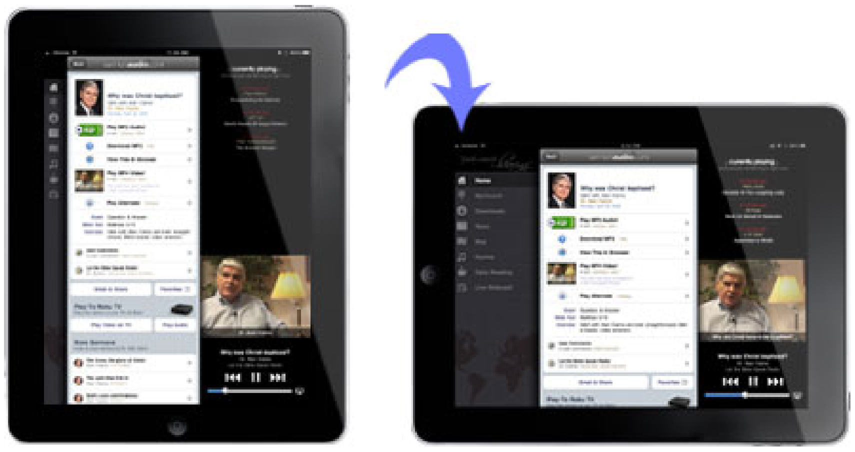
The Encyclopedia of Human-Computer Interaction, 2nd Ed.

14. Context-Aware Computing

BY ALBRECHT SCHMIDT

A tablet computer switching the orientation of the screen, maps orienting themselves with the user's current orientation and adapting the zoom level to the current speed, and switching on the backlight of the phone when used in the dark are examples of computers that are aware of their environment and their context of use. Less than 10 years ago, such functions were not common and existed only on prototype devices in research labs working on context-aware computing.



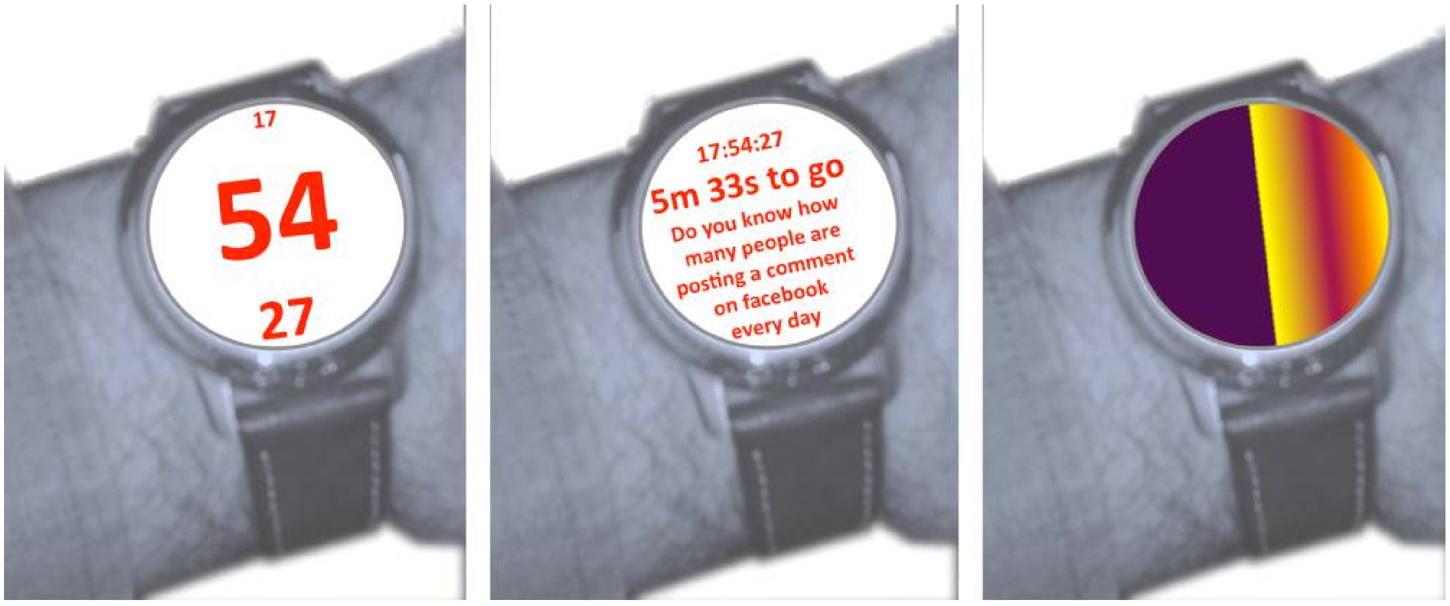
Author/Copyright holder: SermonAudio.com. Copyright terms and licence: All Rights Reserved. Reproduced with permission. See section "Exceptions" in the *copyright terms* *below.*

Figure 14.1: An iPad switching orientation of the screen is a good example of context-aware computing

14.1 Introduction

When we aim to create applications, devices, and systems that are *easy to use*, it is essential to understand the *context of use*. With context-aware computing, we now have the means of considering the situation of use not only in the design process, but in real time while the device is in use. In Human-Computer Interaction (HCI), we traditionally aim to understand the user and the context of use and create designs that support the major anticipated use cases and situations of use. In Context-Aware Computing on the other hand, making use of context causes a fundamental change: We can support more than one context of use that are equally optimal. At runtime – when the user interacts with the application — the system can decide what the current context of use is and provide a user interface specifically optimized for this context. With context-awareness, the job of designing the user interface typically becomes more complex as the number of situations and contexts which the system will be used in usually increases. In contrast to traditional systems, we do not design for a single -or a limited set - of contexts of use; Instead, design for several contexts. The advantage of this approach is that we can provide optimized user interfaces for a range of contexts.

Let us assume the following example: You are asked to design a user interface for a wrist watch. In your research you find out that people will use it indoors and outdoors, they will use it in the dark as well as in sunlight, they will use it when they run to catch the train as well as when they sit in a lecture and are bored. As a good user interface designer, you end up with many ideas for an exciting user interface for each situation: For example, when the user is running to catch the train, the user interface should show the time highlighting the minutes and seconds in a very large font. On the other hand, when the user is attending a lecture the user interface should show the time in a very small font, and additionally provide a funny quote. In a traditional design process, you would realize – after creating your sketches and design briefs – that you have to decide which one of your ideas for a user interface you want to use. You would realize that supporting all the situations in a single design will not work. The typical result is a compromise – which often loses much of the edge of the ideas you initially came up with. However, if you take the approach of Context-Aware Computing, you could create a context-aware watch, where you combine all your situation-optimized designs in a single design. If you designed your watch so that it could recognize each of the situations that you had found in your initial research (e.g. running to catch the train, attending a lecture, etc), your watch could reconfigure itself based on the recognized context. Figure 2 shows a design sketch for a context-aware watch.


17
54
27
17:54:27
5m 33s to go
Do you know how many people are posting a comment on facebook every day

Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.2: Design sketches that illustrate ideas for time visualizations in different contexts. Left: for users that run to catch the train; making it easy to see the minutes. Middle: a time visualization for boring lectures and meetings; shows a count down to the end, and some information to engage the user. Right: visualization that gives only a very coarse idea of the time, similar to the information you get from the sun, e.g. for hanging out with friends when time does not matter. With context-awareness, you could create a product that combines all three visualizations and choose the most appropriate one according to the recognized context.

Accelerate your career: Get industry-trusted Course Certificates

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

“Mobile User Experience (UX) Design”

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder, Interaction Design Foundation

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Head of Instruction, Interaction Design Foundation

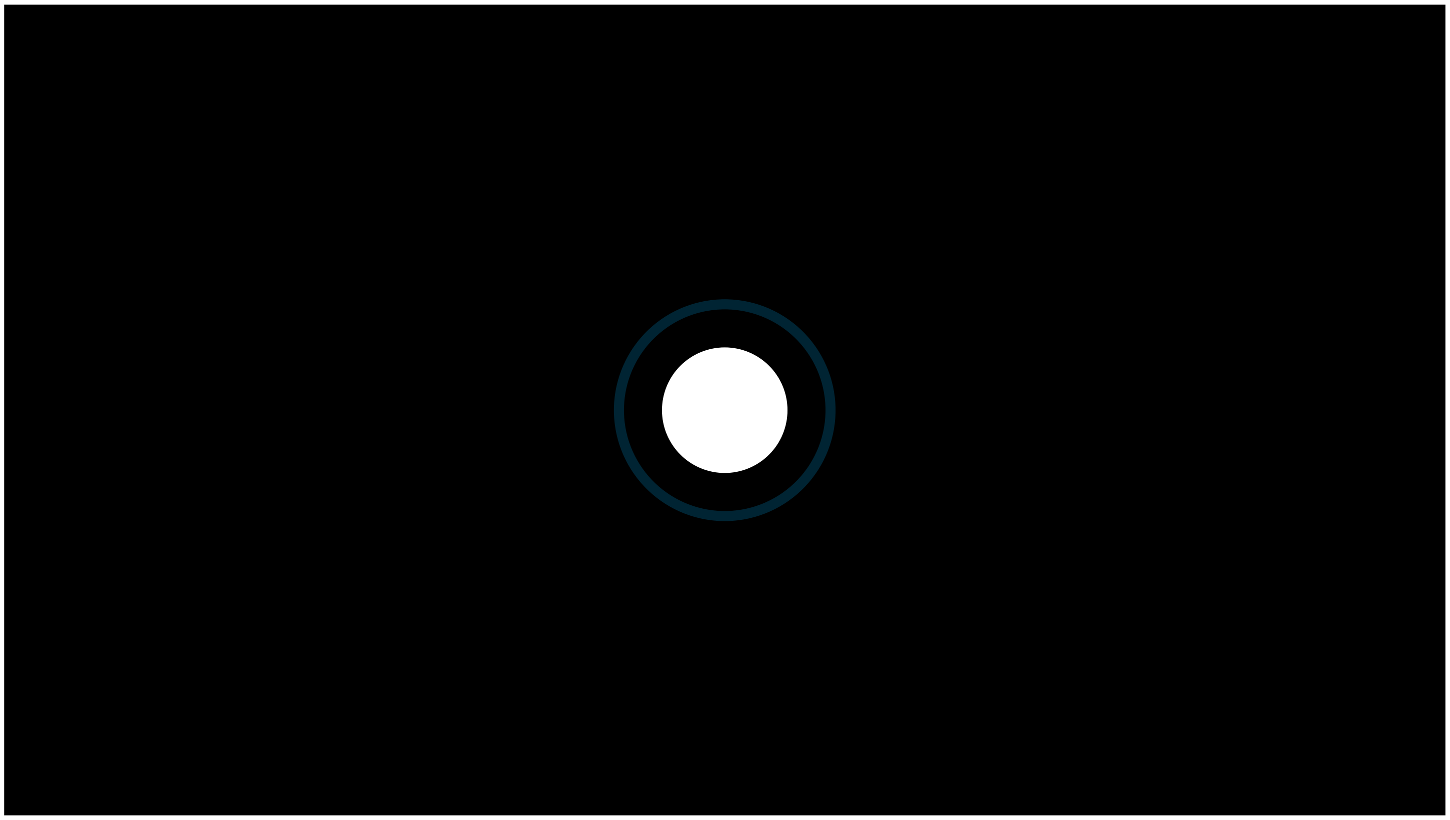
BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner’s Guide

INTERMEDIATE UX COURSES

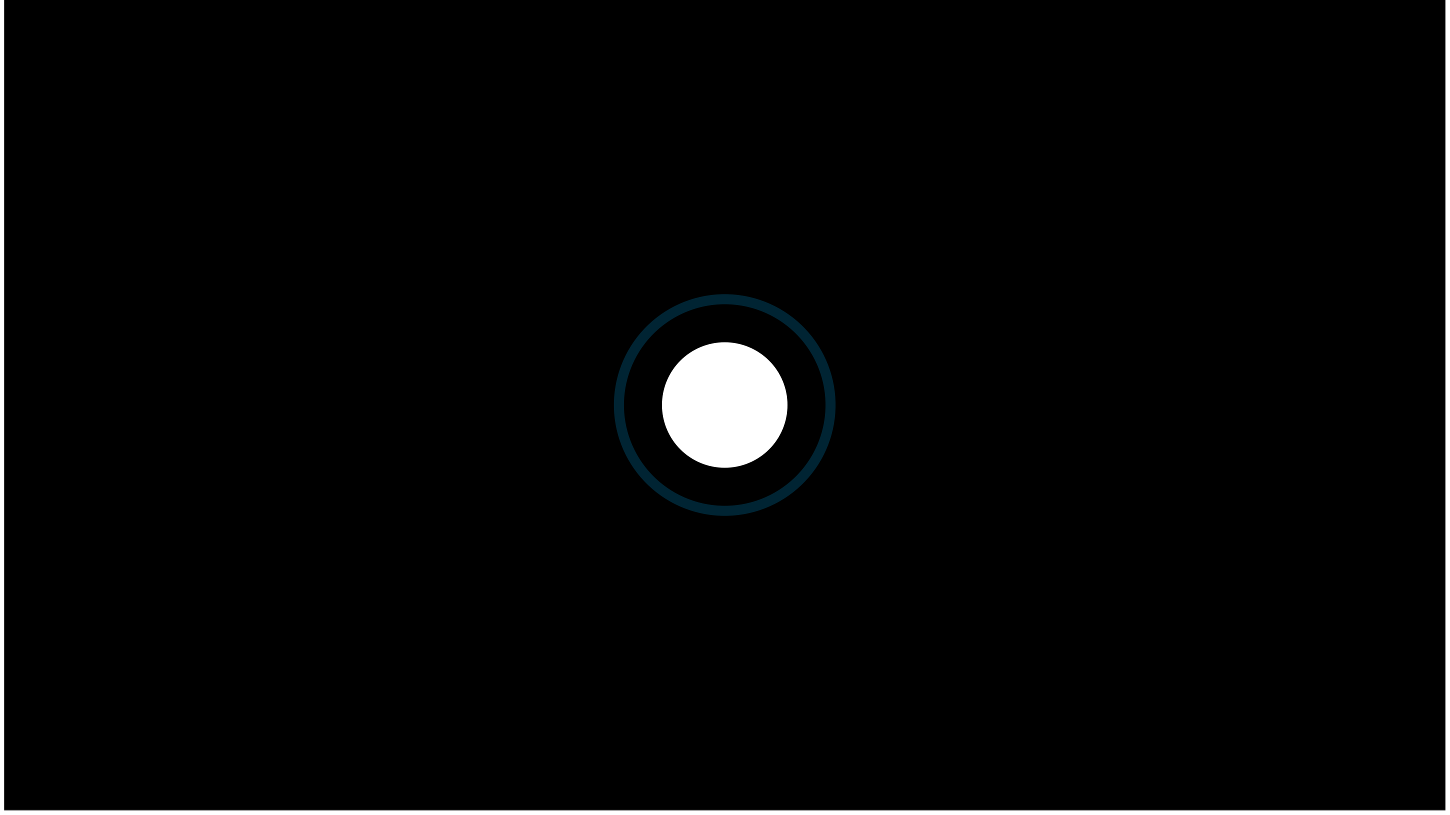
- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

The example shows the great advantage of context-aware computing systems as the freedom of design is increased, but at the same time systems become more complex and often more difficult to design and implement. In this chapter, we introduce the basics for creating context-aware applications and discuss how these insights may help design systems that are easier and more pleasant to use.



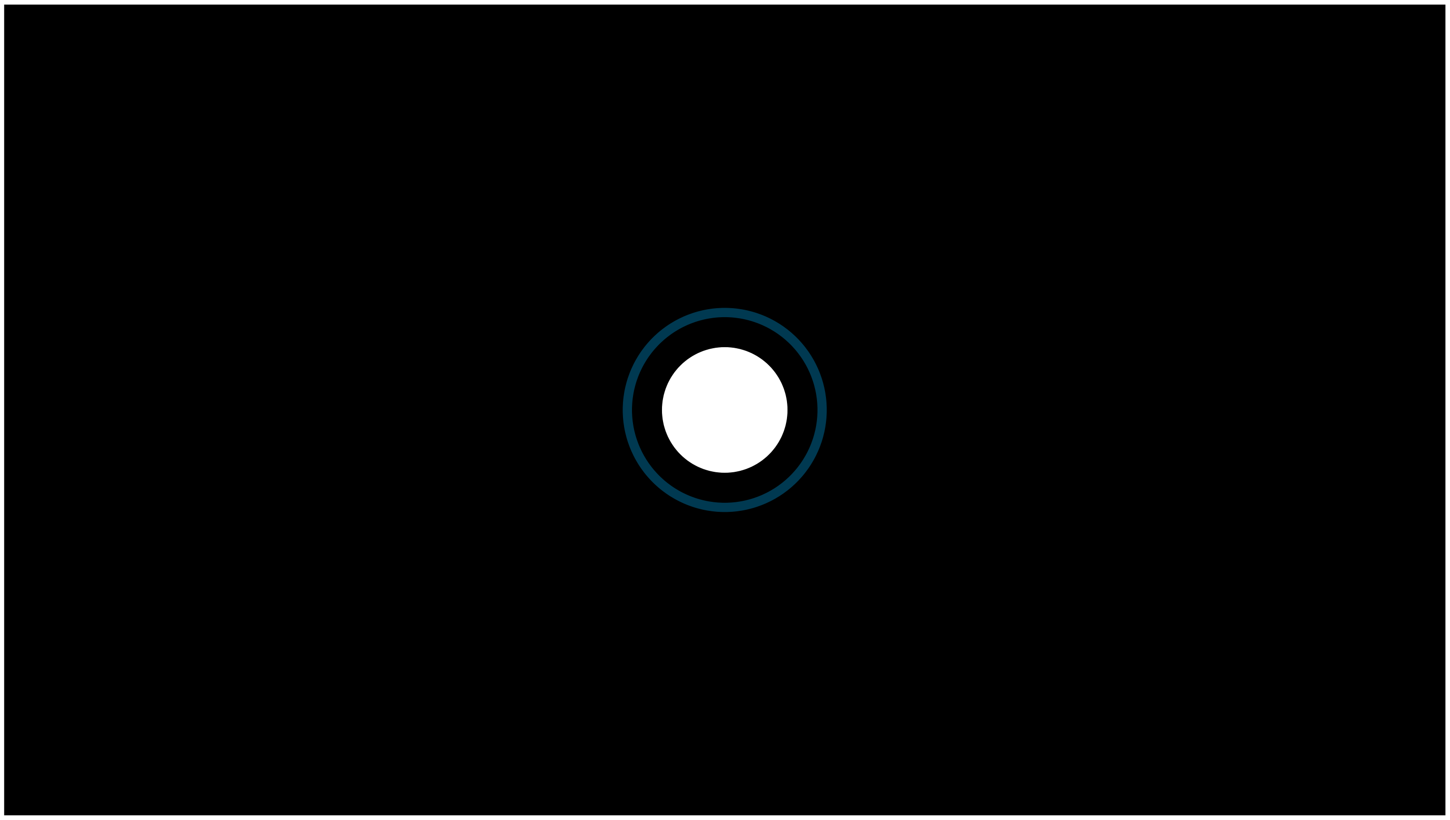
Author/Copyright holder: Courtesy of Albrecht Schmidt. Copyright terms and licence: CC-Att-ND (Creative Commons Attribution-NoDerivs 3.0 Unported).

What is Context?



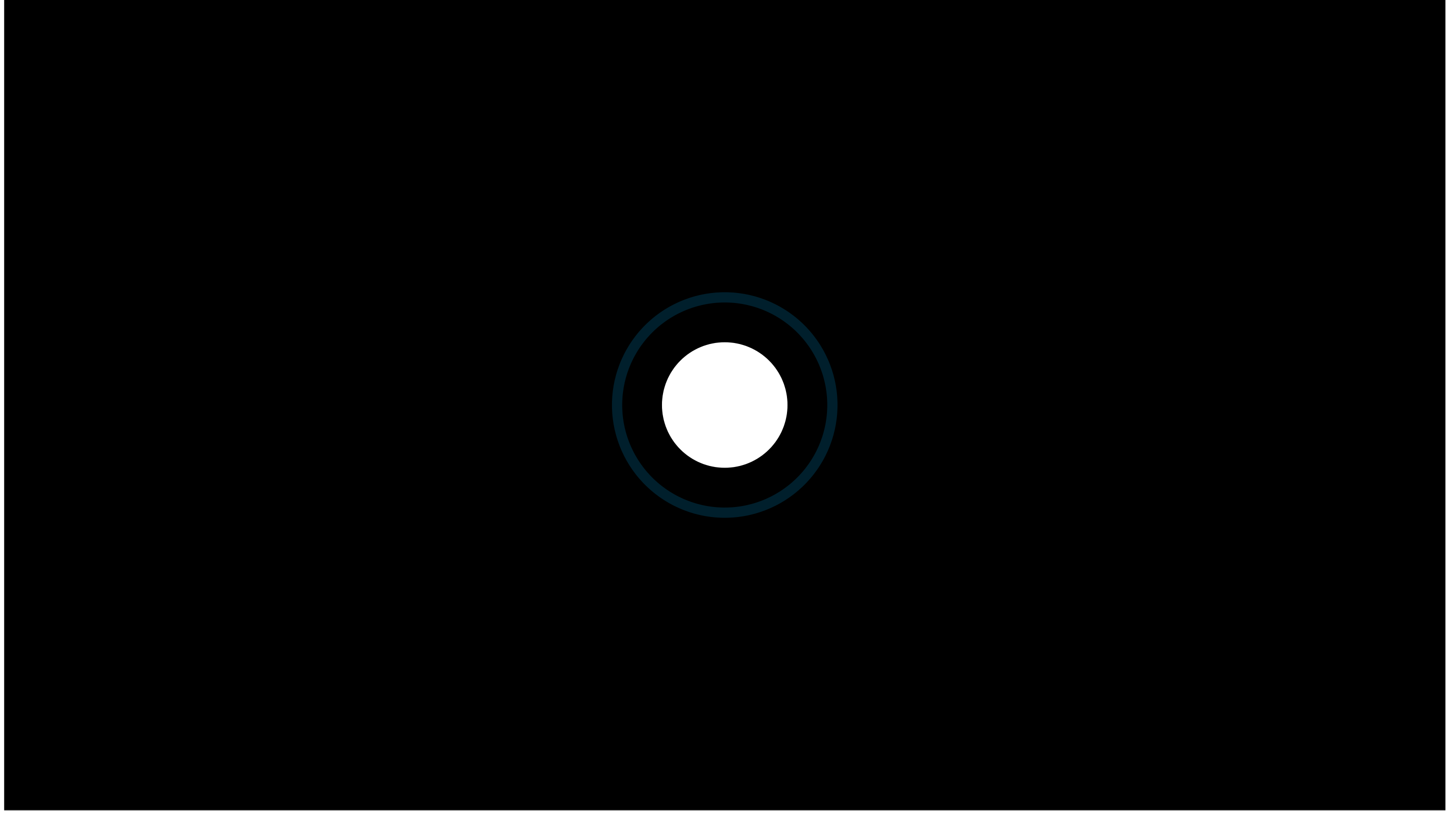
Author/Copyright holder: Courtesy of Albrecht Schmidt. Copyright terms and licence: CC-Att-ND (Creative Commons Attribution-NoDerivs 3.0 Unported).

Advice, Criticisms and Business Value



Author/Copyright holder: Courtesy of Albrecht Schmidt. Copyright terms and licence: CC-Att-ND (Creative Commons Attribution-NoDerivs 3.0 Unported).

Guidelines for Designing Products using Context Aware Computing



Author/Copyright holder: Courtesy of Albrecht Schmidt. Copyright terms and licence: CC-Att-ND (Creative Commons Attribution-NoDerivs 3.0 Unported).

Current Challenges and the Future of Context Aware Computing

14.2 Context-Awareness

In the early days of computing, the context in which systems were used was strongly defined by the place in which computers were set up, see Figure 3. Personal computers were used in office environments or on factory floors. The context of use did not change much, and there was little variance in the situations surrounding the computer. Hence, there was no need to adapt to different environments. Many traditional methods in the discipline of Human-Computer Interaction (HCI), such as contextual inquiry or task analysis, have their origin in this period and are most easy to use in situations that do not constantly change. With the rise of mobile computers and ubiquitous computing, this changed. Users take computers with them and use them in many different situations, see Figure 4.

At the beginning of the mobile computing era, in the late 80s and 90s, the central theme was how to make mobility transparent for the user, automatically providing the same service everywhere. Here, transparent meant that the user did not have to care about changes in the environment and could rely on the same functionality independent of the environment.

In the early 90s, research into ubiquitous computing at Xerox PARC caused a shift in thinking. In addition to making functionality transparent, such as providing network connectivity throughout a campus without the user realizing the hand-over between different networks, researchers discovered the potential to exploit the context of use as a resource to which systems can be adapted. In his 1994 paper at the Workshop on Mobile Computing Systems and Applications (WMCSA), Bill Schilit introduces the concept of context-aware computing and describes it as follows:

> Such context-aware software adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment.
>
> -- Schilit et al 1994

The basic ideas is that mobile devices can provide different services in different contexts – where context is strongly related to the location of a device.

Much of the initial research of context-aware computing hence focused on *location-aware* systems. In this sense, the widely-used satellite navigation systems in cars today are context-aware systems. However, context is more than location, as we argue in (Schmidt et al 1999) and throughout this chapter.

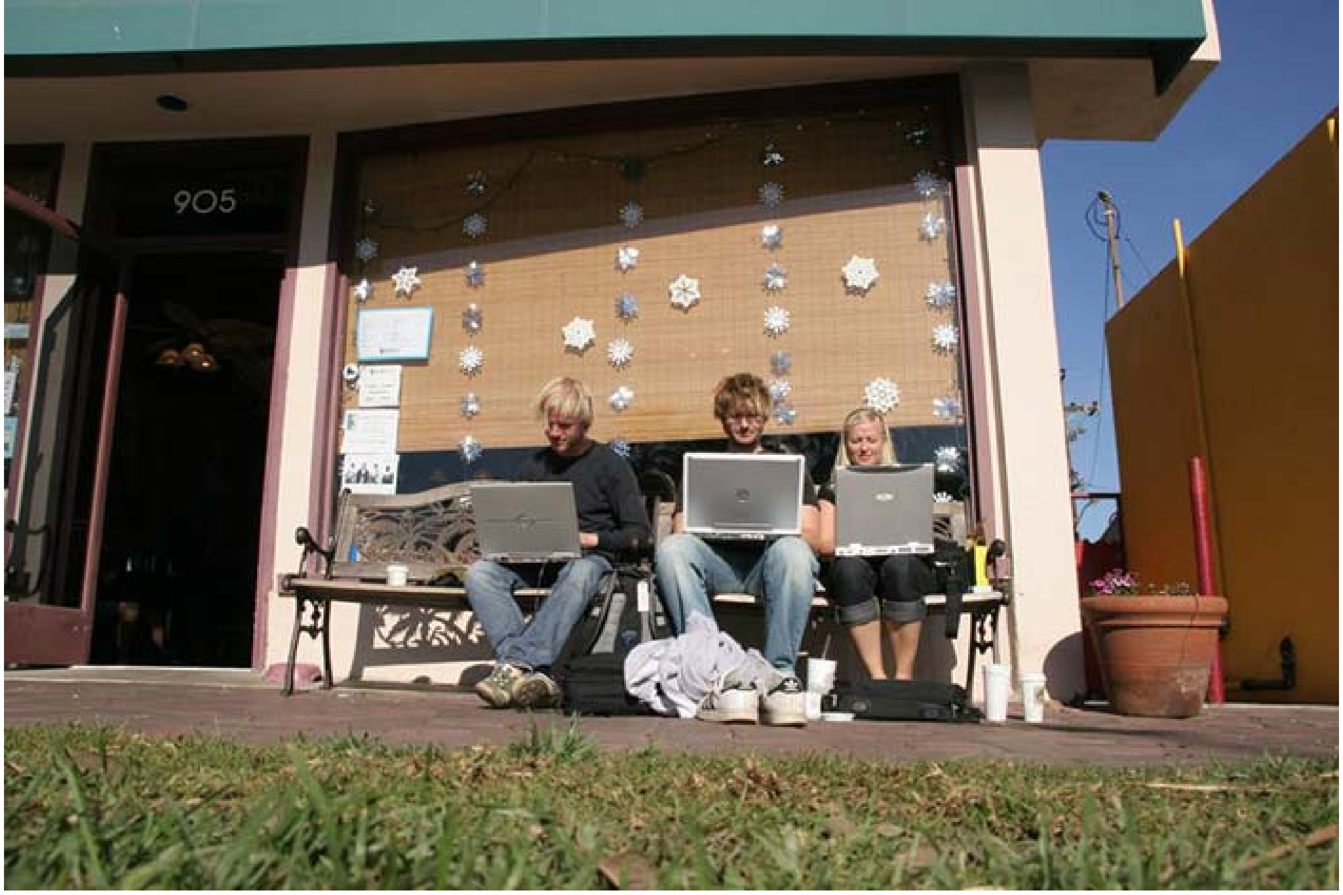
Author/Copyright holder: Courtesy of Library of Virginia. Copyright terms and licence: pd (Public Domain (information that is common property and contains no original authorship)).

Author/Copyright holder: Courtesy of Library of Virginia. Copyright terms and licence: pd (Public Domain (information that is common property and contains no original authorship)).

Author/Copyright holder: Courtesy of The Library of the London School of Economics and Political Science. Copyright terms and licence: pd (Public Domain (information that is common property and contains no original authorship)).

Figure 14.3 A-B-C: In the early days of computing, the context was defined by the computer as the computer was the actual workplace. Later, computers were set up in a specific location to help with a specific task, and hence the context was strongly defined by the location of the computer. Personal computers were used in office environments or on factory floors.

Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms *below.*



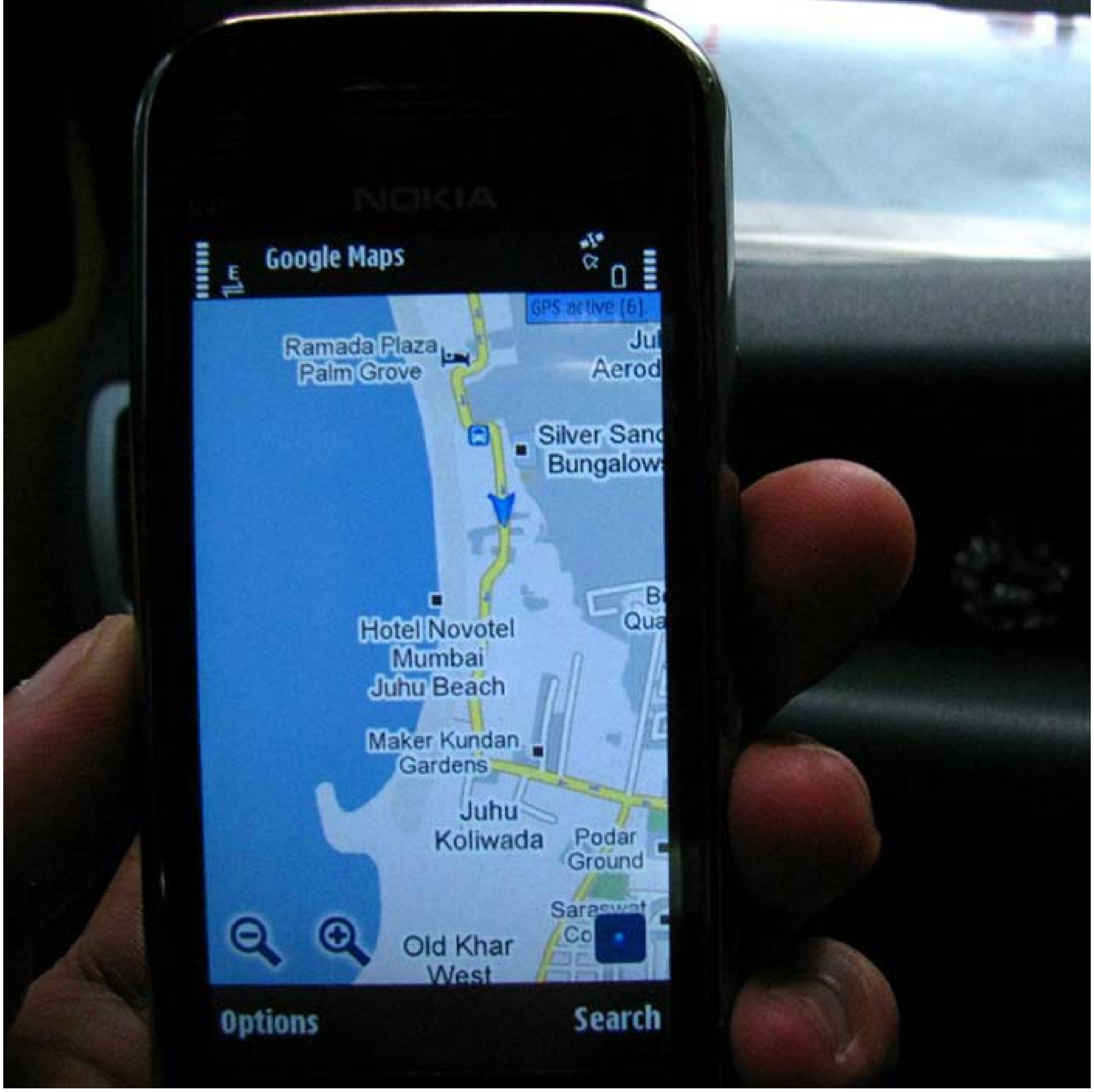
Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.4 A-B: Even in the early days of mobile computing, where notebook computers were considered mobile devices, users could choose the context in which to work. With the rise of mobile and handheld computers and ubiquitous computing, this changed even more radically. Users take computers with them and use them in many different situations in the real world. Next time you go for a walk, observe the multitude of mobile usage scenarios and you will be surprised what people do with their devices

14.2.1 Example 1: SatNav as context-aware system

In a Satellite Navigation System (SatNav), the current location is the primary contextual parameter that is used to automatically adjust the visualization (e.g. map, arrows, directions…) to the user's current location. An example is shown in Figure 5. However, looking at current commercial systems, much more context information is used and much of visualization has been changed. In addition to the current GPS position, contextual parameters may include the time of day, light conditions, the traffic situation on the calculated route or the user's preferred places. Beyond the visualization and whether or not to switch on the backlight, the calculated route can be influenced by context, e.g. to avoid potentially busy streets at that time of day.

Author/Copyright holder: Courtesy of Eirik Solheim Copyright terms and licence: CC-Att-SA-2 (Creative Commons Attribution-ShareAlike 2.0 Unported).



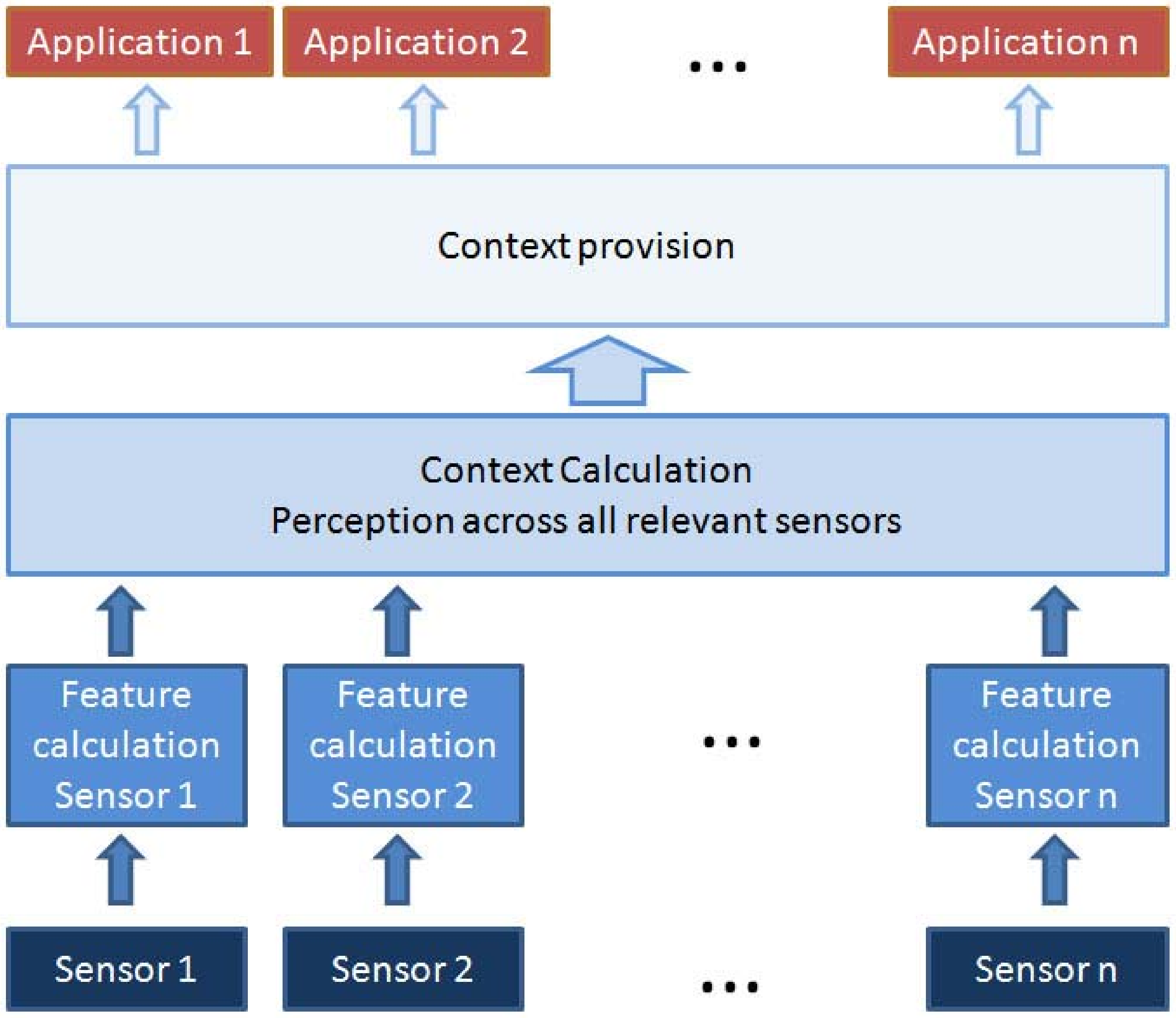
Author/Copyright holder: Courtesy of Satish Krishnamurthy. Copyright terms and licence: CC-Att-2 (Creative Commons Attribution 2.0 Unported).

Figure 14.5 A-B: Navigation devices have become common and are widely used in cars and by pedestrians. Figure 4a shows a TomTom navigation application on a Nokia N95 device. Figure 4B shows Google Maps on another Nokia device. SatNavs are probably the most widespread context-aware computing systems

14.2.2 Example 2: Automatic light as context-aware system

At house entrances and in hotel hallways automatic lights have become common. These systems can also be seen as simple context-aware systems. The contextual parameters taken into account are the current light conditions and if there is motion in the vicinity. The adaptation mechanism is fairly simple. If the situation detected is that it is dark and that there is someone moving, the light will be switched on. The light will then be on as long as the person moves, and after a period where no motion is detected, the light will switch off again. Similarly, the light will switch off if it is not dark anymore.

These simple examples outline the basic principle of a context-aware system. In Figure 6 a reference architecture for context-aware computing systems is shown. Sensors provide data about activities and events in the real world. Perception algorithms will make sense of these stimuli and classify the situations into context. Based on the observed context, actions of the system will be triggered.


Application 1
Application 2
...
Application n
Context provision
Context Calculation
Perception across all relevant sensors
Feature calculation Sensor 1
Feature calculation Sensor 2
...
Feature calculation Sensor n
Sensor 1
Sensor 2
...
Sensor n

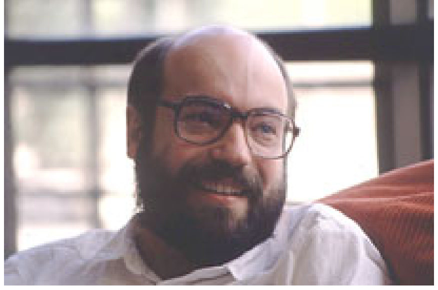
Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.6: The drawing depicts a reference architecture for context-aware computing systems. It assumes the acquisition of data from sensors support to contextual behavior of multiple applications

14.3 Context-Awareness as Enabler for Ubiquitous Computing

The notion of context-awareness is closely related to the vision of ubiquitous computing, as introduced by Mark Weiser (see Figure 7) in his seminal paper in the journal Scientific American. As computers become a part of everyday life, it is essential that they are easy to use. This is highlighted by the following statement.

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.
>
> -- Weiser 1991



Author/Copyright holder: Mark Weiser. Copyright terms and licence: All Rights Reserved. Used without permission under the Fair Use Doctrine (as permission could not be obtained). See the "Exceptions" section (and subsection "allRightsReserved-UsedWithoutPermission") on the page copyright notice.

Figure 14.7: Mark Weiser stated the vision of ubiquitous computing. Context-awareness is an essential building block for realizing this vision

Many people regard this level of integration of computing technologies as the ultimate goal for computers. In such a situation, technologies would not require much active attention by the user, and would be ready to use at a glance. If this is achieved, computers *disappear* – not in a technical sense, but from a psychological perspective.

> In essence, that only when things disappear in this way are we freed to use them without thinking.
>
> -- Weiser 1991

To realize such ubiquitous computing systems with optimal usability, i.e. *transparency of use*, context-aware behaviour is seen as the key enabling factor. Computers already pervade our everyday life - in our phones, fridges, TVs, toasters, alarm clocks, watches, etc - but to fully *disappear*, as in the Weiser's vision of ubiquitous computing, they have to anticipate the user's needs in a particular situation and act proactively to provide appropriate assistance. This capability require means to be aware of its surroundings, i.e. context-awareness.

There are many examples of computing systems that are so well implemented that users are not aware that they have interacted with them. Cars are a prime example: ABS (anti-lock braking system) and ESP (Electronic Stability Program) are integrated in cars and influence their usage in extreme situations. Nevertheless, most people will not consciously be thinking of these technologies when operating a car. These technologies are ubiquitous and have disappeared from the user's conscious mind.

14.4 The notion of context

The term context is widely used with very different meanings. The following definitions from the dictionary, as well as the synonyms, provide a basic understanding of the meaning of context.

context, noun. *Cause of event*
the situation within which something exists or happens, and that can help explain it
Cambridge Dictionary

Synonyms for context: circumstance, situation, phase, position, posture, attitude, place, point, terms, regime, footing, standing, status, occasion, surroundings, environment, location, dependence.

Context-aware computing literature also has several definitions and explanations of what context is. The following are prominent examples that highlight the basic understanding shared in the community.

At the University of Kent, research was conducted that looked at how mobile devices with GPS (externally connected), network access, and further sensors can be used to support the fieldwork of mobile workers. The research team suggested the following definition:

> [...] 'context awareness', a term that describes the ability of the computer to sense and act upon information about its environment, such as location, time, temperature or user identity. This information can be used not only to tag information as it is collected in the field, but also to enable selective responses such as triggering alarms or retrieving information relevant to the task at hand.
>
> -- Ryan et al 1998

Author/Copyright holder: Courtesy of Tom Oates. Copyright terms and licence: CC-Att-SA-3 (Creative Commons Attribution-ShareAlike 3.0).

Figure 14.8: Lancaster castle is one of the locations that was featured in the GUIDE tourist system.

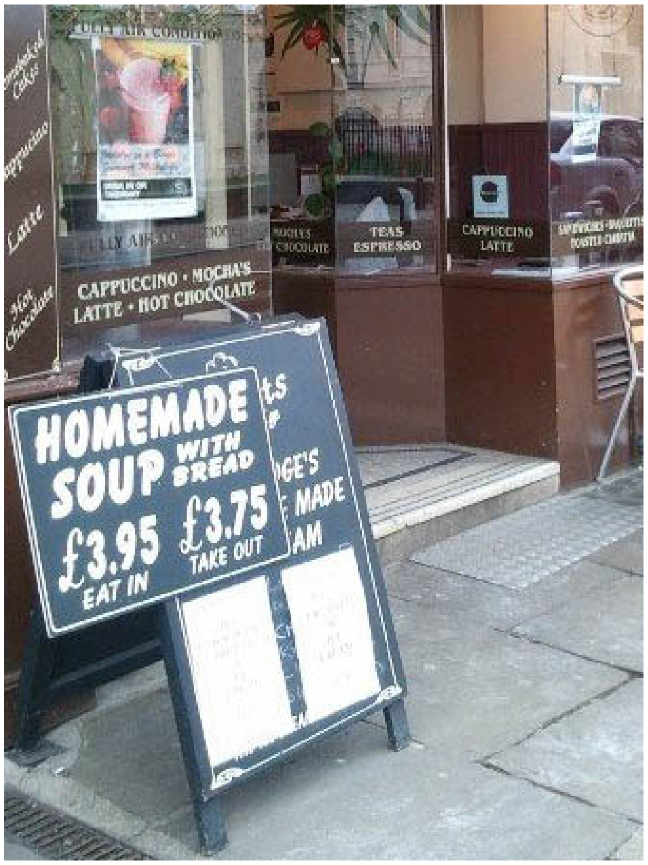
The GUIDE project (GUIDE 2001) at Lancaster University was the first larger and public installation of a research prototype to explore context-awareness in the domain of tourism. It focused on how context can be used to advance a mobile information system for visitors to the historic town of Lancaster. Keith Mitchell suggests the following notion of context in his thesis, based on work with the GUIDE system:

> [...] two classes of context were identified, namely personal and environmental context. [...]. Examples of environmental context include: the time of day, the opening times of attractions and the current weather forecast.
>
> -- Mitchell 2002

This is an understanding of context where the users themselves are part of the context (e.g. profiles, preferences). Technically, GUIDE followed an interesting approach as it used a modified browser in which context information was used in the background to adapt content and presentation.

Anind Dey has suggested a very generic description of what constitutes context:

> Context 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 application themselves.
>
> -- Dey 2000



Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms *below.*

Figure 14.9: This traditional board advertises the same product to all people who pass by - home made soup with bread. In the future such boards will be replaced with digital screens and then it becomes possible to make the content adapt to the current context. If you are interested in the future visions of public display networks have a look at pd-net.org

Accelerate your career: Get industry-trusted Course Certificates

INTERACTION DESIGN FOUNDATION
Est. 2002

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

"Mobile User Experience (UX) Design"

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder,
Interaction Design Foundation

CERTIFICATE
You can verify this certificate at:
https://www.interaction-design.org/victor_johnson
VERIFICATION

Head of Instruction,
Interaction Design Foundation

BEGINNER UX COURSES

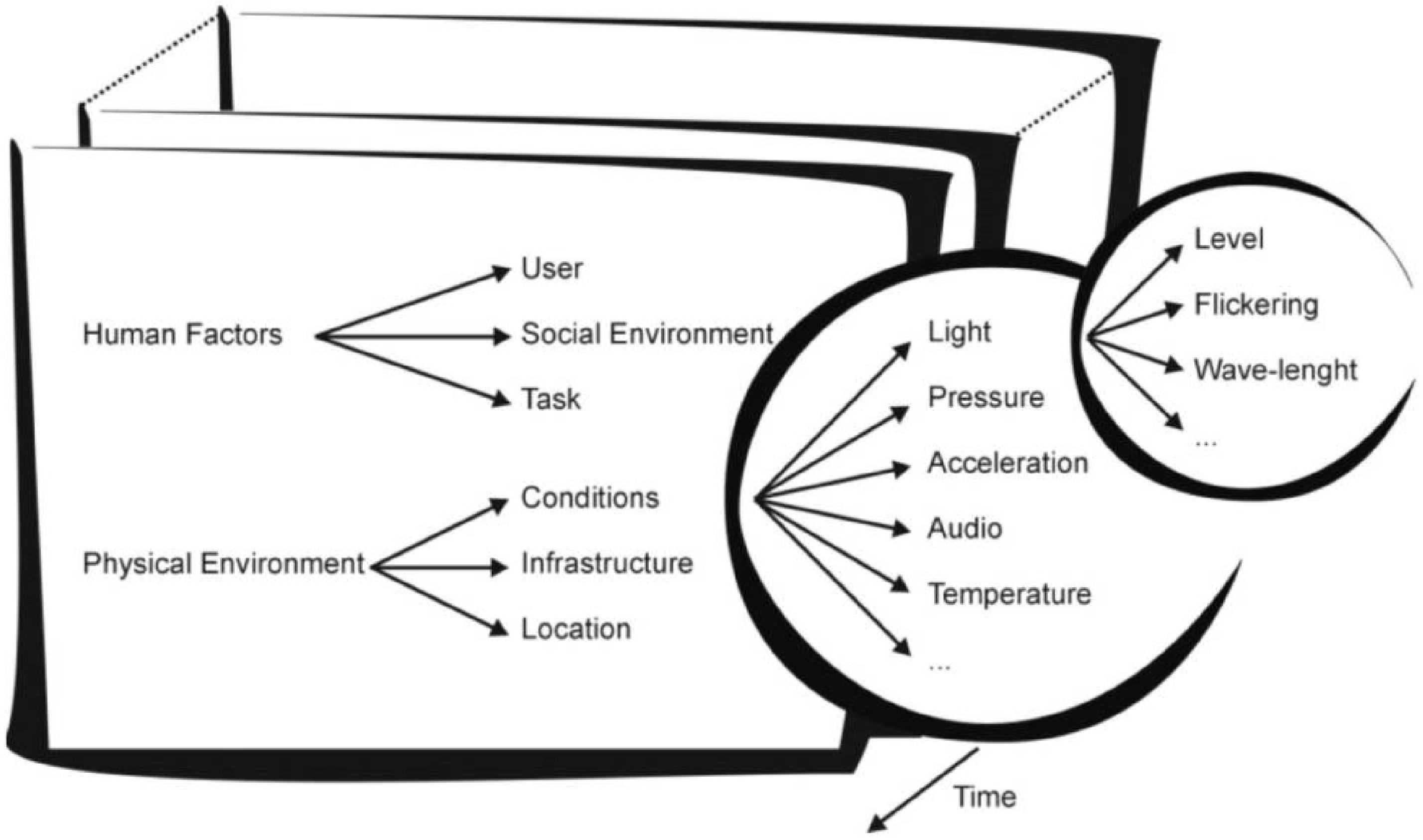
- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner's Guide

INTERMEDIATE UX COURSES

- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

For practical purposes, context is often hierarchically structured describing the relevant features. The feature space described by myself (Schmidt et al 1999) is an example of such a structured representation of context, see Figure 10. Let us assume you want to design a digital replacement of a menu you find often at the entrance of a restaurant (see Figure 9 for an example). If you have a non context-aware version, this would typically show the special of the day. Instead, if you designed it as context-aware, you would want to have different suggestions on the menu depending on who is walking past it. If parents with children walk by, you would show the family-oriented menu; if a couple is looking at it in the evening, you would show the menu for a candle light dinner; and if it is hot and sunny in the afternoon, you would advertise the selection of ice cream you have. A feature space for this application could include the people looking at it, the time of day, and the weather. People could be refined to number of people, age, and gender. The weather could include temperature and whether it rains or not. By providing such a structured space, it becomes easier to link contexts in the real world to adaptations in the system. Try as an example to do a full feature space for the menu and define appropriate adaptations. Even a checklist could be considered as a very simple example of a non-hierarchical feature space.

There is no feature space that is complete and describes all possible options – such a feature space would in fact be an attempt to provide a complete description of the world. The usual approach is to create a feature space for the specific context-aware application that is designed. The advantage of a feature space is that by looking at a set of parameters, it can be easily determined if a situation matches a context or not.



Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms *below.*

Figure 14.10: Context feature space, detailing light as one feature in the conditions of the physical environment

Design Hint 1: When building a context-aware system, first create a (hierarchical) feature space with factors that will influence the system behaviour

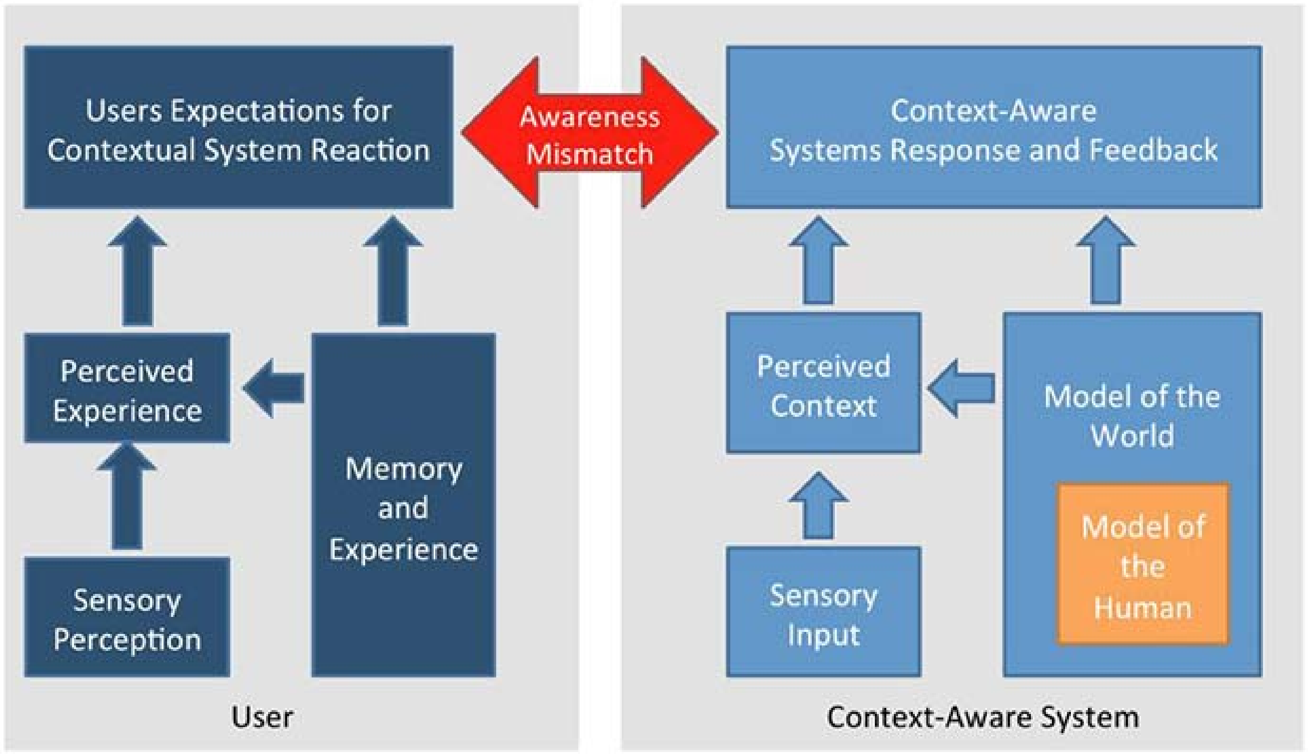
Knowing which are the factors that should influence the system behaviour, one can start to look at how these factors can be determined in the devices. In many cases this will require sensors that allow the provision of context.

14.5 From sensors to context

The ultimate goal of a context-aware system is for the system to arrive at a representation of the surrounding world that is close to the perception of the user. An important question is how to narrow the gap between the user's and the system's perception (or understanding) of the real world. For location, different means of sensing (e.g. GPS) and interpretation (World Geodetic System, WSG84, post code) are well-established. However, for many other sensors there is typically no single and well-understood way for interpreting the sensed information.

The user's perception of the surroundings is based on human senses, but relates at the same time to experience and memory. Human perception is multifaceted. When walking home from the bus stop late in the evening, a user may perceive that it is dark, quiet, and cold, but at the same time he may perceive the situation as scary. Another user, who was busy the whole day and surrounded by people, may perceive the situation also as dark, quiet, and cold, but at the same time as relaxing and free. This example shows that relying on sensor data alone does not provide the complete picture. It is important to remember that even a perfect design and implementation will not be able to perceive the environment in exactly the same way as the user does.

We now have the following ingredients: The user's perception and the user's experience which both lead to the user's expectations; the system's perceived context drawing from the sensor input; the system's model of the world including a model of the user driving the system's reaction. See Figure 5. The main goal of a good and usable design should be to minimize the "awareness mismatch", as illustrated in Figure 11.



Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms *below.*

Figure 14.11: The shown User-Context Perception Model (UCPM) highlights the parallel perception processes in the user and in the system. If they are different we create systems with an awareness mismatch, where the system behaviour does not correspond with the users' expectations.

The User-Context Perception Model (UCPM) is a model created to help the designer understand the challenge he faces in creating context-aware systems. It does not describe the way humans work, nor does it prescribe how to implement the system. Nevertheless, the model of a context-aware system, as shown in the model, can provide a good starting point for designing the system architecture of context-aware applications.

Accelerate your career: Get industry-trusted Course Certificates

Est. 2002

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

"Mobile User Experience (UX) Design"

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

[illegible]
Founder,
Interaction Design Foundation

You can verify this certificate at:
https://www.interaction-design.org/victor_johnson

[illegible]
Head of Instruction,
Interaction Design Foundation

BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner's Guide

INTERMEDIATE UX COURSES

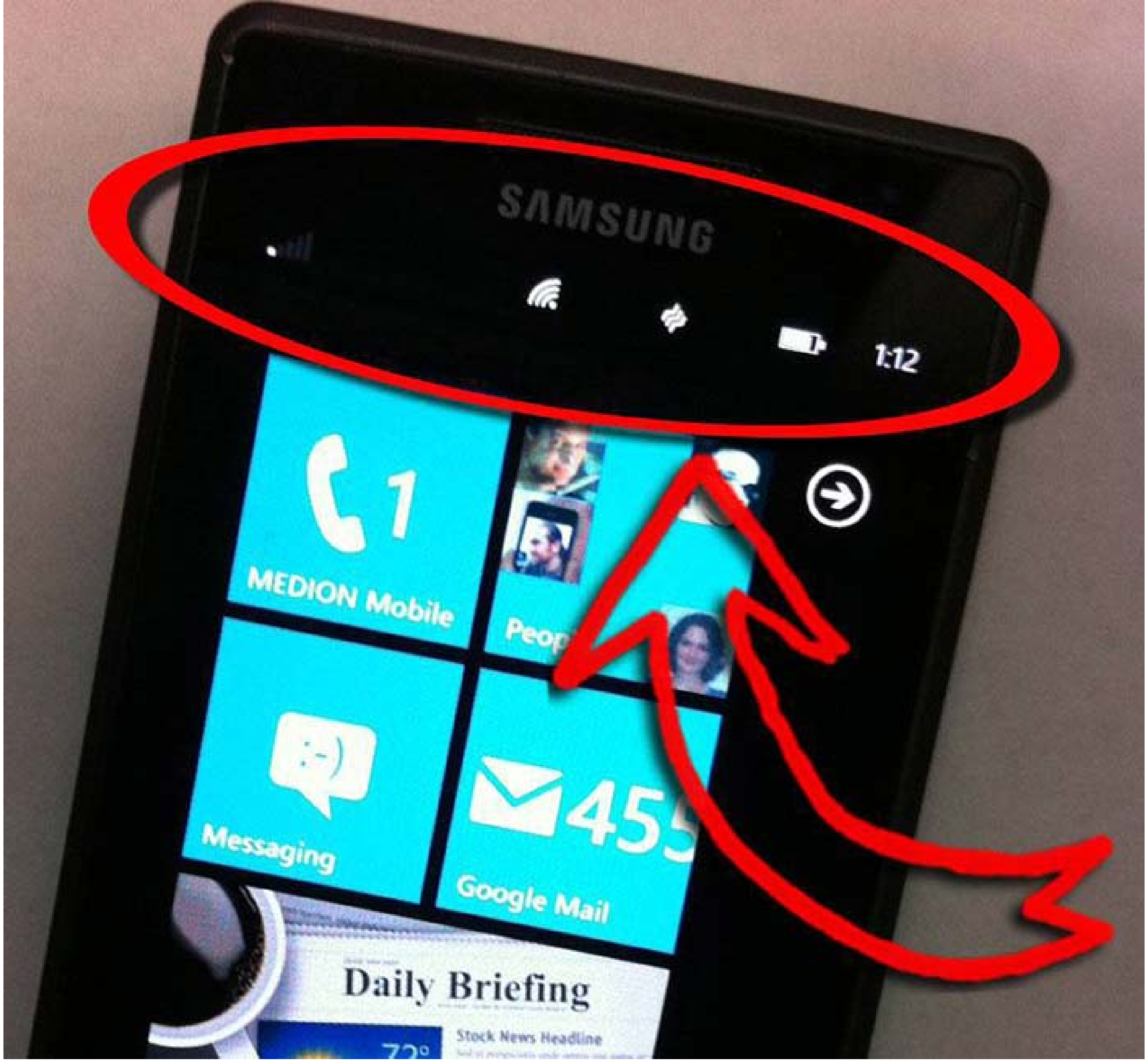
- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

By considering the example of a car navigation system, we can look a bit more into the details of the UCPM. If you use a navigation system, you will probably have noticed that it works very well if you are in a city you have never been to before. If you use it around the area where you live, you may, however, sometimes be surprised about the route it tries to direct you to. This phenomenon can be explained with the UCPM. Let's assume the context-aware system (right side in Figure 11) is of equal quality in both locations; This means that the difference must be on the user side. The sensory perception (e.g. visual matching of buildings and places you know based on your eyesight) is different in a familiar place and a new place. For the new place, you lack reference points, and the Memory and Experience part in the model differs significantly. In the familiar environment, you will have expectations about which route to take and which way would be a good choice. In the unfamiliar environment, you lack experience and reference points, and hence your expectation is simply that the system will guide you to your destination. The result is that a navigation system that successfully guides you to your destination with a *non-optimal route* will satisfy your expectations in an *unfamiliar* environment, but be frowned upon in a *familiar* environment. A non-optimal route could include taking a detour of a few blocks because the map is out of date, or having to wait at three traffic lights, where you could have alternatively used the slightly longer way over the bridge without traffic lights. In the familiar environment, we have a substantial awareness mismatch, whereas when navigating in new surroundings, we have a minimal awareness mismatch.

Design Hint 2: In the user interface, provide information about the sensory information that is used to determine the context in order to minimize the awareness mismatch.

The quality of context-aware systems, as perceived by the user, is directly related to the awareness mismatch, and a good design aims at designing systems with minimal awareness mismatch. A prerequisite for creating a minimal awareness mismatch is that the user understands what factors have an influence on the system. In the example of a simplistic car navigation system, this factor is only current location and nothing else. In such a case the user knows that the system's reactions are based purely on the current location as well as the destination, and the user may attribute the system's response to these factors. In cases where further parameters play a role - e.g. a navigation system that takes traffic into account - it may be more difficult for the user to understand the causalities behind the system's behaviour. In such an example, the navigation system may suggest different routes in the morning and the evening as the traffic situation is not the same. If the user has no knowledge that the system makes the routing suggestions based on the current location and the traffic situation, it is likely that the user will have a hard time understanding what the system does. As an important rule in the design of context-aware systems, the user should be made aware of the sensory information that the system uses.

There are many examples of devices and applications that provide such feedback, e.g. the type of wireless connectivity in a mobile phone and the symbol for GPS reception. Such hints are essential for the user to understand system behaviour. For example, the user may understand, and accept, that there are significant differences in download speed on a mobile device when supplied with the information that download in some cases happen over the GSM network and in other cases over a WiFi connection. However, if the user has no concept of the difference between a data connection over GSM and WiFi, all this information will not be of much help, and the awareness mismatch remains. Therefore, design hints such as the above-mentioned Design Hint 2 are not absolute rules and do not exempt the designer from doing user studies and usability tests: *Know thy user*, as a popular one-liner goes.



Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.12: Phones provide very simple context-information in the user interface. In this picture the phone has connectivity to the GSM-network as well as to a WiFi base station. Having this information allows the user to better understand system behaviour - for example in the event that the users talks on the phone and the speech quality gets worse after entering a building. Looking at the bars indicating the network strength, the user may realize that the coverage is inadequate. As you have a mental model of the problem and its solution, you move towards a window or back towards the door to regain a satisfactory signal quality.

The very basic idea of sensor-based context awareness is the assumption that similar situations (considered as one context) are represented by similar stimuli. Therefore, sensors may be used to determine contexts based on the assumption that *in similar contexts the sensory input of the characterizing features is similar*. The difficult part is to assess and define what the relevant and characterizing features are. As humans we do this in our everyday activities over and over again. We realize there is meeting in progress when entering a room with people sitting around a table talking - even if we do not know the room and anyone taking part in the meeting. The basic approach is to make an (implicit) analysis of the sensory input received from the surroundings and compare this to situations experienced earlier. Let us assume there is an evening meeting at the University of Stuttgart in the first floor meeting room of the SimTech building, and furthermore that there is a meeting in Lancaster in the InfoLab on the second floor meeting room - both at 10am. Let's compare sensory readings for these two situations and add two further situations: A student lab session in Stuttgart, and cleaning of the meeting room in Lancaster.

	[Situation]			
[Feature]	**Meeting Stuttgart**	**Meeting Lancaster**	**Lab Session**	**Cleaning**
Geographic location:	Stuttgart	Lancaster	Stuttgart	Lancaster
Light on or off:	light on	light off	light off	light on
Number of people in the room:	7	9	8	1
Language spoken:	German	English	German	None
Activity in the room:	sitting	sitting	sitting	movement
Power consumption in the room:	2kw	1,6kw	3,4kw	3kw
Devices in use:	laptop, phone	projector, laptop	laptop, phone	vacuum cleaner

Table 14.1: Example of situations and their characterizing features

If we create a matrix in which we count how many features are the same, we arrive at the results in Table 2. Using this feature set, we see that the Meeting in Stuttgart is more similar to a Lab Session than to another meeting in Lancaster. If we would choose another set of features, we would get different similarities. This illustrates how important it is to choose the right features for classification. It is important to find the specific features for a context, and in many cases adding further features may be counter-productive.

[Similarity]	**Meeting in S**	**Meeting in L**	**Lab Session**	**Cleaning**
Meeting in S	7	1	4	1
Meeting in L	1	7	2	1
Lab Session	4	2	7	0
Cleaning	1	1	0	7

Table 14.2: Counting similar features for each pair of situations. It becomes clear that just counting any set of features is not going to work well. Choosing the "right" features that are characteristic is essential.

The general approach is to look at which sensor input you expect in a certain context. In Table 3, two examples are given. A meeting is detected when several people are present and when these people interact. When the sensor information indicates an ongoing change in light and a certain audio level, as well as an indoor location where the user is stationary, we assume the user is watching TV. The examples show that the expected sensor readings are related to a feature space, described in Figure 10. Looking at these descriptions of the expected sensor input, it is apparent that the detection is never perfect. It is easy to create situations that are not a meeting, but classified as a meeting (e.g. having lunch together is likely to be classified as a meeting with the description below). Similarly, we can create a situation that belongs to the context, but is not recognized with the expected sensor input. If the user watches TV while in the garden and perhaps even uses subtitles and has the sound switched off, this would not be recognized.

Such descriptions can be made on very different abstraction levels (e.g. people are present vs. the passive infrared sensor indicating movement). The used descriptions are typically depending on the types of sensors assumed.

Context	***Expected Sensor Input***
Meeting	Several people present Interaction between people
User watching TV	Light level/color is changing, certain audio level (not silent), type of location is indoors, user is mainly stationary

Table 14.3: Illustrates example assumptions made for specific sensory inputs on two contexts

Design Hint 3: Find parameters which are characteristic for a context you want to detect and find means to measure those parameters

In current systems, a wide variety of sensors are used to acquire contextual information. Important sensors used are GPS (for location and speed), light and vision (to detect objects and activities), microphones (for information about noise, activities, and talking), accelerometers and gyroscopes (for movement, device orientation, and vibration), magnetic field sensors (as a compass to determine orientation), proximity and touch sensing (to detect explicit and implicit user interaction), sensors for temperature and humidity (to assess the environment), and air pressure/barometric pressure. There are also sensors to detect the physiological context of the user (e.g. galvanic skin response, EEG, and ECG). Galvanic skin response measures the resistance between two electrodes on the skin. The value measured is dependent on how dry the skin is. Typically, such measurements can be used to determine reactions that change the dryness of the skin, e.g. surprise or fear (lie detectors are based on similar mechanisms). In principle, one can use all types of sensors available on the market to feed the system with context information.

Accelerate your career: Get industry-trusted Course Certificates

Est. 2002

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

"Mobile User Experience (UX) Design"

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder, Interaction Design Foundation

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Head of Instruction, Interaction Design Foundation

BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner's Guide

INTERMEDIATE UX COURSES

- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

In some applications, it may make sense to use more sensors of the same type to ease the context detection task. For example, to determine the number of speakers and locating their position in a room is straightforward with a set of microphones, whereas this is impossible with a single microphone. The quality of the information gained may also be improved by using a set of sensors rather than one. A simple example is that with a single light sensor one can only detect the light level in the environment, but a larger number of light sensors are the basis for a camera.

To match sensory information with contexts, a matching has to be performed. These perception tasks are typically done by using means of machine learning and data mining. The simplest way is to describe a set of features that define a situation. Then, in any given situation, the system will monitor its sensory input and check if the features match the sensory input. Simple rule-based systems fall into this category. Another example is to record typical situations and calculate representative features for these situations. In a new situation, the features are calculated and compared to the learned (recorded) situations. With a simple so-called "nearest neighbour matching", the current context can then be calculated.

The quality of the algorithms that calculate the contexts should be assessed to determine how well the system works. These algorithms can be optimized for precision or recall, similarly to classical information retrieval systems. When assessing context-aware systems, it is important to take the probability of a certain context into account; otherwise very rare events may be missed. Assume the following example: You want to build a fire alarm, and you assume that in 10,000 days you will have 1 day where there is a fire. If you pick up a stone from the ground and announce that it is a fire alarm, you can be pretty sure that it does not work as such. Nevertheless, you can still claim that your "fire alarm" will work in 99.99% of the time. However, when providing information for the context "fire (0%)" and the context "non-fire (100%)", it becomes immediately obvious that a stone is not a useful device for this purpose. To assess this, a confusion matrix is used. It shows the relationship between the real situation and the perceived context for each context defined.

	Perceived context by the "system"	
	Fire	No Fire

Situation in the real world	Fire	0 %	100 %
	No Fire	0 %	100 %

Table 14.4: Confusion matrix for the stone

		Perceived context by the system	
		Fire	No Fire
Situation in the real world	Fire	100 %	0 %
	No Fire	0 %	100 %

Table 14.5: Confusion matrix for an optimal fire alarm

Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.13: In order to know how well a context detector works, you need to know the recognition performance for each context. In comparison to an actual fire alarm, you most likely agree that a stone will not work well as a fire alarm.

14.6 Using context in applications and user interfaces

When sensory information is available, and a given context can be determined as a result, various functions and behaviours of systems and applications can be linked to contexts. As mentioned earlier, the main motivation for calculating/establishing a given context is to increase a system's understanding of its surrounding environment. This enables the designer to create systems that act differently in different contexts, and if they are well-designed they match the user's expectations in this context, i.e. an *awareness match* cf. Figure 11.

Different behaviours can be designed on different levels within a system and range from low-level functionality (e.g. selecting the most appropriate network protocol for the current context), to application behaviour and supported functions (e.g. a mobile device used within the company network may access all company documents, whereas the same device used outside the company may only access a subset of documents), to changes on a user interfaces level (e.g. the zoom level of a map is dependent on the speed with which a car is driving). Often it is hard to clearly discriminate in which category such adaptive behaviour falls.

We generally discriminate between the following types of context-awareness:

- Context-adaptive systems - proactive applications, function triggers, and adaptive applications
- Adaptive and context-aware user interfaces
- Managing interruptions based on situations
- Sharing context and context communication
- Generated data for metadata and implicitly user-generated content
- Context-aware resource management

These basic categories help in the design of context-based applications. In some cases there may not be a clear discrimination between them, or they may be combined in a single application. As early as in the original paper by Schilit et al (1994), a discussion and a table of how context can be used was included. They included a table where they discriminated between what is context-dependent (information or commands) on one side and how context is used (manually or automatically) on the other side. This view of context-aware applications mainly reflects the first and the last types from the above list, i.e. proactive applications and resource management.

It is highly recommended to read this paper by Bill Schilit et al (Schilit et al 1994) as it is the cornerstone and central foundation of context-aware computing. If you are interested in more details on the original work on context-awareness, you may want to read Bill Schilit's PhD thesis.

	manual	automatic
information	proximate selection and contextual information	automatic contextual reconfiguration
command	contextual commands	context-triggered actions

Table 14.6: A basic taxonomy of how to create context-aware systems was introduced by Schilit et al in 1994.

14.6.1 Context-adaptive systems - proactive applications, function triggers, and adaptive applications

Proactive applications take initiative on behalf of the user, based on the environment and context. An example is a heating system that pro-actively starts heating the house when the context *user on her way home* is detected. A further example is a system that automatically launches a bus timetable application when the user is starting the device at the bus stop. The basic idea of proactive applications is that the system anticipates – based on context – what application the user will need and already executes it. Technically, these are sometimes referred to as triggers. A context *triggers* the launch of an application.

Adaptive applications are conceptually very similar. The main difference is that *functions* are triggered based on context rather than on complete *applications*. However, the granularity of applications and functions can differ greatly; hence the discrimination is not of great relevance. When creating context-adaptive systems, the basic approach is first to define a set of contexts that are relevant for adaptation, then to select a set of functions or applications that are used, and lastly to create a mapping between the contexts and the functionalities. This can be done by making a table like Table 7, which applies to an adaptive "home screen." The mapping also defines in what way the trigger is executed. In this example, most of the application functions are executed “On Entry” which means the function is triggered when the user enters the context. When the user switches to another application, the trigger will not be repeated. The trigger named “Always” will make sure that the triggered application is repeatedly called; in our case, the map will always be shown on the home screen when the user is in the car. In case the user switches to another application, the trigger will check every 30 seconds and switch back to the map (given there is no user activity). A further example is a trigger named “Every 60 Seconds." Such a trigger is useful to continuously (in this case every 60 seconds) call a function while the user remains in a certain context. The final example of a trigger is “On Exit” which calls a function when the user leaves a context. There is no need for a formal description, but creating a table as shown below helps in the design and implementation of the application. The table also shows that contexts can be present in more than one row as some contexts require more than one function to be triggered. Similarly, functions are not exclusive to one context; a function can be triggered by many contexts.

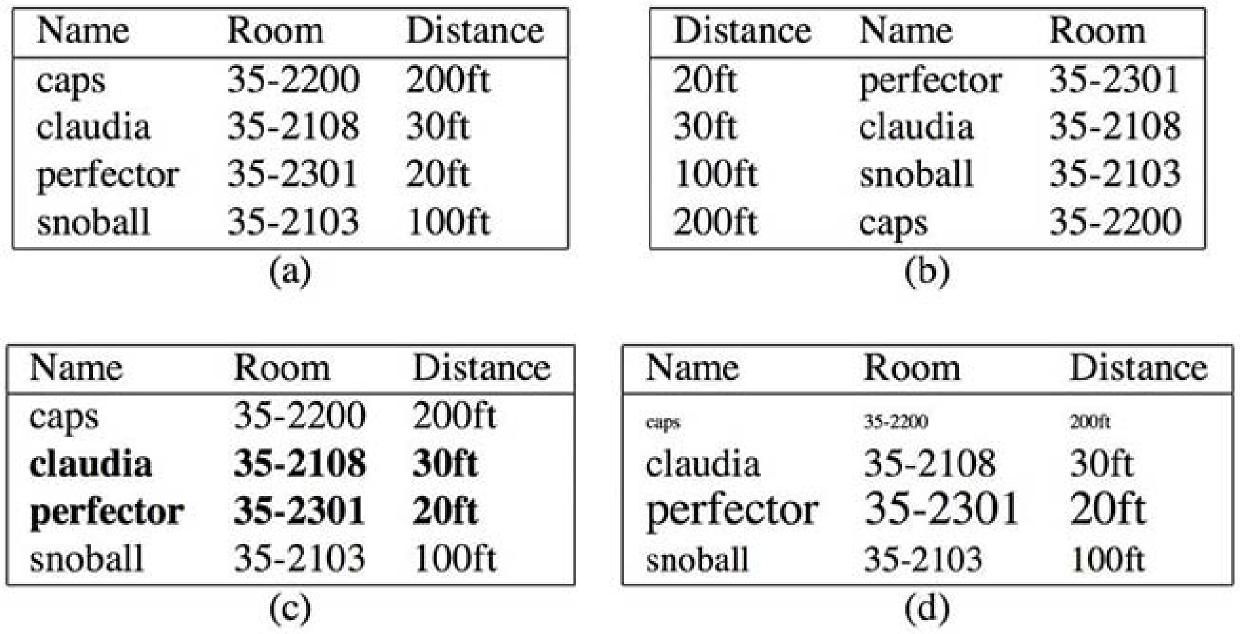
Context	**Timing/Trigger-Mode**	**Triggered Function**
In the office	On Entry	Show calendar on home screen
On the bus	On Entry	Run Music Player
In the car	Always, check every 30 second	Show map on home screen
In the car	Every 60 Seconds	Submit current location to web service
At home	On Entry	Show Facebook messages on home screen
At home	On Exit	Show todo and shopping list
At the gym	On Entry	Run Music Player

Table 14.7: Context-Function mapping for a sample application

Design Hint 4: Designing proactive applications is very difficult because the system has to anticipate what the user wants. In many cases, it is much easier not to present “the application” and rather to present a set of potential interesting applications which the user can launch. For example, a context-aware "home screen" may offer a selection of applications that are useful in a given context rather than attempting to choose the right one.

14.6.2 Adaptive and context-aware user interfaces

Context-aware user interfaces are a special case of context-aware functions. Basically, this means a system where the context-aware functions are user interface elements. The level of complexity in adaptation and awareness may differ greatly. A very simple example of a context-aware user interface is the back light of a device that is switched on when the environment is dark. Further examples are audio profiles that suit a particular situation or screen layouts optimized for a given context. On a mobile device, the input modalities may be dependent on the context. For example, in a car a mobile device may use a simple menu with a large font that can be activated by simple voice commands, but the same mobile device may present a more complex user interface when used during a meeting. The paper by Bill Schilit et al (1994), which originally introduced the idea, uses an example of a user interface to select a printer. In Figure 14 options are shown how to present the printer selection menu when *proximity* is available as context. This seems an obvious way of presenting the list of printers – but even today, decades later, we do not see this approach in operating systems.



Name	Room	Distance
caps	35-2200	200ft
claudia	35-2108	30ft
perfector	35-2301	20ft
snoball	35-2103	100ft

(a)

Distance	Name	Room
20ft	perfector	35-2301
30ft	claudia	35-2108
100ft	snoball	35-2103
200ft	caps	35-2200

(b)

Name	Room	Distance
caps	35-2200	200ft
claudia	**35-2108**	**30ft**
perfector	**35-2301**	**20ft**
snoball	35-2103	100ft

(c)

Name	Room	Distance
caps	35-2200	200ft
claudia	35-2108	30ft
perfector	35-2301	20ft
snoball	35-2103	100ft

(d)

Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms *below.*

Figure 14.14: The user interface design show different ways of presenting context (in this case proximity of the printer) to the user; from (Schilit 1994)

Context-aware user interfaces may include both output and input as well as various modalities. Making user interfaces context-aware provides means for creating a user experience that is tailored specifically to each context. This is, however, not as easy as it may sound. Users learn how to use the user interface and adapt their behaviour to it. Users will remember where a menu item is located or which navigation task to perform to get to a specific function. By making the placement of user interface elements adaptive and the structure of user interface dynamic, we make it harder to learn to use a given user interface. If adaptive presentations are not understandable, e.g. by making the underlying causality behind the adaptation clear to the user, it may hinder the user's ability to memorize user interfaces and to interact efficiently.

Accelerate your career: Get industry-trusted Course Certificates

Est. 2002

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

“Mobile User Experience (UX) Design”

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder, Interaction Design Foundation

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Head of Instruction, Interaction Design Foundation

BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner’s Guide

INTERMEDIATE UX COURSES

- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

Taking menu items in a WIMP-style interface as an example, one could argue that menu items should be reordered according to their usage frequency and unused items should disappear. Adaptive menus have been available in earlier Microsoft Office versions, and it proved to be a bad idea – it confused many people and made it much harder to explore and learn to use the product. The subsequent version of Microsoft Office combined stability and context-awareness by keeping all menu items visible but graying out functions that are not available in the current context. This shift worked very well.

Design Hint 5: Use adaptation in the UI with great care and ensure that it is understandable to the user. Good designs maintain stability and support the user in memorizing the UI while using context to reduce complexity

14.6.3 Managing interruptions based on situations and sharing context in communication

Interruptions happen all the time. You write an email, and in the middle of the sentence you receive an SMS. Interrupted by the notification tone, you switch your attention to the phone screen and read the message. Then you look back at the email … and you cannot remember what you wanted to write. Such situations are common, but most of the time they are not critical, and we have learned to cope with them.

Looking at the interruptions, we can state a basic fact: Computers and communication devices are very rude. Imagine you are in line at the library. You are the third in line, and you wait patiently while she deals with the people in front of you. Suddenly a person walks in and goes directly to the front of the queue, cuts off the conversation between the librarian and the person first in line, and asks about the book she has ordered a week ago. This happens rarely, and everyone would be annoyed with the person. When it comes to telecommunication, however, this happens all the time. If two people are having a face-to-face conversation and one of them receives a phone call, that person is very likely to switch her attention to the call and interrupt the face-to-face conversation. This behaviour is perhaps rooted in the old model of synchronous telecommunication where phone calls were expensive and important – which is less true nowadays. Also, before the advent of caller-id, you could not simply return the phone call as you would not know who had called you unless you actually answered the call. Although technology has changed a lot, some of our behaviour around new technologies are still rooted in an understanding of older technology.

There are several ways context can be used to minimize interruptions. For example:

- Context as a source to schedule interruption and communication
- Context sharing to guide the timing of communication

Using context information, we can decide when and how to deliver asynchronous communication. In the example of the SMS interrupting the process of email writing, one can imagine that the notification is postponed until the user presses the send button in the email program. Another option is to change modality, e.g. instead of having the phone deliver the audio notification, we could have a notification like a bubble in the status bar on the computer, which would be less intrusive to the current task. This example shows, however, that there is a trade-off in the design. If we postpone the notification, the user may get the message too late, and if we use an additional or alternative modality, we may still interrupt the user.

In synchronous communication, automation is really hard to achieve. In this case, context sharing is a promising way of using context to improve the user experience. In Schmidt et al (2000), we introduced the idea of publishing one's context to potential calls - e.g. a status like "I am in a meeting" - and leaving the decision to call or not to the caller. The rationale is that only by knowing the caller's context as well as the context of the person to be called, one can decide whether or not the connection should be established. For an introduction of context-aware communication, see (Schmidt et al 2000) and (Schmidt et al 2001).

With the advent of social media services like Facebook and Google+, it has become possible to use context in mainstream communication systems. For example, some phones integrate the address book with Facebook status and location information of the users.

14.6.4 Generated Data for Metadata and Implicitly User Generated Content

It is obvious that whatever we do, we do in a context. If we write a paper – we do it somewhere, at some time of the day, after another activity, or before another activity. We may be together with other people while we write it, and we are likely to look up other material while we write. Currently, the text we write does not reflect the context in which it was created. You cannot see that the words you read in this chapter were written during several train journeys. However, if we have context available, we could attach meta-information to each word we write. When later looking at the text, we could look up where it was created and who was present while it was written. One domain for which automatic collection of meta-information is useful is the support of personal memory and personal search. Imagine you look for meeting notes. You may not remember when the meeting took place, but you may remember the place where it was and the people who were present, and that it was late in the evening when it ended. If this meta-information was recorded, you could use it for searching.

Services like YouTube, Wikipedia, Flickr, or Facebook are all examples of media where people generate content and share it. All this content is explicitly generated: videos are recorded, articles are written, and photos are taken – and shared. This explicitly generated content has created a wealth of information and has changed the Web fundamentally.

If we look at context information, there is an equally interesting source of data of user-generated information: *implicitly* user-generated data. If you drive your car from your home to the office, you generate information. Assume you record information from the car (e.g. from the acceleration sensor, vibration sensor, temperature sensor, rain sensor, friction between tires and road) and the navigation system (e.g. speed, location, direction) and share this information on the Web. If a number of people share such context information, it will constitute an entirely new domain of information, ranging from real-time traffic information to road conditions and fine-grained weather reports. Although this scenario is technically feasible, it leaves many questions open with regards to privacy.

14.6.5 Context-Aware Resource Management

Managing resources (beyond the user interface) is only indirectly related to the user experience. The basic approach of context-aware resource management is to optimize the operation of a device and its use of resources based on context. One very prominent example is to maximize battery power by using context information, and another one is to switch between available networks based on the current context. They are often realized in lower layers in the operating system.

One should remember that these adaptations may have an impact on the user experience. Transparent resource management is essential for the ease of use of the system; imagine you would have to select the appropriate base station for your mobile phone communication each time you travel to another part of the city. Even if you would just get a message box for each time you register with a new base station, this would render a mobile phone more annoying than useful. This transparency is great as long as it works. If it does not work, or if the user is puzzled about an automatic adaptation done by the system, we should however provide means to inquire into the problem for the user.

Accelerate your career: Get industry-trusted Course Certificates

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

"Mobile User Experience (UX) Design"

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder, Interaction Design Foundation

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Head of Instruction, Interaction Design Foundation

BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner's Guide

INTERMEDIATE UX COURSES

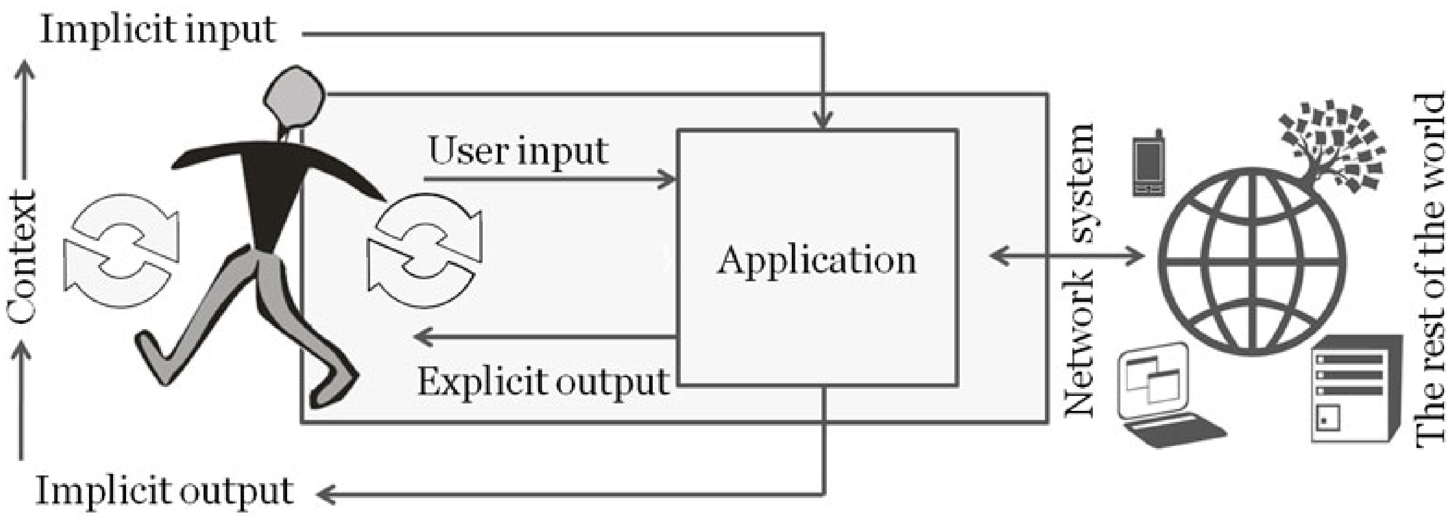
- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

This brings us back to a basic design challenge and a trade-off often faced when implementing context-aware systems: finding the balance between visibility and transparency. And this question is often related to visual design as well as abstraction. Take a simple signal strength indicator of a mobile phone: it shows some context information about the resource "connectivity". In this case, the whole information on the quality of the network interface, including package loss, delays, SNR, RSS, etc. are represented by five bars, and this abstraction allows the user to reason, even if he does not know much about wireless radios.

An interesting area in which context information is used as a resource for gaming and for creating interesting experiences is Contextual Gaming, i.e. playing games in different real-world situations. There is a basic introduction to gaming in context and explanations of concrete examples on how to map context and actions in Holleis et al 2011.

14.7 Implicit Human Computer Interaction

Implicit Human-Computer Interaction, iHCI, (Schmidt 2000) generalizes the concept of context awareness in human-computer interaction. Explicit interaction (traditional, explicit interaction with computers) contradicts the idea of invisible computing and disappearing interfaces. In order to create natural interaction, it appears we need to understand *implicit interaction* in addition to *explicit interaction*. Explicit and implicit interaction can be based on different modalities, including command line, Graphical User Interfaces (GUI), and interaction in the real world. Figure 9 outlines an interaction model taking implicit and explicit human computer interaction into account.



Author/Copyright holder: Unknown (pending investigation). Copyright terms and licence: Unknown (pending investigation). See section "Exceptions" in the copyright terms below.

Figure 14.15: This model explains the concept of implicit and explicit human computer interaction, from Schmidt 2000.

Definition: Implicit Human-Computer Interaction (iHCI)
iHCI is the interaction of a human with his environment, including artefacts, with the aim of accomplishing a goal. Within this process, the system acquires implicit input from the user and may present implicit output to the user.

Definition: Implicit Input
Implicit input are actions and behaviour of humans which are done to achieve a goal, and which are not primarily regarded as interaction with a computer, but captured, recognized, and interpreted by a computer system as input.

Definition: Implicit Output
Output of a computer that is not directly related to an explicit input, and which is seamlessly integrated with the environment and the task of the user.

Implicit human-computer interaction is not an alternative to traditional explicit human-computer interaction; it is rather orthogonal. In order to make computers more attentive and natural to use, we need implicit communication channels between humans and computers. Alan Dix used the term *incidental interaction* to describe a similar concept.

14.8 Summary and Future Directions

Context-awareness is an exciting and challenging area of human-computer interaction. The basic idea is to give computers perceptual qualities (“eyes and ears”) in order to make them recognize the situations in which users interact with information systems. Using sensors, situations can be detected and classified as contexts. Once the system has recognized in which context an interaction takes place, this information can be used to change, trigger, and adapt the behaviour of applications and systems. The input side of implicit human computer interaction looks at information that users generate in order to interact with the real world and thus provides a generalization of context-awareness in human computer interaction.

Creating context-aware interactive systems is hard. One has to keep in mind that users learn how to interact with systems, and that they adapt their behaviour. It is essential that users understand the varying and adaptive behaviour of the application and link it to the situations they are in. Otherwise, they will have a very difficult time learning to use the system. Hence, it is central to create understandable context-aware systems that conform to the users’ expectations. In short: Well-designed context-awareness is a great and powerful way to make user-friendly and enjoyable applications. If done wrong, however, context-aware applications may be a source of frustration. Just think of an automatic light, and you will probably come to think of examples that work very well, and others that do not.

Location-awareness as a special form of context have become mainstream as most medium and high-end phones have a GPS receiver and other means for location detection included. Awareness of your friends’ and families’ whereabouts combined with pedestrian navigation is likely to change the way we coordinate our behaviour, and we will gradually utilize our environment differently as technology changes over the years. Instead of calling someone up to tell that you are late or to ask where to meet, they will have that information readily available (because they are aware of your movements/location). From a design and research perspective, the grand challenge for such a scenario is to create models that allow the individual to control his or her visibility, potentially implicitly, with minimal effort.

With devices that enable us to monitor users in more detail, context-awareness will be included in consumer devices in an ever-increasing degree. Imagine if technologies like cameras and the Kinect, a motion sensing input device by Microsoft for the Xbox 360 video game console, were included into appliances, devices, and your office and home environment. Recognizing where people are and what they do will enable designers to create attentive applications – applications that look at what you do and then react appropriately. The shower will recognize which member of your family is going to use it (e.g. based on the body profile) and pre-select that person's favourite temperature. Designers may explore how appliances can be operated with minimal interaction – potentially just "being there" is enough to work with your environment. Here, a central challenge is to provide means in the user interface to correct wrong choices made by the system, and in a way where the user feels in control.

Accelerate your career: Get industry-trusted Course Certificates

BEGINNER UX COURSES

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

“Mobile User Experience (UX) Design”

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

Founder, Interaction Design Foundation

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Head of Instruction, Interaction Design Foundation

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner’s Guide

INTERMEDIATE UX COURSES

- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

Another interesting area is implicitly generated content. If we live in sensor-rich environments and with sensor rich devices, we have an unprecedented opportunity to create models of how humans live and interact. Collecting GPS traces to create maps is a start - openstreetmap.org is a good example. If we had similar amounts of information about what people eat, how they sleep, and how much they talk to each other, we would be able to arrive at conclusions like, "people who eat an apple between 5 and 7 in the evening sleep 20% better than people who watch a soap opera." Take some time and think this example through … I guess you will come up with many new ideas for systems and applications, but at the same time it may also scare you. It is our responsibility as system designers to make a better and more interesting world with the tools and technologies we have at our disposal – and context is an extremely powerful one!

It is exciting to think about how rich sensing and communication will change the way we live. Together with Kristian Kersting and Marc Langheirich, I wrote the article “Perception beyond the here and now” (Schmidt et al 2011). The magazine article discusses how sensor-equipped computing devices are overcoming longstanding temporal and spatial boundaries to human perception.

14.9 Where to learn more

If you want to learn more about context-awareness there are several good resources. A starting point could be chapter 2 and 3 of my PhD dissertation. In these chapters related work is discussed and an approach for context-acquisition is given.

There are several researchers that have worked, and continue to work, on this topic. Their papers may be a good starting point. For early work and foundations (1994-2001) have a look at the publications by Keith Cheverst, Anind Dey, Jason Pascoe, Bill Schilit, and Albrecht Schmidt.

14.9.1 Conferences

During the last ten years, the field has become broader, and research on context-awareness that relates to Human Computer Interaction has been published in the following conferences:

14.9.1.1 CHI - Human Factors in Computing Systems

2011 2010 2009 2008 2007 2006 2005 2004 2003 2002 2001 2000 1999 1998 1997 1996 1995 1994 1993 1992 1991 1990 1989 1988 1987 1986 1985 1983 1982

14.9.1.2 UbiComp - International Conference on Ubiquitous Computing

2012 2011 2010 2008 2007 2006 2005 2004 2003 2002 2001 2000 1999

14.9.1.3 PerCom - IEEE International Conference on Pervasive Computing and Communications

2008 2007 2007 2006 2006 2005 2005 2004 2004 2003

14.9.1.4 Pervasive - International Conference on Pervasive Computing

2008 2008 2007 2007 2006 2005 2005 2004 2002

14.9.1.5 LoCA - Symposium on Location and Context Awareness

2009 2007 2006 2005

14.9.2 Journals

Springer Personal and Ubiquitous Computing

2009 2008 2007 2006 2005 2004 2003 2002 2001 2000 1999 1998 1997
IEEE Pervasive Computing

2012 2010 2008 2009 2008 2007 2006 2005 2004 2003

14.9.3 Other resources

If you are interested in thinking about, or designing, the future of sensing and context there are many directions to look. One area is reality mining, which relates very closely to implicit content generation as discussed above: For an introduction see reality.media.mit.edu/, and publications can be found at reality.media.mit.edu/publications

14.10 References

Dey, Anind (2000). *Providing Architectural Support for Building Context-Aware Applications - Ph. D. Thesis Dissertation*. College of Computing, Georgia Tech http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf

GUIDE (2011). *The GUIDE Project Home Page - at Lancaster University*. Retrieved 19 January 2011 from GUIDE: http://www.guide.lancs.ac.uk/overview.html

Mitchell, Keith (2002). *Supporting the Development of Mobile Context-Aware Computing - Ph.D. Thesis*. Department of Computing, Lancaster University

Ryan, Nick S., Pascoe, Jason and Morse, David R. (1998): Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant. In: Gaffney, V., Leusen, M. van and Exxon, S. (eds.). "Computer Applications in Archaeology - British Archaeological Reports". Oxford: Tempus Reparatum

Schilit, Bill N., Adams, Norman I. and Want, Roy (1994): Context-Aware Computing Applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications December, 1994, Santa

Cruz, CA, USA.

Schmidt, Albrecht (2000): *Implicit Human Computer Interaction Through Context*. In Personal and Ubiquitous Computing, 4 (2)

Schmidt, Albrecht, Beigl, Michael and Gellersen, Hans-Werner (1999): *There is more to context than location*. InComputers & Graphics, 23 (6) pp. 893-901

Schmidt, Albrecht, Takaluoma, Antti and Mäntyjärvi, Jani (2000): *Context-Aware Telephony Over WAP*. InPersonal and Ubiquitous Computing, 4 (4) pp. 225-229

Schmidt, Albrecht, Langheinrich, Marc and Kersting, Kristian (2011): *Perception beyond the Here and Now*. In IEEE Computer, 44 (2) p. 86–88

Schmidt, Albrecht, Stuhr, Tanjev and Gellersen, Hans-Werner (2001): Context-Phonebook - Extending Mobile Phone Applications with Context. In: 3rd Mobile Human-Computer Interaction Workshop 2001.

Weiser, Mark (1991): *The Computer for the 21st Century*. In Scientific American, 265 (3) pp. 94-104

14.10 Commentary by Keith Cheverst

Context-aware behaviour is standard on modern smart phones and for me it's a little strange (and encouraging) to think that a short ten or so years ago we were still writing research papers on the subject. Albrecht has been an eminent figure on context-awareness during this time and this chapter effortlessly captures both the depth and breadth of the subject matter in a text that is refreshingly easy to digest.

I sneakily used a draft of the chapter as the reading material for a master's level HCI class recently and it worked very well indeed. The chapter provides an excellent historical perspective on the subject matter including the key motivating forces behind context-awareness, namely mobile and ubiquitous computing. It was good to be reminded of the early examples introduced by Bill Schilit such as the listing of printers according to their proximity - something I would still like to see. But Albrecht's own examples presented in the chapter helped to both illustrate key concepts and stimulate much interesting design discussion.

Accelerate your career: Get industry-trusted Course Certificates

Est. 2002

COURSE CERTIFICATE

Victor Johnson

with membership number 1234 has successfully completed the course

"Mobile User Experience (UX) Design"

conducted from Jan 1st 2018 to Apr 2nd 2018

Distinction: Best in class

You can verify this certificate at: https://www.interaction-design.org/victor_johnson

Founder, Interaction Design Foundation

Head of Instruction, Interaction Design Foundation

BEGINNER UX COURSES

- Closes in 17 hrs 54 mins 35 secs—94% booked: Human-Computer Interaction - HCI
- Closes in 6 days—42% booked: User Experience: The Beginner's Guide

INTERMEDIATE UX COURSES

- Closes in 17 hrs 54 mins 35 secs—96% booked: Emotional Design — How to Make Products People Will Love
- Closes in 5 days—49% booked: Mobile User Experience (UX) Design

One important key concept that Albrecht addresses very clearly and which students can struggle with is the hierarchical model by which raw sensor data can be translated into higher level context triggers. While working through the detailed worked example in this section, I found myself musing on how familiar my current class of students are with sensors, such as accelerometers and gyroscopes, compared to a class of just a few years ago.

In addition to its technical/architectural treatment of context-awareness the chapter is also rich in its coverage of 'implications for HCI'. As Albrecht argues clearly, context-awareness provides a powerful tool for the interaction designer. When used well it enables the design of systems that can help 'take the load off' but when used poorly can lead to the production of systems that prove burdensome to say the least.

One of the key challenges faced by any designer wishing to implement context-aware behaviour is that of maintaining predictability and I found myself nodding in agreement while reading Albrecht's discussion of what he terms 'the awareness mismatch' which he succinctly models using the User-Context Perception Model (UCPM).

Another challenging issue for the designer is how best to maintain appropriate levels of control for the user, to keep the user in the loop so to speak. This is especially important for those context-aware systems that implement proactive/adaptive behaviour. As Albrecht states "...a central challenge is to provide means in the user interface to correct wrong choices made by the system, and in a way where the user feels in control". I am reminded very much of an annoying 'habit' exhibited by my current 'smart' phone. I get frustrated when it insists on pausing music playback whenever I place the thing face down. I often place it face down because the speaker is on the back of the phone and so by placing it face down (i.e. 'speaker up') I achieve the most volume. At times like this I wish there was some kind of 'hold' button that would cause the phone to enter a mode whereby all context triggers based on physical actions would be disabled (or rather banished). However, for whatever reason (patent?) the phone doesn't have one and so I don't feel as if I have this control (maybe there is such a feature described somewhere in the manual...). Next to the hold button I would have a 'show me the rule explaining why you just did that' button. As Albrecht states: "...if the user is puzzled about an automatic adaptation done by the system, we should however provide means to inquire into the problem for the user". This reminds me of some of the research by Judy Kay on scrutability (Kay, 1998) within the user modelling domain. At Lancaster we explored this issue with the development of a system that enabled the user to scrutinise (and possibly override) rules inferred by a proactive context-aware system (supporting office environment control) based on context history (Cheverst, 2005). Supporting such user inquiry while maintaining simplicity is clearly hard (and not a simple matter of adding more buttons...). But unchecked the feeling of 'why is IT doing that and how can I stop it!?" can start to feel like a war of attrition with a contrary being. Indeed, when summarising context-aware behaviour, Albrecht states "The basic idea is to give computers perceptual qualities ("eyes and ears") in order to make them recognize the situations in which users interact with information systems". So my phone needs to be smart enough to recognise and discern between the situation where I place it face down because I want greater volume and the situation where I quickly place the phone face down because the phone has started to ring during a meeting and I want it to go silent immediately. Albrecht provides a wonderfully clear worked example to demonstrate how one can procedurally think about the appropriate matching of sensory-based inputs to certain situations, i.e. "to assess and define what the relevant and characterizing features are". So hopefully future designers of context-aware behaviour will take the opportunity to read Albrecht's chapter and produce the kind of smart behaviours that assist rather than hinder and befriend rather than estrange.

References

- Kay, J.: A scrutable user modelling shell for user-adapted interaction, PhD Thesis, Basser Department of Computer Science, University of Sydney, Australia (1998)
- Cheverst, K, Byun, H, Fitton, D, Sas, C, Kray, C & Villar, N, 'Exploring Issues of User Model Transparency and Proactive Behaviour in an Office Environment Control System', *Special Issue of UMUAI (User Modelling and User-Adapted Interaction) on User Modeling in Ubiquitous Computing*, pp. 235-273. (2005)

‹ Requirements Engineering

Usability Evaluation ›

Learnt something new? Share with your friends: