Communication and Adaptation in a Ubiquitous Environment
Abstract

Context awareness means sending the right information to the right user at the right time. Context is our environment, which can be anything around us such as location, lights, noises etc. To make the context interact with our mobile devices or sensors, there must be protocols for communication and data formats for the “sent” or “received” contextual information so we can give very specific context information to the user. Since this communication and adaptive part is not well understood, in this paper we are interested in investigating the technology used for adaptation. We will also explain how this technology works to adapt itself to changes in the environment.

Keywords: ubiquitous, context aware computing, protocols, data formats, middleware, web services, adaptive systems
Preface

I would like to thank all the teachers and supervisors in my bachelor program for their help, guidance and patience that allowed me to achieve this last step in my program. I especially want to thank Ola Petersson who guided me during this whole thesis project and provided good tips and regular meetings. Finally, thanks are also due to my family for their never-failing support. I hope you will enjoy reading my thesis work. I found that my topic was interesting to investigate, since I did not have any knowledge whatsoever about it before I started my research.
# Contents

## List of Figures

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Related work</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Problem formulation</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.5 Social Aspect</td>
<td>2</td>
</tr>
<tr>
<td>1.6 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>1.7 Scope/Limitation</td>
<td>3</td>
</tr>
<tr>
<td>1.8 Target group</td>
<td>3</td>
</tr>
<tr>
<td>1.9 Outline</td>
<td>3</td>
</tr>
<tr>
<td>Method</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Scientific Approach</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Systematic Literature Review</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Interview</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Ethical consideration</td>
<td>6</td>
</tr>
<tr>
<td>Context Awareness</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Defining context</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Context-aware computing</td>
<td>7</td>
</tr>
<tr>
<td>Context-Aware Middleware in Adaptive Systems</td>
<td>9</td>
</tr>
<tr>
<td>4.1 Communication Component</td>
<td>10</td>
</tr>
<tr>
<td>4.2 Adaptation Component</td>
<td>11</td>
</tr>
<tr>
<td>4.2.1 Web Services</td>
<td>12</td>
</tr>
<tr>
<td>4.2.2 REST</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Context Component</td>
<td>13</td>
</tr>
<tr>
<td>4.3.1 Context Acquisition Layer</td>
<td>13</td>
</tr>
<tr>
<td>4.3.2 Context Aggregation Layer</td>
<td>14</td>
</tr>
<tr>
<td>4.3.3 Context Reasoner Layer</td>
<td>15</td>
</tr>
<tr>
<td>4.3.4 Application Layer</td>
<td>15</td>
</tr>
<tr>
<td>Context Aware Middleware</td>
<td>16</td>
</tr>
<tr>
<td>5.1 Some Existing Middleware</td>
<td>16</td>
</tr>
<tr>
<td>5.1.1 Aura</td>
<td>16</td>
</tr>
<tr>
<td>5.1.2 SOCAM</td>
<td>16</td>
</tr>
<tr>
<td>5.1.3 CAPNET</td>
<td>17</td>
</tr>
<tr>
<td>5.1.4 CARMEN</td>
<td>17</td>
</tr>
<tr>
<td>5.1.5 CARISMA</td>
<td>17</td>
</tr>
<tr>
<td>5.1.6 GAIA</td>
<td>17</td>
</tr>
<tr>
<td>5.1.7 CORTEX</td>
<td>18</td>
</tr>
<tr>
<td>5.2 Evaluation of Context-Aware Middleware</td>
<td>18</td>
</tr>
<tr>
<td>Reference Architecture</td>
<td>19</td>
</tr>
<tr>
<td>Discussion</td>
<td>21</td>
</tr>
</tbody>
</table>
8 Conclusion and Future Work

A Appendices
   A.1 Interview Questions ........................................... A
   A.2 Component Diagram for "SwipeEat" .......................... B
   A.3 Diagrams of the Scenarios ................................. C

References .................................................. G
List of Figures

4.1 Adaptation System Architecture ........................................ 10
4.2 Communication Technologies ........................................... 11
4.3 Web Service Main Standards ........................................... 12
4.4 Growth of REST over SOAP ............................................ 13
4.5 Context Handling Architecture ........................................ 14
5.6 Aura Architecture ....................................................... 16
5.7 Evaluation of Context Aware Middleware ............................ 18
6.8 Reference Architecture for "SwipeEat" ............................... 19
7.9 General Use Case ....................................................... 21
7.10 CORTEX Flow ......................................................... 22
7.11 Component Diagram for CORTEX ................................... 23
7.12 Sequence Diagram for CORTEX ..................................... 23
1.13 Component Diagram for "SwipeEat" ................................. B
1.14 "List Matching" Use Case ............................................ C
1.15 Sequence Diagram for "List Matching" Scenario ................... D
1.16 "Matching by Notification" Use Case ............................... E
1.17 Sequence Diagram for "Matching by Notification" Scenario ....... F
1 Introduction

Suppose you go to the city for some shopping and step into mall of Scandinavia with your phone in your pocket. This phone has a middleware installed in it. After a few minutes it detects a wireless network and connects to it. While you are walking around the mall the phone detects a web service, and, since the middleware knows your preferences, it informs you of interesting discounts in some video game stores. While you continue walking around the mall, the phone notifies you again about discounts on sushis proposed by another store in the mall.

Context aware applications (CAA) have as a goal to provide precise information to the user depending on her preferences. To achieve this objective, CAA must use some technologies to be able to process the context information and give the user the adequate information. It also uses some protocols for communication with the context provider, and data formats to put the contextual data in an understandable format to be shown to the end user. Since CAAs operate in a highly distributed heterogeneous environment, it makes the adaption of CAA to the different protocols and data formats harder. This has caused some middleware to appear in order to make the adaption softer without the need of end user interaction.

1.1 Background

The ubiquity and connectivity of software systems cause a context variability that can be utilized in software to provide services tailored to the user’s context and preferences. Protocols, data formats and different types of communication technology exist to enable CAAs to provide services to the user. There is also much middleware that facilitates the adaptation to the varying environment. It is thus able to process the context information gathered, so the end user does not have to interact with the application when changing environments. The objective of middleware is to make the adaptation to the environment easier, which is done by covering a number of different properties that we will discuss later in this paper. Each type of middleware emphasizes one specific or several properties. There is no middleware which is the best choice for all applications but your particular goals should determine which one to choose.

1.2 Related work

Philippe Roose et al. [1] give us a detailed explanation of adaptive systems. They explain how the adaptive systems work and show their different components. They focus on the middleware part by decomposing it into components, namely context, communication and adaptation, and explain the technologies used to make them work properly. They conclude by giving an evaluation table of the technologies used by different types of middleware.

Marco Bessi and Leonardo Brun [2] give an overview of traditional middleware systems and their incapacity to support CAAs. They emphasize the importance of creating context aware middleware that can support new requirements imposed by mobility. They also give an overview of the approaches used to create context aware middleware. They conclude by giving a current state-of-the-art in middleware for CAAs.

Daniel Romero [3] gives us a list of properties that context-aware middleware should cover. He also provides us a table that shows which middleware fulfills which properties.
He lists some types of middleware and gives us a short description and characterization of them. He concludes by giving an evaluation table of the middleware dealt with earlier in his paper.

Punnarumol Temdee and Ramjee Prasad have written a book [4] which I can recommend to any person who is new in the world of context awareness. This book, which is very well structured, puts you directly into the world of context aware computing by giving a very good definition of context awareness and different kinds of context. It also shows how CAAs are designed, and describes how the process of retrieving context is done. It mentions how communication is made between the applications and the different kinds of sensors available in the environment. It talks about the security in CAAs and describes their vulnerabilities and how to maximize their security. Finally they give a list and short descriptions of context aware middleware and applications.

Hong-Linh Truong and Schahram Dustdar [5] give a state-of-the-art description of context-aware techniques for Web services. It includes an explanation of Non Web Service-based Context-aware Systems and Web Service-based Context-aware Systems where they mention some middleware made for both kinds of systems. They give a concrete explanation of how context-aware systems in web service environments work, including how context is stored, distributed, reasoned, sensed, adapted and represented. In addition they deal with privacy and security techniques for context. They also give a list of existing surveys on context-aware systems and frameworks.

1.3 Problem formulation

The systems that provide context, that is, share context data, use different protocols for discovery and sharing, and data is represented in different forms. This requires that CAAs are able to recognize and adapt to these formats and protocols. It also uses different technologies to process the context data retrieved from the environment. How to do this efficiently is currently not well understood. Thus the goal of this project is to investigate the state-of-the-art in the literature, and present a classification of different kinds of middleware used for adaptation. After that we will develop a reference architecture for CAAs.

1.4 Motivation

Nowadays the use of CAAs is increasing so fast that our lives can be put in danger if these technologies are not used or carried out efficiently. We can take as an example a recent accident that happened in the USA, where a person was following a malfunctioning GPS that led him into a lake [6]. That is why it is important to understand how to make CAAs work efficiently, a problem this paper attempts to address. Moreover context-aware computing is also used in malls for advertising purposes, and it is important for these technologies to be used in the most efficient way to make the advertising process work perfectly when it comes to targeting users’ preferences.

1.5 Social Aspect

As the use of computers, smartphones and the world-wide web has increased in the last decades, computing has more and more been considered as a technology of mass consumption and marketing [7]. The use of context to get better information about the user’s
preferences may raise the issue of data privacy. The data retrieved from the user’s smartphone such as location, preferences and any kind of personal information might be saved and used subsequently to improve the quality of service and better target the preferences of the device’s owner [8]. According to the studies conducted by Höskuldur Borgthorson et al. [7] and Janne Lindqvist et al. [9], it has been proven that the users generally are unaware of the fact that data are collected by the applications installed in their smartphones. They actually express surprise and some discomfort when they find out. It is of great importance that application developers provide clear information to the users about how their personal data will be dealt with. P. Joseph Charles et al. [10] provide several techniques for privacy control that will enable the developers to better handle the users’ personal data.

### 1.6 Objectives

<table>
<thead>
<tr>
<th>O1</th>
<th>Conduct a literature review about context, context aware computing and adaptive systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2</td>
<td>Conduct a literature review about the types of communication used in adaptive systems</td>
</tr>
<tr>
<td>O3</td>
<td>Conduct a literature review about the technologies used for adaptation</td>
</tr>
<tr>
<td>O4</td>
<td>Synthesize the findings from the literature reviews and classify types of communication used in context aware computing</td>
</tr>
<tr>
<td>O5</td>
<td>Synthesize the findings from the literature reviews and explain how CAAs adapt to different environments</td>
</tr>
<tr>
<td>O6</td>
<td>Provide a classification and evaluation of the different technologies used for adaption</td>
</tr>
<tr>
<td>O7</td>
<td>Design a reference architecture for a CAA</td>
</tr>
</tbody>
</table>

### 1.7 Scope/Limitation

In this paper we are interested in making a clear explanation of the different technologies used in CAAs. We will explain how these technologies also can process the context information retrieved after a change in the environment happens. To achieve this goal, we are going to investigate the different components of adaptive systems, present a state-of-the-art in the literature and give a brief description of different existing types of middleware, followed by an evaluation table of this middleware. To conclude, we will create a reference architecture of a CAA based on the findings of this research. On the contrary, this thesis will not have as a goal to create a new protocol, data format or middleware for context-aware computing.

### 1.8 Target group

This work can be of interest for software engineers or network engineers, for a more holistic view of how communication, adaptation and context processing in CAAs is carried out. It will also attempt to give a better understanding of some existing middleware.

### 1.9 Outline

In section 2, we describe the methods used to achieve the objectives of this thesis. In section 3, we give a definition of context and context awareness. We also explain different kinds of context-aware computing. In section 4, we explain what dynamic adaptation
consists in and explain different parts of the adaptive system such as communication and context, etc. In section 5, we give a list of different types of middleware, a brief description of their characteristics, ending by an evaluation. In section 6, we provide a reference architecture for an application. In section 7, we discuss the reference architecture and the choice of middleware. Finally, in section 8 we present a conclusion and future work that can be considered.
2 Method

In this section, we describe the methods used to achieve the objectives of the thesis.

2.1 Scientific Approach

The goal of this thesis being to present information that has been acquired from other papers, the diversity of this information conducted us to make an investigation to find the data that would be relevant for this thesis. A Systematic Literature Review (SLR) was carried out to constitute the data collection and single out the most useful papers.

2.2 Systematic Literature Review

The first thing to do when conducting an SLR is to establish the questions that need to be answered. The three first objectives specified in 1.6 are the same that have been used in the SLR. Below, we present them as research questions:

- RQ1: What is context, context aware computing, and adaptive systems?
- RQ2: What are the types of communication used in adaptive systems?
- RQ3: What are the technologies used for adaptation?

A strategy to find and extract the relevant information that will answer the questions has to be established. The following process was used:

1. Selection of information sources: To answer the research questions, we did not limit ourselves to any particular libraries. We looked for all relevant papers or books from known online digital libraries provided by university such as IEEE, Science direct, ACM etc.

2. Keyword Finding: The subject of the thesis and the research questions specified were used to determine the keywords that should be employed. Below, we present a list of some keywords used, related to each research question.

   - RQ1: Context, context awareness, adaptive systems, context aware computing, ubiquitous environment.
   - RQ2: Communication in adaptive systems, communication in middleware.
   - RQ3: Adaptation in adaptive systems, context aware middleware.

These keywords mentioned above have been used in several different expressions to reach better results.

3. Exclusion Strategy: The articles were first selected based on their titles, publication dates and keywords. The criteria were that the titles should make a direct or indirect reference to the keywords mentioned above. Then, the dates of publication were checked, and articles more than five years old would be discarded. However some older articles were taken into account to give a better background to the subjects concerned. The last two steps consisted in reading and examining the abstract of each paper found, in order to narrow down the number of possible articles. Finally, all candidate papers were read and either kept for later reference or discarded as irrelevant. Overall, the study started with 80 papers which were further reduced to 30.
4. **Retrieve new sources from references (snowballing):** Some of the 30 papers kept had some relevant and interesting references that we read and examined and sometimes kept as references.

5. **Classify the references:** Finally, the relevant sources were classified by topics and publication dates for easy access.

2.3 **Interview**

Since I was new in the field of context-aware computing and adaptive systems, I took advantage of an opportunity to contact a Tunisian professor whose Ph.d.thesis principally deals with the human-machine interaction and related services in a ubiquitous environment. The interviewee was contacted at the beginning of my thesis research after finding some basic information about my topic. The goal of the interview was to make sure that I was on the right track for my research and to obtain some advice for the upcoming thesis. The interview was done via phone call, and the list of questions can be found in Appendix A.1.

2.4 **Ethical consideration**

The interview has not been recorded and only the valuable information was transcribed without any private information.
3 Context Awareness

Context aware computing is the use of software and hardware in order to deliver relevant information to the user depending on her environment and preferences.

3.1 Defining context

The definition which best describes context in context-aware computing is: It is any information that can be used to characterize 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 [11]. Context can be split into four categories [4, 11]:

- Computing context: Everything related to communication, network connectivity and bandwidth. It also includes local computing resources such as printers, displays, etc.
- User context: Any information related to the user itself which can be location, user profile, social situation.
- Physical context: Lighting, noise, traffic conditions, temperature, etc.
- Temporal context: Anything related to time, such as day, month, year and time of the day.

From a different perspective, context could be divided into operational and conceptual context [4]:

- Operational context: It is related to the operation of the system which involves context acquisition, modeling, and processing.
- Conceptual context: it tackles meaning and relationships between contexts.

3.2 Context-aware computing

To give the adequate and suitable information to the user, a context aware system must merge context, user and computing data to be able to provide the best services. Context plays an important role in providing the most suitable information to the user through available context information. Context-aware systems can be in two modes, the configuration mode and execution mode [4]. They are slightly different, the execution mode being the fact that the application will execute the action by itself according to the context, while the configuration mode is set in advance by the user to be behave in a certain way in a certain situation. These two modes can be active or passive, which we are going to define below [4].

- Active execution: The application or system will act by itself according to the environment, which is called a self-adaptive system. An example can be that while calling in a noisy place the sensibility of the microphone will change by itself according to the degree of noise in the environment and without the involvement of the user.
- Passive execution: Here the user will be involved by setting some rules that tell the application how to behave in some particular situations.
• Active configuration: This mode does not require any involvement of the user, the application will configure itself by knowing the preferences and habits of the user so it will know how to behave in each particular situation.

• Passive configuration: The application expects the user to provide manually her preferences and tasks which gives a better control of the application. It takes away some responsibility from the developers, but on the other side, they oblige the user to cooperate with the application.
4 Context-Aware Middleware in Adaptive Systems

In this section we are interested in clarifying how CAAs adapt to the dynamic environment. We assume that the user is always moving and wants her application to adapt to the environment without any interaction. We are going to describe in particular the adaptation layer and its component. We will describe the different kinds of communication used, investigate the service-based technology to achieve the adaptation and lastly we will show how the context data is interpreted to match the user’s preferences.

Middleware plays an important role in facing various challenges such as: service discovery, mobility, environmental changes and context retrieval. Its goal is to hide heterogeneity, support the deployment of sensors and collect context information. Although many types of middleware share common goals, all of them have some distinguishing unique features that make them different from each other and very powerful for the targeted applications [4]. To be able to face these challenges and offer the best user experience, any type of middleware must satisfy some properties. A list of requirements that middleware should fulfill can be found under [3, 12]. Below follows a description of the requirements we considered the most relevant for our research.

- Context Management: It is responsible for context retrieval and processing, which means that it has to deal with different kinds of context providers, data representation and communication. According to our scenario presented at the beginning of the paper, the context retrieval aims at obtaining the necessary information related to the user’s preferences and the wireless network.

- Adaptation: It is related to the actions that must be performed when changes are observed in the context. In our scenario the middleware adapted to the service found and notified the user about the discounts in the stores.

- Communication: One of the most important requirements for middleware is supporting different kinds of protocols, communication technologies and interaction paradigms.

- Service Discovery: It is the ability to discover and invoke services available in the environment. It uses service discovery protocols that define how the service will be advertised and discovered.

- Context Abstraction: It is about how context is expressed and formalized. A higher abstraction level can result in better understanding and use of context.

- Fault Tolerance: It is about how the middleware adapts and behaves when an unexpected fault occurs. Some may just stop when a fault occurs, and others may continue working and get rid of the failure.

Figure 4.1, inspired by [1], shows the different components of an adaptive system. Below, we describe more in depth the three components of the middleware layer that are communication, adaptation and context.
4.1 Communication Component

This part of the adaptation system is dealing with the transfer of data from a device to another. To reach this goal, communication must be established between the provider and the requester and this can be done by using different kinds of communication mechanisms. Below we are going to describe the most used communication mechanisms [1].

- Event-based Communication: It is characterized by the fact that the publisher and subscriber are decoupled which means that they have no information about each other [1, 13]. Event-based systems are topic based and content based, the event notification service is responsible for matching notifications with subscriptions [13]. The advantage of this communication is that the device wakes up just when an event is triggered which gives the opportunity to the device to save resources [14].

- TupleSpace Communication: This kind of communication is based on a shared memory space that the provider and requester can access at any time. The provider can push data into the space which is called tuples and the requester can get these tuples from the same space. This technology permits to the provider and requester
not to be connected to each other at the same time, which permits saving resources [15, 1].

- Connector-based Communication: Connectors provide a unified I/O interface. This kind of communication allows the application not to be concerned by remote communication. The connector can make the connection between two devices (A) and (B) through Bluetooth. In case the bluetooth connection with the device (A) is lost, the connector can use the Service Resource Discovery service to look for a computer that can connect (A) through Bluetooth and (B) through WI-FI, using the computer as a proxy to connect both devices [1].

### 4.2 Adaptation Component

There are several approaches used for adaptation middleware such as: Reflection, Micro-kernel, Aspect-oriented programming and Service-based mechanism. Below we are going to give a brief explanation of the first three, followed by a more in-depth, discussion of the service-based mechanism.

- Reflection: It involves the capability of a system to watch and reason about its own state and finally alternate its own execution.

- Micro-Kernel: It uses the practice of the micro-kernels OS, only the basic functions will be loaded into the kernel, the rest will be loaded at runtime. Several types of middleware use this technology such as: MUSIC and BASE.

- Aspect-oriented programming: It necessitates the involvement of the developer to set three things: join-points, cut-points and advice. A join-point is a point in a program where additional behavior can be added, A cut-point describes the join point. Advice is a piece of code that can be added and run before, after, or around a join-point. Advice is put into the program when a join-point is matched with a cut-point.

- Service-Based mechanism: It is the most popular one that is best suited for ubiquitous computing thanks to its flexibility for integrating new devices. It also covers most of the ubiquitous-computing requirements such as: heterogeneity, scalability,
security, mobility, and discovery [1]. We can take Web Services and REST as examples of service-based implementation. Below, we will explain how web services and REST work.

### 4.2.1 Web Services

Web services are based on many different standard technologies that give them the power to reduce heterogeneity in order to facilitate the integration of applications [16]. Among these standard technologies, there are four that are considered main standards [17]. They are described below:

![Figure 4.3: Web Service Main Standards](image)

- According to [17] and the W3C definition [18], the WSDL is responsible for providing a description of the public interface of the service in XML format, regardless of what message formats or network protocols are used to communicate.

- Also referring to [17] and the W3C definition [19] XML is a text-based format responsible for representing any type of information such as data, document, configuration etc. so the messages can be understood at either end. It is also the most widely-used format for sharing structured information.

- According to [17] and [16] SOAP is responsible for formatting and packaging the information to be exchanged which will be conveyed using typically HTTP. W3C [20] defines it as follows: "It is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols."

- According to [16] UDDI is responsible for organizing information about the service and store it in repository which is a common registry. It provides functionalities to make it easier to publish and find services.

### 4.2.2 REST

After SOAP which is a protocol that has been used since the 90s, REST came to fix all the heavy problems that are inherent in the SOAP protocol and propose an easier way to communicate between client and server. Its full name is Representational state transfer, which is an architectural style for distributed hypermedia systems, where instead of using WSDL to represent a service, a service is represented in terms of resources which have a unique URI through which services are accessed. REST can rely on different transfer protocols such as HTTP, FTP and SMTP; however the most widely used is HTTP. The RESTful services use the HTTP methods CREATE, GET, DELETE and UPDATE to
manage the resources. REST has another advantage over SOAP, which is the data format for messaging. SOAP uses just XML which is considered heavy, hence it increases the load on the server as well as the network traffic whilst REST uses either XML or JSON and most commonly JSON which is a lightweight data format and a human readable language, easier to parse and use by the computer. It is estimated to be at around one hundred times faster than XML [21, 22]. In recent years we can see the significant growth of REST over SOAP [23].

![Figure 4.4: Growth of REST over SOAP](image)

### 4.3 Context Component

To be able to provide the best experience to the user dependent on the context to complete her daily tasks, a good understanding of how context information is processed to match the user’s preferences is required. Figure 4.5 depicts the four-layer architecture of context handling, proposed in [24].

#### 4.3.1 Context Acquisition Layer

This layer is responsible for fetching context data which can be provided dynamically from sensors or by the user’s interaction:

- **Dynamic Sensor Based Context**: Context information is retrieved dynamically from sensors. This information can be anything related to context such as location, temperature, event, etc. This technique for extracting data can be carried out through two methods [4]. It can be done by the sensing method, which is based on extracting data from the sensors, or by the deriving method, which is based on previously computed contextual data collected before, which is cached or saved in the application, and make use of it.

- **Static User Context**: The context information is provided manually by the user by configuring some settings. The application can also get data in off-line mode from agenda, reminder etc.
4.3.2 Context Aggregation Layer

The aggregation layer will select relevant context so it can facilitate the work of the next layer to provide a higher level context information. This layer contains four components that will be described below.

- **Pre-Processor**: It will suggest all possible meaningful contextual information which can be location, temperature, etc., by combining the dynamic and static context data collected by the acquisition layer.

- **Context Negotiator**: It is one of the most important components of the aggregation layer, it will check the correctness and validity of the raw data available.

- **Feature Extractor**: It takes the most relevant attribute of the pre-processed raw data proposed in the earlier phase and characterizes the raw context data into significant features such as event, resource capability in the user’s environment etc.

- **Context Classifier**: It classifies the features provided by the feature extractor. It will combine all the features together and provide medium level context data. Eventually
a new context class will be created or existing context classes will be used to support context reasoning.

4.3.3 Context Reasoner Layer

This is the process that will deduce high level context from a set of low level contexts. This layer is composed of three components and storage facilities.

- **Context Reasoner**: It is responsible for taking data from the Context Classifier of the previous layer and producing a high level context data referencing to the two storage facilities that are the knowledge repository and rule repository. These are respectively responsible for storing smart environment information using an ontology based representation and dictating the behavior of the services in reacting to service invocation.

- **Time Stamp**: It stamps the date and time information before storing it in the data repository which contains important information about the user’s activity, location etc.

- **Data Extractor**: It has as a goal to expose the data repository to the application.

4.3.4 Application Layer

This layer consists of one component; the context application interface which will run the CAA on the user’s device.

- **Context Application Interface**: It has two goals, namely providing the desired user interface and identifying the type of communication standard that will be used.
5 Context Aware Middleware

Middleware is a part of CAA that cannot be ignored since it plays a crucial role in context management and adaptation. The large amount of old and new middleware in the area of context awareness and their large diversity when it comes to functionalities encourage us to make a state-of-the-art in the literature to investigate more into their goals.

5.1 Some Existing Middleware

Below, we present middleware that are the most used and relevant to our research: Aura, SOCAM, CAPNET, CARISMA, CARMEN, GAIA and CORTEX. We will also give a brief description of their characteristics and goals. Finally, we compare them to each other with respect to the requirements listed in section 4; context management, communication, fault tolerance, adaptation, service discovery and context abstraction.

5.1.1 Aura

Aura, from Carnegie Mellon University, provides services for managing application and context tasks. When a user moves from an environment to another the task representation is migrated and the service providers associated are instantiated in the new location. Context observers provide the context information and it is used to derive user intents [3]. Aura consists of four main elements including the context observer for collecting and sending context, the task manager for managing the user tasks, the environment manager for managing context suppliers and related service and finally the context suppliers for providing context information to other components [4]. The figure below taken from [3] is a clear description of the different components of Aura.

![Aura Architecture](image)

Figure 5.6: Aura Architecture

5.1.2 SOCAM

Service-Oriented Context-Aware Middleware (SOCAM), from the National University of Singapore, is a platform to build context aware mobile services [3]. SOCAM is an ontology based middleware composed of Context providers, Context Interpreters, context database, location service, and context-aware mobile services [4, 3]. The context interpreters including context reasoners, take appropriate actions according to the rule and the changes of context, and the context Databases containing all instances of the ontology.
The Location service locates the context providers and the mobile services are applications and services that can adapt their behavior according to the context information. SOCAM is well-known and used for smart home environment [4].

5.1.3 CAPNET

CAPNET, from the University of Oulu, is a lightweight middleware architecture that supports the development of ubiquitous applications and services for mobile devices. It is capable of service discovery, user interface building, management of the local and network resources, asynchronous messaging, context information management and storage. It also has the power to switch from one network connection to another [25].

5.1.4 CARMEN

The primary objective of CARMEN is to handle context aware resource management. It has the ability to make an automatic reconfiguration of wireless Internet services depending on context changes [25]. CARMEN's objectives are [26]: (1) Support for heterogeneous radio access technologies by designing an interface to provide an abstraction of radio-based MAC layers for mesh networks. (2) Support for mobile unicast and broadcast services in a mesh environment. (3) Create a cost-effective mesh network that supports carrier grade services. By using the proxies, CARMEN has the ability to keep resources accessible even after migrating from one environment to another [25].

5.1.5 CARISMA

Context-Aware Reflective middleware System for Mobile Applications (CARISMA), from University College London, uses the reflection paradigm to enhance adaptive and context-aware mobile applications development. Through a reflective API, applications can dynamically check the current configuration of their profile and change it by adding, deleting, or updating the associations previously encoded [27]. CARISMA is a distributed architecture, it uses markup scheme and rule-based method for context modeling and reasoning. The context is stored as an application profile in XML. Since it uses the reflection model for adaptation, the mobile devices can change operating context rapidly [4].

5.1.6 GAIA

GAIA, from the University of Illinois, has the ability to manage resources, and services enclosed physical spaces. It provides similar functionality to an operating system but extended to ubiquitous computing spaces. It coordinates software entities and heterogeneous networked devices contained in a physical space. User data and applications are not enclosed in any active space, which makes it possible to the user to move between different active spaces and their data, and application will always be available. GAIA provides services for event management which provides the possibility to distribute events in active spaces, context information query which makes the adaptation of applications to the environment possible, detection of space repository, and file management which associates context file systems to relevant context information [3, 25, 28]. GAIA Microserver is a lightweight platform that extends GAIA [3]. The need of GAIA Microserver to be able to provide the different services detected by GAIA is a necessity. Since GAIA is responsible for detecting the different services in the environment and GAIA Microserver is used to give access to them through the mobile devices [3].
5.1.7 CORTEX

CORTEX is a context-aware middleware from University of Lancaster that provides support for pervasive and ad hoc environments [3]. It is based on sentient objects (SOs) and component framework. According to [29], "Sentient Object defines software abstractions that ease the use of sensor information, and associated actuation, by CAAs. At the heart of the model is an event-based communication model that permits loose coupling between objects, and consequently supports mobility and application evolution." Using a reflection API, CORTEX can be reconfigured at runtime. The SOs can take several responsibilities, they can take in charge the enabling of the device to discover and access the services or could also provide services to your list of preferences [3]. It uses XML as data format for events and SOAP for messaging [30].

5.2 Evaluation of Context-Aware Middleware

This table will evaluate the capabilities of the six types of middleware that we described above. The evaluation is made according to the requirements that we mentioned in 4. The information that this table contains is based on the two surveys [3, 25].

![Table](image)

Figure 5.7: Evaluation of Context Aware Middleware

As Figure 5.7 demonstrates, the fulfillment of the requirements differs from one type of middleware to another. We can see that Communication and Context Management are fulfilled by all types of middleware which confirms that these two requirements are the most important ones that make middleware context-aware. The adaptation is also a very important requirement which makes middleware respond to the varying environment. The SOCAM middleware, however, does not fulfill this requirement. There is also some middleware that adapts dynamically to the context, which means that there is no need for the user interaction while others do static adaptation which obliges the user to interact with the application in order to make the adaptation process work. We can also see that all middleware fulfills the Service Discovery requirement except for CARISMA middleware. Finally, the Fault Tolerance requirement is not fulfilled by any middleware except for GAIA.
6 Reference Architecture

Based on the literature review about the Adaptive Systems and middleware presented in sections 4 and 5, we could propose a reference architecture that highlights the adaptive system and middleware that are the best suited to our application context. We considered the adaptive system presented as a layered architecture in figure 4.1 as reference. Among the seven types of middleware presented in my study, we chose CORTEX as middleware for our application. This choice was based on many evaluation criteria retrieved from relevant references and presented in section 5. In section 7, we present a deeper discussion and description of our criteria and our choices.

Below we present a reference architecture for "SwipeEat". It is an application which aims to match the nearby restaurants to the list of preferences you provide to the application. In our application the user can swipe between the restaurants that match her preferences. She can like and save the restaurants she is interested in. She can also be notified when she passes by a restaurant that matches her preferences. Thus our application has two scenarios that we name "Matching by Notifications" and "List Matching". In the discussion section, we describe the two scenarios more in depth and we also explain why we chose CORTEX middleware for our application.

The reference architecture shown in figure 6.8 illustrates three main actors that work together to reach the goal of our application. The user can be anyone using our application. The user interacts with the application interface to set her preferences. The sensors may be any system in our environment that can reach and communicate with our system. Our system which consists of a middleware layer and an application layer is responsible for treating the information obtained from the sensors and the user. The middleware commu-
communicates with the surrounding sensors using event-based communication, adapts our system to the varying of the environment, and filters the information received from the sensors. The middleware layer interacts with the application interface, which can then notify the user and provide the matched information in a human readable language.

Figure 1.13 shows a component diagram for our reference architecture. We can see three components namely the Publisher, Middleware and Subscriber components. The publisher component consists of the sensors that surround us and interact with our middleware by providing it with information. The middleware is composed of three sub components: Context Handling, Adaptation and Communication. They will consistently work together to provide the user with the best results. As described in Section 5, the communication component in CORTEX middleware is event-based which means that the communication between the sensors and the middleware is based on notifications. The middleware will be notified of the information that could correspond to the user preferences. Moreover, the Adaptation component is based on the reflection API that CORTEX uses, which permits it to be reconfigured at runtime. It also uses the web service standards such as SOAP and XML. The Context Handling component is responsible for treating the information retrieved from the sensors and checking if it matches the user’s preferences. Finally the Subscriber component, which is basically our application interface, takes the matched results from the middleware and presents it to the user in a human readable language.
7 Discussion

Our system is divided into two layers, an application layer and a middleware layer. The application layer is composed by a user interface component while the middleware layer consists of three components that are communication, adaptation and context handling. Figures 6.8 and 1.13 show the interaction between the user and the application as well as between the middleware and the sensors established in our environment. The user provides the application with a list of food preferences, which then will be converted to a set of data and exchanged with the middleware installed in the device. The role of the middleware in our scenario is to adapt itself to the different kinds of sensors around it and try to interact with them. The context handling component processes the data collected from sensors and sends the data that best matches best the users’ preferences to the application layer. Finally the data is shown in the user interface.

For further illustration, we continue by giving two example scenarios for our application.

Figure 7.9: General Use Case

List Matching

By means of the user interface, the user can enter her preferred food for example: pasta, sushi, pizza, etc. she can also enter the origin of this food such as Italy, Sweden etc. as well as determine the range of discovery; for instance 20km. Based on these settings and preferences the middleware will be able to detect and interact with the sensors around and show all the restaurants matching her preferences and settings. Lastly, she can swipe left to like the restaurant and save it in the list of favorite restaurants or right if she dislikes the restaurant. Figures 1.14 and 1.15 show a use case and a sequence diagram that will give a better understanding of the scenario.
Matching by Notifications

In this scenario the middleware will run in background. Based on the same preferences and settings as the first scenario. For instance if the user is walking around a mall without opening the application, the middleware can still detect some nearby web services around and notify the user if the stores around match her preferences. Figures 1.16 and 1.17 that show a use case and a sequence diagram explain to us how this scenario is achieved.

Choice of Middleware

To begin with, we will mention the criteria that were taken into consideration to choose the middleware of our application.

Our first criterion on choosing the middleware of the application was that the middleware should use event-based communication since it would best suit the objectives of our application. The second criterion is that the middleware should be service-based in term of adaptation. From the list of middleware provided above, there are many that use event-based communication such as: SOCAM, GAIA, CAPNET and CORTEX. However except for GAIA which is service-based, and CORTEX which is mostly based on reflection and using some web service standards such as SOAP and XML, none of the other types of middleware uses service-based mechanism. As mentioned above GAIA is divided into two parts: GAIA which aims to detect the services in the environment and GAIA Microserver that is responsible for forwarding these services to the application. Below we introduce the middleware chosen for our application and justify our choice.

We eventually chose CORTEX in the "SwipeEat" application due to its event-based communication, which only notifies the user when an event occurs - in our situation, if the user is in a vicinity of a restaurant which matches her preferences. As mentioned above in section 4.1, the event-based communication reduces the device's resource consumption since the device just wakes up when an event is triggered. For the adaptation mechanism CORTEX uses reflection including some web service standards that are SOAP for messaging and XML for representing events to the user. In section 4.2, we defined what reflective adaptation is and provided a detailed description of what web services are. We also defined SOAP and XML. Another big advantage of CORTEX is that it provides support for ad hoc and pervasive environments, which is a necessity in our application, since the user is always moving from one environment to another. Moreover CORTEX is constituted of just one component which is responsible for detecting services and forwards them to the application.

As shown in the figure below, the whole idea behind the CORTEX middleware is to get events as input, control the events if they match the user preferences and then forward them as output events to the user.

As shown in the figure below, the whole idea behind the CORTEX middleware is to get events as input, control the events if they match the user preferences and then forward them as output events to the user.

Figure 7.10: CORTEX Flow

Below, we show how these events get forwarded from the publisher to the subscriber. The
role of the publisher is to push events to the subscriber, the event is forwarded by the SOAP Messaging component which is an XML-based protocol. This will then be filtered and able to notify the user with the events matching her preferences. We also provide a sequence diagram to give a more meaningful description of how the CORTEX middleware works.

Figure 7.11: Component Diagram for CORTEX

Figure 7.12: Sequence Diagram for CORTEX
8 Conclusion and Future Work

In this thesis research, we explain in details what context is, as well as the different types of context and context-aware computing. We also provide a thorough explanation of what middleware is and its different components. We present the necessary information concerning the communication techniques that are used in adaptive systems. We also deal with several adaptation mechanisms, focusing mostly on the service-based mechanism. We explain the different layers in processing context information, to provide the user with the best information tailored to her preferences. We present a state-of-the-art in the literature of some existing middleware. To conclude we provide a reference architecture for an application and justify the choice of the middleware. Future work that can be considered based on this project would be a more detailed explanation of the adaptation mechanisms that are: Reflection, Micro-kernel, Aspect-oriented programming. Furthermore, covering more middleware in the state-of-the-art could be considered.

In our paper, we present an architecture that initially aims at filtering the users’ preferences in the area of catering. This context was the starting point from which we were able to develop a model that could be adapted to all types of preferences as long as it respects the characteristics of the CORTEX middleware chosen. Related fields that could be considered are:

- Handling the preferences in art which could consist in a CCA aiming to guide people inside a museum or for the choice of a museum.

- Handling the preferences in literature which could be having a CCA guiding the users inside libraries or book stores.

- Handling the preferences in music through a CCA guiding you in a festival where different types of music is played on different stages.

Furthermore, we could also propose a generic application that would enable the user to choose the preference to be treated. This kind of application would take the choice of the user and apply it to our architecture which would then be configurable.
A Appendices

A.1 Interview Questions

1. Do you work for a company or a public organization?
2. What is your title position?
3. Why and when have you investigated about context aware computing and ubiquitous environment?
4. How can a device be context aware?
5. Is the use of web services mandatory to adapt to the change of the environment?
6. Is the use of context-aware middleware mandatory to make an application context aware?
   (a) If yes, are there different types of context-aware middleware available?
   (b) If yes, do they differ from each other in terms of adaptation and communication?
7. Can a bad choice of middleware affect the performance of my application?
8. Any suggestions for good papers that can be useful for my research?
A.2 Component Diagram for "SwipeEat"

Figure 1.13: Component Diagram for "SwipeEat"
A.3 Diagrams of the Scenarios

Figure 1.14: "List Matching" Use Case
Figure 1.15: Sequence Diagram for "List Matching" Scenario
Figure 1.16: "Matching by Notification" Use Case
Figure 1.17: Sequence Diagram for "Matching by Notification" Scenario
References


less sensor networks based on tuple space,” in 22nd International Conference on
Advanced Information Networking and Applications - Workshops (aina workshops


towards-natively-context-aware-web-services


services for enterprise application integration,” in 2015 International Conference on


[23] Reza Shafii. (2014, Jan. 15) Minding the api hierarchy of needs with raml and


tectures,” in 2010 IEEE International Conference on Electro/Information Technol-
ogy, May 2010, pp. 1–6.


