitous

Computing. In Proceedings of The Second International Conference on Mobile Systems, Applications, and Services (Mobisys'04), pp. 177-189, 2004.

[HoNg+04] J. I. Hong, J.D. Ng, S. Lederer, J. A. Landay: Privacy Risk Models for Designing Privacy-Sensitive Ubiquitous Computing Systems. In: Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, pp. 91-100, 2004.

[Hong05] J. I. Hong: An Architecture for Privacy-Sensitive Ubiquitous Computing, PhD Thesis, University of California at Berkeley, Computer Science Division, Berkeley, 2005.

[Höök00] K. Höök: Steps to take before IUIs become real. In Journal of Interacting with Computers, Vol. 12, Nr. 4, pp. 409-426, 2000.

[HoSc03] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, W. Retschitzegger. Context-Awareness on Mobile Devices – the Hydrogen Approach. In Proceedings of 36th Annual Hawaii International Conference on System Sciences (HICSS'03), pp. 292-301, 2003.

[IEEE00] IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

(available at http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html, last retrieved 24.02.2008)

[InSu03] J. Indulska, P. Sutton: Location Management in Pervasive Systems. In Proceedings of the Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing, Adelaide, Australia. CRPIT, Vol. 21, pp. 143-151, 2003.

[IsAl+03] L. Ishitani, V. Almeida, W. Jr. Meira: Masks: Bringing anonymity and personalization together. In IEEE Security & Privacy Magazine, Vol. 1, Issue 3, pp. 18–23, 2003.

[J2SE] http://java.sun.com/javase/index.jsp (last retrieved on 24.02.2008)

[JaCr+00] J. Jarmulak, S. Craw, R. Rowe: Genetic Algorithms to Optimise CBR Retrieval. In Proceedings of the 5th European Workshop on Case-Based Reasoning (EWCBR'2000), 2000.

[JiLa02] X. Jiang, J. A. Landay: Modeling Privacy Control in Context-Aware Systems. In IEEE Pervasive Computing, Vol.1, No. 3, pp. 59-63, 2002.

[KaPr+03] G. Kappel, B. Pröll, W. Retschitzegger, W. Schwinger: Customisation For Ubiquitous Web Applications a Comparison of Approaches. International Journal of Web Engineering Technology, Vol.1, Nr. 1, pp 79-111, 2003.

[KaRe+02] G. Kappel, W. Retschitzegger, E. Kimmerstorfer, B. Pröll, W. Schwinger. Towards a Generic Customization Model for Ubiquitous Web Applications. In Proceedings of the 2nd International Workshop on Web Oriented Software Technology (IWWOST'2002), 2002.

[KhCo06] A. Khalil, K. H. Connelly: Context-aware Telephony: Privacy Preferences and Sharing Patterns. In Proceedings of Computer Supported Collaborative Work (CSCW), 2006.

[KoAa06] A. Kofod-Petersen, A. Aamodt: Contextualised Ambient Intelligence through Case-Based Reasoning. In Proceedings of the 8th European Conference on Case-Based Reasoning (ECCBR 2006), pp. 211-225, 2006.

[Kobs01a] A. Kobsa: Generic User Modeling Systems. In User Modeling and User-Adapted Interaction Vol. 11, Issue 1-2, pp. 49-63, 2001.

[Kobs01b] A. Kobsa: Tailoring Privacy to Users' Needs (Invited Keynote). In M. Bauer, P. J. Gmytrasiewicz, J. Vassileva, (eds.): User Modeling 2001: 8th International Conference. Berlin – Heidelberg: Springer Verlag, pp. 303-313, 2001.

[Kobs07] A. Kobsa: Privacy-Enhanced Web Personalization. In The Adaptive Web: Methods and Strategies of Web Personalization, pp. 628-670, 2007.

[Kobs93] A. Kobsa: User Modeling: Recent Work, Prospects and Hazards. In Adaptive User Interfaces: Principles and Practise. Elsevier, 1993.

[Kofo06] A. Kofod-Petersen, Challenges in Case-Based Reasoning for Context Awareness in Ambient Intelligent Systems. In Workshop Proceedings of The 8th International Workshop on Case-Based Reasoning and Context Awareness (CACOA 2006), pp. 287-299, 2006.

[KoKo+03] P. Korpipää, M. Koskinen, J. Peltola, S.-M. Mäkelä, T. Seppänen: Bayesian approach to sensor-based context awareness. In Personal and Ubiquitous Computing, Vol. 7, Issue 2, pp. 113-124, 2003.

[Kork00] M. Korkea-aho: Context-Aware Applications Survey. Internetworking Seminar (Tik-110.551), Spring 2000, Helsinki University of Technology

Available at: http://users.tkk.fi/~mkorkeaa/doc/context-aware.html (last retrieved on 24.02.2008)

[KoSc03] A. Kobsa, J. Schreck: Privacy through Pseudonymity in User-Adaptive Systems. In ACM Transactions on Internet Technology, Vol. 3, Issue 2, pp. 149-183, 2003.

[Lang05] M. Langheinrich: Personal Privacy in Ubiquitous Computing – Tools and System Support. PhD thesis No. 16100, ETH Zurich, Zurich, Switzerland, May 2005.

[Lang99] P. Langley: User modeling in adaptive interfaces. In Proceedings of the Seventh International Conference on User Modeling, pp. 357-370, 1999.

[LaWi66] G.H. Lance, W.T. Williams: A General Theory of Classificatory Sorting Strategies. 1. Hierarchical Systems. In The Computer Journal, Vol. 9 pp. 373-380, 1966.

[Leak96] D. Leake: CBR in Context: The Present and the Future. In Case-based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park: AAAI Press/ MIT Press, pp. 1- 26, 1996.

[LoCu+05] R. Lopez De Mantaras, P. Cunningham, P. Perner: Emergent case-based reasoning applications. In The Knowledge Engineering, Vol. 20, Issue 3, pp. 325-328, 2005.

[LöSi+06] T. Löffler, S. Sigg, S. Haseloff, K. David: The Quick Step to Foxtrot. In Proceedings of the Second Workshop on Context Awareness for Proactive Systems (CAPS 2006), pp. 113-123, 2006.

[MaAg99] A. Madabhushi, J. Aggarwal: A Bayesian approach to human activity recognition. In Proceedings of the 2nd IEEE Workshop on Visual Surveillance, pp. 25–32, 1999.

[MäHi+01] J. Mäntyjärvi, J. Himberg, P. Korpipää, H. Mannila: Extracting the Context of a Mobile Device User. In Proceedings of 8th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine System, 2001.

[MaKi05] T. Ma; Y.-D. Kim; Q. Ma; M. Tang; W. Zhou: Context-aware implementation based on CBR for smart home. In Proceedings of IEEE International

Conference on Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), Vol. 4, pp. 112- 115, 2005

[McSh06] D. Mc Sherry: Completeness Criteria for Retrieval in Recommender Systems. In Proceedings of the 8th European Conference, ECCBR 2006, pp. 9-29, 2006.

[MERR] http://www.merriamwebster.com/ (last visited 24.02.2008)

[MiCh91] G. A. Miller, W. G. Charles: Contextual Correlates of Semantic Similarity. Language and Cognitive Processes, Vol. 6, Issue 1, pp. 1-28, 1991.

[MiKo04] M. Mikalsen, A. Kofod-Petersen: Representing and Reasoning about Context in a Mobile Environment. In Proceedings of the First International Workshop on Modeling and Retrieval of Context (MRC 2004), pp.25-35, 2004.

[Milo97] M. Milosavljevic: Augmenting the User's Knowledge via comparison. In Proceedings of 6th International Conference on User Modeling, UM97, pp. 119-130, 1997.

[NaKi+02] Y. Nakanishi, N. Kitaoka N, K. Hakozaki, M. Ohyama: Estimating communication context through location information and schedule information: a study with home office workers. In Extended abstracts of the 2002 Conference on Human Factors in Computing Systems, CHI 2002, pp 642–643, 2002.

[NuSa+06] P. Nurmi, A. Salden, S. L. Lau, J. Suomela, M. Sutterer, J. Millerat, M. Martin, E. Lagerspetz, R. Poortinga: A System for Context-Dependent User Modeling. In Proceedings of On the Move (OTM) Workshops, pp. 1894-1903, 2006.

[OjKo+03] T. Ojala, J. Korhonen M. Aittola, M. Ollila, T. Koivumäki, J. Tähtinen, H. Karjaluoto: SmartRotuaari – Context-aware mobile multimedia services. In Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia, pp 9–18, 2003.

[OlGr+05] J. Olson; J. Grudin; E. Horvitz: Toward Understanding Preferences for Sharing and Privacy. In Proceedings of the Conference on Human Factors in Computing Systems CHI '05, pp. 1985-1988, 2005.

[Orwa96] J. Orwant: For want of a bit the user was lost: cheap user Modeling. In IBM Systems Journal, Vol. 35, Issue 3-4, pp 398–416, 1996.

[OWL] www.w3.org/2004/OWL/ (last visited 24.02.2008)

[P3P] http://www.w3.org/P3P/ (last visited on 24.02.2008)

[PaBi97] M. Pazzani, D. Billsus, Learning and Revising User Profiles: The Identification of Interesting Web Sites. In Machine Learning, Vol. 27, pp. 313-331, 1997.

[Pasc98] J. Pascoe: Adding generic contextual capabilities to wearable computers. In The Second International Symposium on Wearable Computers, pp. 92-99, 1998.

[Pope05] R. Popescu-Zeletin, S. Arbanowski: In Technologies for the Wireless Future – Wireless World Research Forum (WWRF), Wiley, 2005, ISBN: 978-0-470-01235-2.

[Quil68] M. Quillian: Semantic Memory. In Semantic Information Processing, MIT Press, pp. 227-270, 1968.

[RaAl+04] A. Ranganathan, J. Al-Muhtadi, R. H. Campbell: Reasoning about uncertain contexts in pervasive computing environments. In IEEE Pervasive Computing, Vol. 3, No. 2, pp. 62-70, 2004.

[RaCa03] A. Ranganathan, R. H. Campbell. An infrastructure for context-awareness based on first order logic. In Personal and Ubiquitous Computing, Vol. 7, pp. 353-364, 2003.

[RaOu+05] M. Raento, A. Oulasvirta, R. Petit, H. Toivonen: ContextPhone – A prototyping platform for context-aware mobile applications. In IEEE Pervasive Computing, Vol. 4, Issue 2, pp. 51-59, 2005.

[RDF] http://www.w3.org/RDF/ (last visited 24.02.2008)

[RDFS] http://www.w3.org/TR/rdf-schema/ (last visited 24.02.2008)

[Resn99] P. Resnik: Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. In Journal of Artificial Intelligence Research (JAIR), Vol. 11, pp. 95-130, 1999.

[RiAa05] M. M. Richter, A. Aamodt: Case-based reasoning foundations. In The Knowledge Engineering Review, Vol. 20, pp. 203-207, 2005.

[Rich98] M. M. Richter. Introduction. In Case-Based Reasoning Technology, From Foundations to Applications, Lecture Notes in Artificial Intelligence, pp. 1-15, 1998.

[RoHe+02] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt: A Middleware Infrastructure for Active Spaces. In IEEE Pervasive Computing, Vol. 1, Issue 4, pp 74–83, 2002.

[Ross76] S. Ross: A First Course in Probability, Macmilan, 1976.

[Royc70] W. Royce: Managing the Development of Large Software Systems. In Proceedings of the IEEE Wescon, Vol. 26, pp.1- 9, 1970.

[SaGa+05] N. Sadeh, F. Gandon, O. B. Kwon: Ambient Intelligence: The MyCampus Experience. Technical Report CMU-ISRI-05-123, School of Computer Science, Carnegie Mellon University, July 2005.

Available at http://www.cs.cmu.edu/~sadeh/mycampus.htm (last retrieved on 24.02.2008)

[SaPo+05] A. Salden., R. Poortinga, M. Boudiz, J. Picault., O. Droegehorn, M. Sutterer, R. Kernchen, C. Räck, M. Radziszewski, P. Nurmi: Contextual personalization of a mobile multimodal application. In Proceedings of the International Conference on Internet Computing (ICOMP), 2005.

[Saty01] M. Satyanarayanan, Pervasive computing: vision and challenges. In IEEE Personal Communications, Vol. 8, Issue 4, pp. 10-17, 2001.

[ScAd93] B. N. Schilit, N. Adams, R. Gold, M. Tso, R. Want: The PARCTAB Mobile Computing System. In Proceedings of the Fourth Workshop on Workstation Operating Systems (WWOS-IV), pp. 34-39, 1993

[ScAd+94] B. Schilit, N. Adams, R. Want: Context-Aware Computing Applications. In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[ScBe+99] A. Schmidt, M. Beigl, H. W. Gellersen: There is more to context than location. In Computers & Graphics Journal, Elsevier, Vol. 23, No.6, pp. 893-902, 1999.

[ScCo+07] G. Schultz, O. Coutand, R. van Eijk, M. Miettinen: Privacy, Trust and Group Communications. Book Chapter in Enabling Technologies for Mobile Services: The MobiLife Book, ed. Mika Klemettinen, Wiley, 2007, ISBN: 978-0-470-51290-6.

[ScHi+02] B. N. Schilit, D. M. Hilbert, J. Trevor: Context-aware communication. In IEEE Wireless Communications, Vol. 9, Issue 5, pp. 46-54, 2002.

[ScTh94] B. N. Schilit, M. M. Theimer: Disseminating Active Map Information to Mobile Hosts. In IEEE Network, Vol. 8, Issue 5, pp. 22-32, 1994.

[SeVe+04] N. Seco, T. Veale, J. Hayes: An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In Proceedings of ECAI'2004, the 16th European Conference on Artificial Intelligence, 2004.

[Sigg08] Stephan Sigg: Development of a novel prediction algorithm and analysis of context prediction schemes. Unpublished PhD Thesis, University of Kassel, 2008.

[SiKa05] H. Si, Y. Kawahara, H. Kurasawa, H. Morikawa, T. Aoyama: A context-aware collaborative filtering algorithm for real world oriented content delivery service. In Metapolis and Urban Life Workshop, Ubicomp2005, 2005.

[SoCa04] F. Sørmo, J. Cassens: Explanation Goals in Case-Based Reasoning. In Proceedings of the ECCBR 2004 Workshops, pp. 165-174, 2004.

[Somm07] I. Sommerville: Software Engineering. International Computer Science Series, Addison-Wesley, 8th Edition, 2007, ISBN 13: 978-0-321-31379-9.

[Spal01] L. Spalazzi: A Survey on Case-Based Planning. In Artificial Intelligence Review, Vol. 16, Issue 1, pp. 3-36, 2001.

[Stah03] A. Stahl: Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning. PhD Dissertation, University of Kaiserslautern, 2003.

[Step01] C. Stephanidis: Adaptive Techniques for Universal Access. In User Modeling and User-Adapted Interaction, Vol. 11, Nr. 1-2, pp 159-179, 2001.

[StLi04] T. Strang, C. Linnhoff-Popien: A Context Modeling Survey. In Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004- The Sixth International Conference on Ubiquitous Computing, 2004.

[TeKo03] M. Teltzrow, A. Kobsa: Impacts of User Privacy Preferences on Personalized Systems a Comparative Study. In CHI-2003 Workshop "Designing Personalized User Experiences for eCommerce: Theory, Methods, and Research", 2003.

[ThKo03] S. Theodoridis, K. Koutroumbas: Pattern recognition. Academic Press, 2nd Edition, 2003, ISBN: 0126858756.

[TrHu04] K. N. Truong, E. M. Huang, G. D. Abowd. CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home. In the Proceedings of UBICOMP 2004: The 6th International Conference on Ubiquitous Computing, pp. 143-160, 2004.

[Tver03] A. Tversky. Features of Similarity. In Preference, Belief, Similarity: Selected Writings by Amos Tversky. The MIT Press, pp. 7- 45, 2004.

[UMUA] http://www.umuai.org/ (last visited 24.02.2008)

[VaBo+03] Handbook of Privacy and Privacy-Enhancing Technologies, The Case of Intelligent Software Agents, Editors: G.W. van Blarkom RE, drs. J.J. Borking, dr.ir. J.G.E. Olk, Publisher: College bescherming persoonsgegevens, 2003, ISBN 90 74087 33 7

Available at: http://www.andrewpatrick.ca/pisa/handbook/handbook.html

(last retrieved 24.02.2008)

[VaCa00] K. Van Laerhoven, O. Cakmakci: What Shall We Teach Our Pants? In Proceedings of the 4th International Symposium on Wearable Computers (ISWC 2000), pp. 77-83, 2000.

[WeAh95] D. Wettschereck, D. W. Aha: Weighting Features. In Proceeding of the 1st International Conference on Case-Based Reasoning (ICCBR'95), 1995.

[Weib02] S. Weibelzahl: Evaluation of Adaptive Systems. PhD Dissertation, University of Trier, Germany, 2002

[Weis91] M. Weiser: The Computer for the 21st Century. In Scientific American, pp. 66-75, 1991.

[WePa+01] G. I. Webb, M. J. Pazzani, D. Billsus: Machine learning for user Modeling. In User Modeling and User-Adapted Interaction, Vol. 1, Nr. 1-2, pp. 19-29, 2001.

[WGS84] http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html (last retrieved 24.02.2008).

[WORD] http://wordnet.princeton.edu/ (last visited on 24.02.2008).

[Wüst06] B. Wüst, Framework for middleware executed on mobile devices. PhD Thesis, University of Kassel, 2006.

[WSDL] http://www.w3.org/TR/wsdl (last visited 24.02.2008).

[YAHO] www.yahoo.com (last visited 24.02.2008)

[YoBa97] J. Yoder, J. Barcalow: Architectural Patterns for Enabling Application Security. In Proceedings of PloP 97, The 4th Pattern Language of Programming, 1997.

[YoBa97] J. Yoder, J. Barcalow: Architectural Patterns for Enabling Application Security. In Proceedings of PloP 97, The 4th Pattern Language of Programming Conference, Monticello, Illinois, USA, September 3-5, 1997.

[ZuAl01] I. Zukerman, D. W. Albrecht: Predictive statistical user models for user Modeling. In User Modeling and User-Adapted Interaction, Vol. 11, Nr. 1-2, pp. 5-18, 2001.