UNIVERSITY OF CRETE
FACULTY OF SCIENCES AND ENGINEERING
COMPUTER SCIENCE DEPARTMENT

COURSE CS-564 (OPTIONAL)

**ADVANCED TOPICS**

**IN HUMAN – COMPUTER INTERACTION**

**Course Convenor: Constantine Stephanidis**

# Ambient Intelligence II
# Pervasive computing, implicit interaction and the context of use

## Overview

- Ubiquitous computing (UbiComp), Pervasive computing , Ambient Intelligence
- Core properties of UbiComp
- Implicit Interaction
- Context of Use / Context awareness

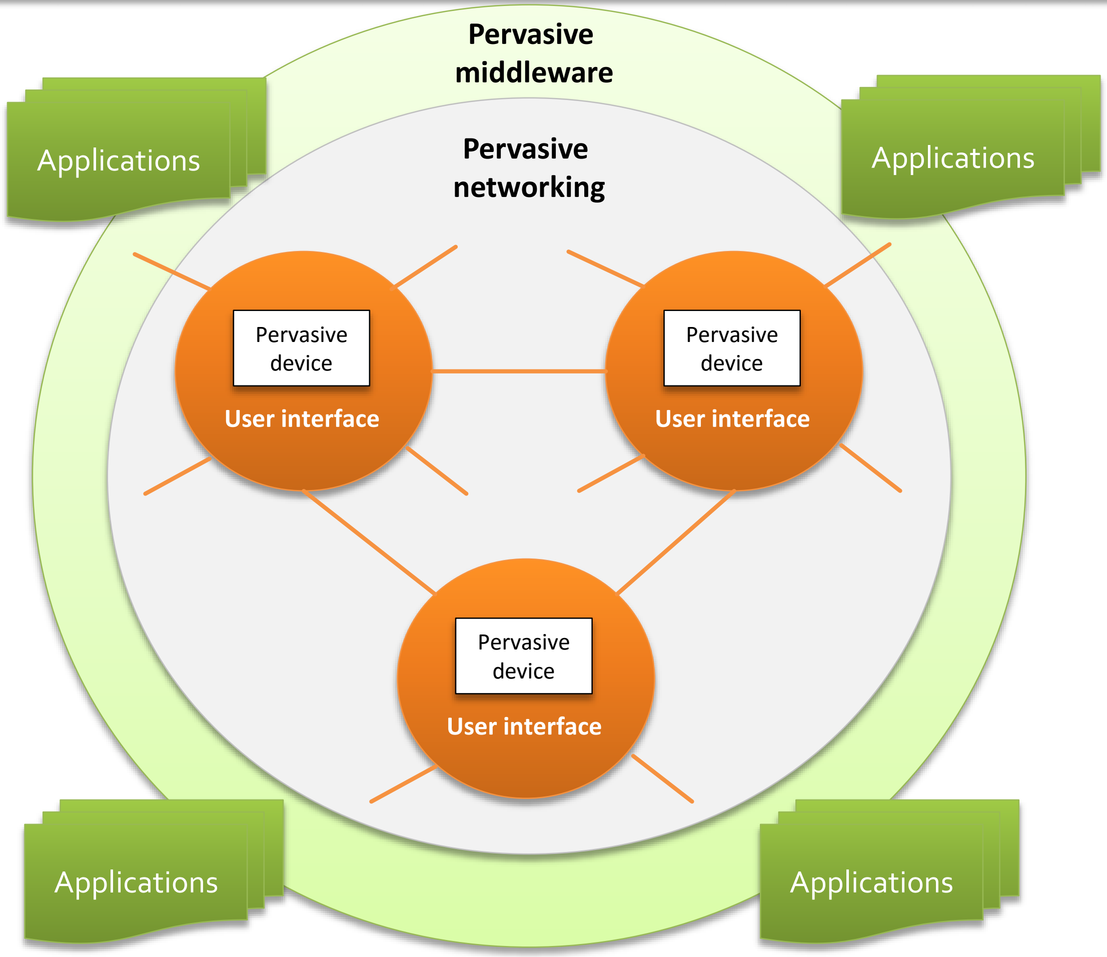
## The origin of pervasive computing

- In 1991, Mark Weiser, then chief technology officer for Xerox's Palo Alto Research Center, described a vision for 21st century computing that countered the ubiquity of personal computers
  - *"The most pro-found technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it"*
- The essence of that vision was the creation of environments saturated with computing and communication capability, yet gracefully integrated with human users so that it becomes a ''technology that disappears''
  - Ubiquitous computing is now also called pervasive computing

## Terminology disambiguation

- The notion of a disappearing computer is directly linked to the notion of "Ubiquitous Computing" (Weiser, 1993), or "Pervasive Computing" as IBM later called it (Krill, 2000)
- Some technical publications equate Ubiquitous Computing, Pervasive Computing, or Everyware Computing (Greenfield, 2006) with Ambient Intelligence
- The nature of Ubiquitous or Pervasive Computing is captured in part by the Oxford Dictionary definition of ubiquitous and of pervasive:
  - **Ubiquitous**: adj. present, appearing, or found everywhere
  - **Pervasive**: adj. (esp. of an unwelcome influence or physical effect) spreading widely throughout an area or a group of people
- Note that while Ambient Intelligence incorporates aspects of context-aware computing, disappearing computers, and pervasive / ubiquitous computing into its sphere, there is also an important aspect of intelligence in this field
- As a result, AmI incorporates artificial intelligence research into its purview, encompassing contributions from machine learning, agent-based software, and robotics
- **Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence** (Augusto, 2007)

# Pervasive computing model



- The technological advances necessary to build a pervasive computing environment fall into four broad areas
  - Devices
  - Networking
  - Middleware
  - Applications

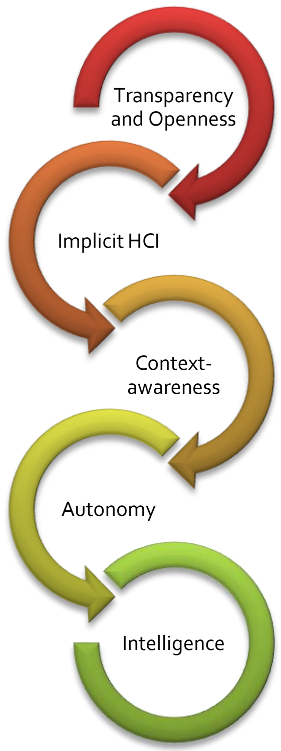
## Networking

- Pervasive computers = Networked computers
  - They offer services that can be locally and remotely accessed
- In 1991, Weiser considered that ubiquitous access via ‘transparent linking of wired and wireless networks, to be an unsolved problem’
  - After a decade of hardware progress, many critical elements of pervasive computing that were exotic in 1991 are now viable commercial products: handheld and wearable computers; wireless LANs; and devices to sense and control appliances
- A range of communication networks exists to support Ubiquitous Computing interaction with respect to range, power, content, topology and design

## Middleware

- Like distributed computing and mobile computing, pervasive computing requires a middleware “shell” to interface between the networking kernel and the end-user applications running on pervasive devices
- This pervasive middleware mediates interactions with the networking kernel on the user’s behalf and keeps users immersed in the pervasive computing space

# Core Properties of UbiComp Systems

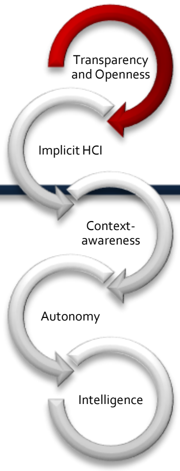


1. Computers need to be networked, distributed and transparently accessible
2. Human computer interaction needs to be hidden more
3. Computers need to be context aware in order to optimise their operation in their environment

Two additional core types of requirements for Ubiquitous Computing systems:

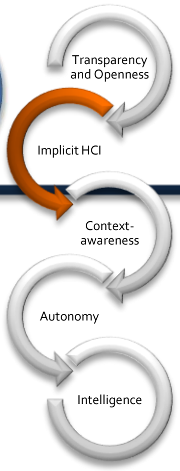
4. Computers can operate autonomously, without human intervention, be self governed, in contrast to pure human computer interaction (point 2)
5. Computers can handle a multiplicity of dynamic actions and interactions, governed by intelligent decision making and intelligent organisational interaction. This may entail some form of artificial intelligence in order to handle:
   a) incomplete and non deterministic interactions;
   b) cooperation and competition between members of organisations;
   c) richer interaction through sharing of context, semantics and goals

## Transparency and Openness



- Ubiquity ⬅➡ Transparency
  - Paradox: how can something be everywhere yet be invisible?
  - The point here is not that one cannot see (hear or touch) the technology but rather that its presence does not intrude into the workplace environment, either in terms of the physical space or the activities being performed
- Openness allows systems to avoid having to support all their functions at the design time, avoiding closed implementation
  - Distributed systems can be designed to support different degrees of openness to dynamically discover new external services and to access them
  - For example, a UbiComp camera can be set to discover printing services and to notify users that these are available. The camera can then transmit its data to the printer for printing

# Implicit Human-Computer Interaction (1/2)



- Because technology by its very nature is artificial, it separates the artificial from the natural. What is considered natural is subjective and cultural and to an extent technological
  - The obtrusiveness of technology depends in part on the user's familiarity and experience with it
- The original UbiComp vision focused on making computation and digital information access more seamless and less obtrusive

## Implicit Human-Computer Interaction (2/2)

- Implicit Human-Computer Interaction (iHCI)
  - iHCI has been defined as ‘an action, performed by the user that is not primarily aimed to interact with a computerised system but which such a system understands as input’ (Schmidt, 2000)
  - “Implicit interactions are those that occur without the explicit behest or awareness of the user” (Ju & Leifer, 2008)
- Reducing the degree of explicit interaction with computers requires striking a careful balance between several factors
  - It requires users to become comfortable with giving up increasing control to automated systems that further intrude into their lives, perhaps without the user being aware of it
  - It requires systems to be able to reliably and accurately detect the user and usage context and to be able to adapt their operation accordingly

## Everyday implicit interaction conversations (1/3)

- Diner:
  - Placing your empty coffee mug at the edge of the table is an implicit signal to the waiter to refill it
  - The waiter might also approach and tilt the coffee pot slightly to signal a question
    - The response might be a slight moving of the mug to signal “yes”
- NYC doorman
  - A doorman sees you approaching and dramatically reaches for the door handle
  - As soon as you see that someone is trying to open the door for you, by instinct you will move away from the door in an arc that automatically signals “no” to the doorman

# Everyday implicit interaction conversations (2/3)

**Counterexample: the weird doorman**

- Imagine a doorman who behaves as automatic doors do
  - He does not acknowledge you when you approach or pass by
  - He gives no hint which door can or will open
    - Until you wander within six feet of the door, whereupon he flings the door wide open
  - If you arrive after hours, you may to stand in front of the doors for a while before you realize that the doors are locked because the doorman's blank stare gives no clue
- This behavior is typical of our day-to-day interaction not only with automatic doors, but any number of interactive devices
  - Our cell phones ring loudly, even though we are clearly in a theatre
  - Our computers interrupt presentations for software updates

## Everyday implicit interaction conversations (3/3)

- While in a  conversation, both the behaviour of participants and what is happening in the surrounding environment supply valuable information
- The robustness of human-to-human communication is based on the implicitly introduced contextual information, such as
  - gestures, body language, and voice

## The environment is the interface

- Explicit human computer interaction → "one person with one computer"
  - the user tells the computer (e.g. by command-line, direct manipulation using a GUI, gesture, or speech input) exactly what to do
- This paradigm is being shifted towards a ubiquitous and pervasive computing landscape, in which implicit interaction and cooperation is the primary mode of computer supported activity
- It is expected that explicit user input will be replaced by technologies that sense the physical world, and control it in such a way that it becomes merged with the virtual world
  - This interaction principle is referred to as *implicit interaction:* input to such a system does not necessarily need to be given explicitly or attentively

## Categories of Implicit Interaction (1/2)

- **Person-to-Person**: technology acts on behalf of people, once a certain relation (e.g., spatial relation like “*close to each other*”) has been verified
  - Example applications are friend finder, or lost and found systems
- **Person-to-Artifact**: provides functional autonomy to the artifact, but still the control remains with user
  - The digital device acts as an entity in the system
  - It can be a virtual object like a software agent working for the system
  - The beneficiary of the system is still the user, but interacting with a digital object instead of a person on the other side
  - Example applications are variants of smart spaces like intelligent classrooms, interactive kitchen, etc., and smart box applications like smart cabinets, smart refrigerators, etc.

# Categories of Implicit Interaction (2/2)

- **Artifact-to-Artifact** is a very undeveloped, yet promising category of interaction mode
- Whether the beneficiary of the application is a human user or the system itself,  autonomous digital objects interact with each other to achieve certain goal
- The end user does not necessarily need to be involved in the interactions among artifacts, this can happen in the background

## Implicit Interaction Concepts

- Concepts needed to facilitate implicit interaction:
  1. **Perception:** the ability to have perception of the use, the environment, and the circumstances
  2. **Interpretation:** mechanisms to understand what the sensors see, hear and feel; and
  3. **Applications** that can make use of this information
- On a conceptual level (1) and (2) can be described as situational context, and (3) are applications that are context enabled

## Identifying Implicit Human – Computer Interaction

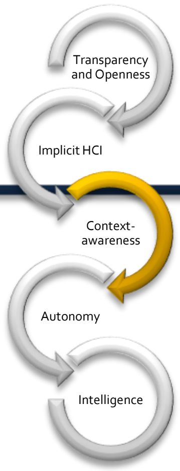
- The following questions help to identify implicit HCI:
  - What happens around an application while the application is in use? Are there any changes at all?
  - Do the surroundings (behaviour, environment, circumstances) carry any valuable information for the application?
  - Are there any means to capture and extract the information in a way that is acceptable for the application or device (processing cost, sensor cost, weight, etc.)?
  - How to understand the information? What interpretation and reasoning is possible and useful? What is an appropriate way for the application to react?
- The concept of contextual information is visible throughout these questions, making context awareness a key factor to implicit interaction

## Quantitative and Qualitative Space

Implicit interactions can be triggered based on:

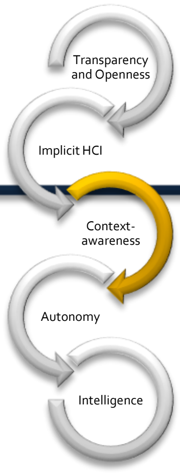
- Quantitative criteria
  - criteria of metric space
  - e.g., exact location, absolute orientation, etc.
- Qualitative criteria
  - relative spatial relations among objects
  - e.g., collocated, left, front, near, far, etc.

## Context awareness (1/2)



- Context awareness refers to the ability of the system to **recognize** and **localize** objects, as well as people and their intentions
- The context of an application is understood as "*any information that can be used to characterize the situation of an **entity/artifact***"
- **Entity/artifact:** "*a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*"

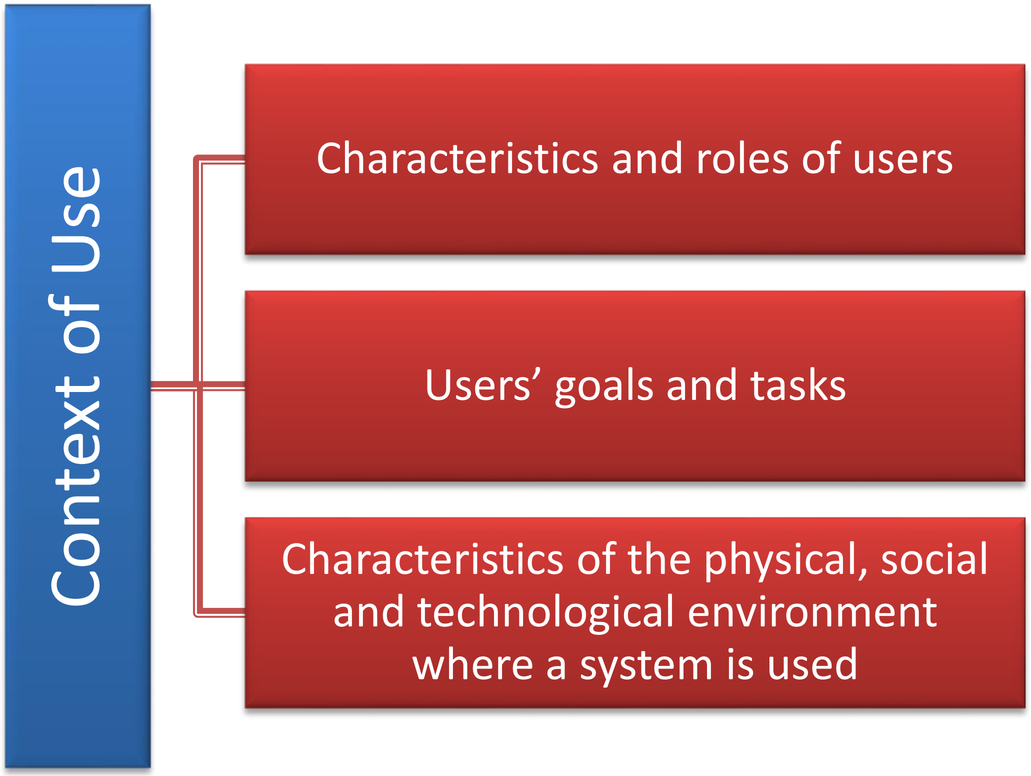
# Context awareness (2/2)



- The aim of UbiComp systems is not to support global ubiquity
  - but rather to support context based ubiquity
- Benefits of context based ubiquity:
  1. limiting the resources needed to deliver ubiquitous services because delivering omnipresent services would be cost prohibitive
  2. limiting the choice of access from all possible services to only the useful services
  3. avoiding overburdening the user with too much information and decision making
  4. supporting a natural locus of attention and calm decision making by users
- A key design issue for context aware systems is to balance the degree of user control and awareness of their environment
  - Design issues include how much control or privacy a human subject has over his or her context.
  - Is the subject aware of the fact that his or her context is being acquired and kept?
  - Does the subject knows who has access to that context or to which parts?

## Context of Use characteristics in traditional HCI

- The context of use (CoU) in user-interface design is usually considered as including the characteristics and roles of users, their goals and tasks, as well as the characteristics of the physical, social and technological environment where a system is used


Context of Use
Characteristics and roles of users
Users' goals and tasks
Characteristics of the physical, social and technological environment where a system is used

## Context of use in AmI (1/2)

- In AmI, the context of use becomes much more complex and articulated,
  - as the number and potential impact of relevant factors increase dramatically with respect to conventional computing devices,
  - particularly regarding the (co-) presence of people, computing devices and other elements in the surrounding environment
- In other words, interaction is no longer a one-to-one relationship between a human, a specific task and a specific device in a static context

## Context of use in AmI (2/2)

- Rather, it becomes a many-to-many relationship between diverse users and a multitude of devices in a dynamically changing environment, where none of the typical context of use factors can be assumed as stable over interaction
- It is believed that the key to identifying suitable design approaches in the context of AmI relies precisely in the interplay between the user, the physical and social environment, the available technologies and the tasks of users
- All these factors are highly dynamic and interrelated and vary over time

## Context of use factors in AmI

### User groups

- Individual vs. group
- Language skills
- Age and gender
- Physical and cognitive capabilities
- Attitudes and motivations
- Cultural background
- Knowledge of the environment applications, services and interactive features
- Domain knowledge
- Interaction preferences
- Emotions and psychological conditions

### Human Activities

- Goals
- Activities
- Output
- Typical location
- Tasks and steps
- Frequency
- Importance
- Duration
- Dependencies and intertwining with other activities
- Required assistance
- Delegation
- Collaboration

### Technological environments

- Available hardware devices, including input/output
- Network
- Software platforms
- Middleware
- Already existing applications and services
- Sensors and other equipment
- Available data from sensors
- Available reasoning resources

### Physical environments

- Time parameters (day, time, etc)
- Auditory environment
- Thermal environment
- Visual environment
- Vibration
- Space, furniture
- User position and posture
- Health hazards

### Social environments

- Activity practices
- Roles
- Interruptions
- Communication structure
- People co-presence
- Privacy threats

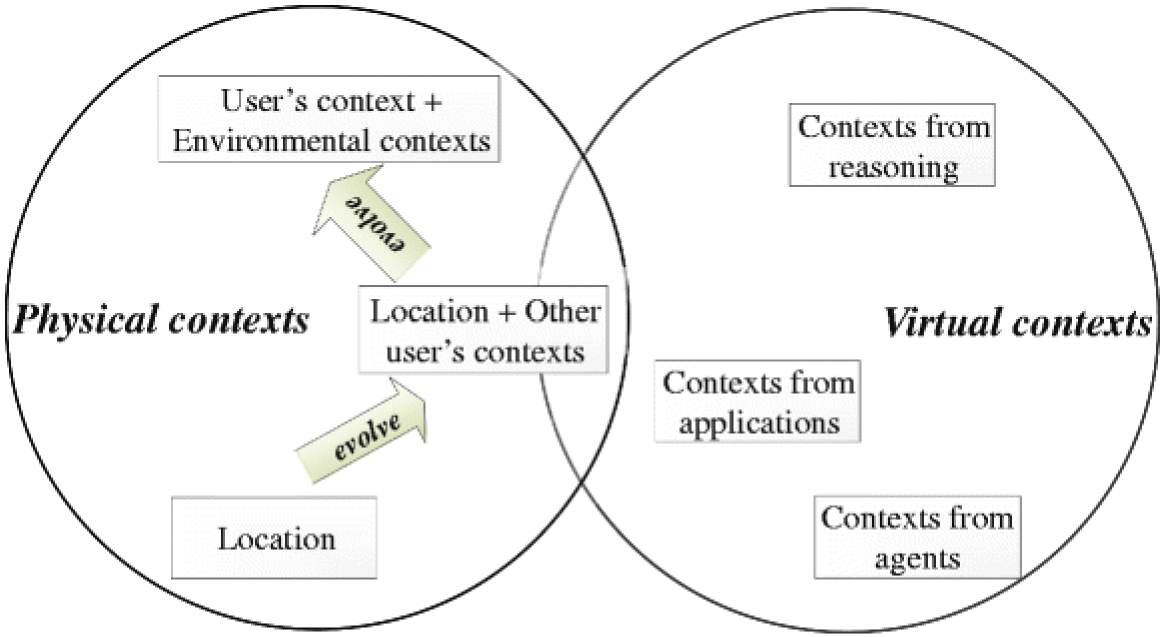
## Context-awareness in AmI

- The definition of the context of use in AmI is also strictly related to the issue of monitoring, as it is critical towards identifying the elements to be monitored, as well as the conditions and parameters according to which monitoring should take place
- In other words, context of use factors are not only what the designers should know, but also what the system should be aware of
- This need is commonly referred to as context-awareness, defined as *"The environment can determine the context in which certain activities take place, where context relates meaningful information about persons and the environment, such as positioning and identification"* (Aarts & de Ruyter, 2009)
  - For example, if a person is in a private situation, the environment should know that it is the user's preference not to be disturbed in such situation, and should not allow disturbing the person

# A context-aware reference framework for AmI

- Zhang et al (2011) among others have the view that contexts refer to pieces of information that capture the characteristics of ubiquitous computing

- They classify contexts into physical and virtual contexts based on context sources


User's context + Environmental contexts
evolve
Physical contexts
Location + Other user's contexts
evolve
Location
Contexts from reasoning
Virtual contexts
Contexts from applications
Contexts from agents

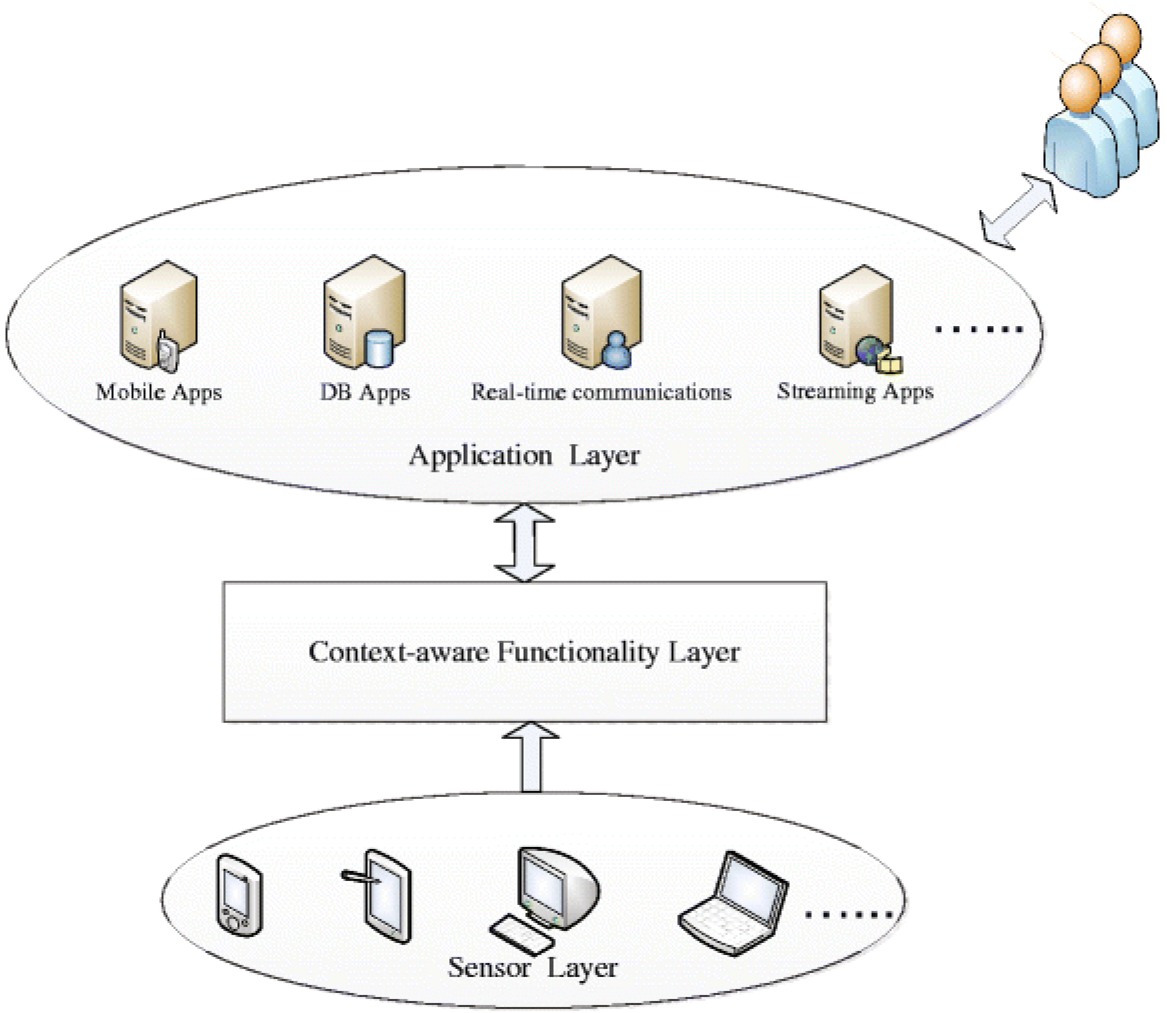
## Physical and virtual contexts

- Physical contexts refer to contexts that can be aggregated by sensing devices
  - E.g., accelerated speed, air pressure, light, location, movement, sound, touch and temperature
    They can be easily gathered by sensing devices, and are widely used in various context-aware applications
- Virtual contexts are contexts that are specified by users or captured from user interactions
  - E.g., user preferences, business processes, goals and tasks
    They enable ubiquitous media applications to be much more adaptive by further understand objects' contexts

## The three layers of context-awareness (1/2)

- According to Zhang et al (2011) context-awareness consists of three layers

1. Sensor layer
2. Context-aware layer
3. Application layer



## The three layers of context-awareness (2/2)

- The Sensor layer acquires contexts from context sources
- The Context-aware layer encapsulates context-aware functionalities and leverages raw contexts to implement these functionalities as context services
- The Applications access the provided services to achieve their respective functionalities

## Context acquisition

- Context acquisition refers to gathering contexts from context sources, which is a prerequisite for context-aware applications
- Contexts can be captured from sensing devices and virtual sources
- The way to implementing context acquisition functionality relies on the application requirements and models
  - It may be either colligated in the sensor layer where sensed data and its interpretation form the contextual information or be implemented as an independent model
- In the sensor layer, there are two primary paradigms to gather contexts: event-driven and query-based

## Event-driven paradigm

- Event-driven paradigm defines a model that a sensor node will immediately send its readings to the sink node when it detects an event
- By allowing asynchronous communication, it satisfies the requirement held in ubiquitous computing in which most devices are resource-constrained. It is applied in Impala and TinyDB
- The representative case of event-driven model is the Pub/Sub paradigm, which exploits the power of the event-driven model that subscribers subscribe sophisticated subscriptions without worrying about how their subscriptions are delivered

## Query-based paradigm

- The query-based paradigm allows database-style query operations to be executed among sensing nodes and sink nodes through a declarative, SQL-like but a distributed interface
  - When an application requires certain kinds of context, it raises a query to sink nodes which turn to aggregating and searching sensor readings reported by nearby sensors
  - Given that sensors keep capturing information and only few readings are useful, query-based acquisition paradigm is appropriate in most cases, therefore it is supported by many projects (TinyDB, SensorWare and CACQ)
- Furthermore, query-based context acquisition paradigm can be easily customized to meet the challenges in ubiquitous computing
  - Consider that devices are seriously limited by power: Cougar improves the query-based model by distributing the query among the nodes to collect the raw contexts and do calculations with minimum energy consumption

## Common contexts captured by sensing devices

| Name | Captured contexts |
|---|---|
| Audio sensors | Noise, music, decibel levels |
| Video cameras | Emotions, presence, behavior |
| Motion detectors | Presence, single or multiple users |
| Pressure sensors | Pressed, occupied, hand gestures |
| Light sensors | Ambient light, indoor brightness |
| Accelerometers | Motion, vibration, physical state |
| Bluetooth | Location |
| Infrared | Location |
| RFID | Location, activity, situation |
| GPS | Location |
| Environmental sensors | Weather, temperature, humidity |
| Event monitors | Events, schedules, notification, errors, updates |
| Action listeners | |
| Data loggers | |
| Agent process | |

## Localization

*Active Bat is a low-power, wireless indoor location system. It relies on multiple ultrasonic receivers embedded in the ceiling and measures time-of-flight to them*

*Ubisense real-time location system*

- Localization technologies aim at the location of persons, objects, artifacts, etc. in the environment
  - Outdoors the Global Positioning System (GPS) is mostly used for fine-grained location sensing, but coarse location information is also available from cellular network infrastructures such as the Global System for Mobile Communications (GSM)
  - Indoors, location sensors are typically embedded in the environment, as in the Active Badge system. Collocation can be sensed with radio beacons

# Components of RFID systems

| Elements | Criteria | Category | Description |
|---|---|---|---|
| Tags | Power source | Active | Sensitive |
| | | Passive | Noisy |
| | In-board memory | Read only | Unchangeable, ex., ID |
| | | Write | Writable, ex., states |
| Readers | Mobility | Stationary | Fixed location |
| | | Mobile | Connected to wireless network |
| | | mRFID | Connected to GSM, GPRS type |
| Systems | Service | Data aggregation | Data processing |

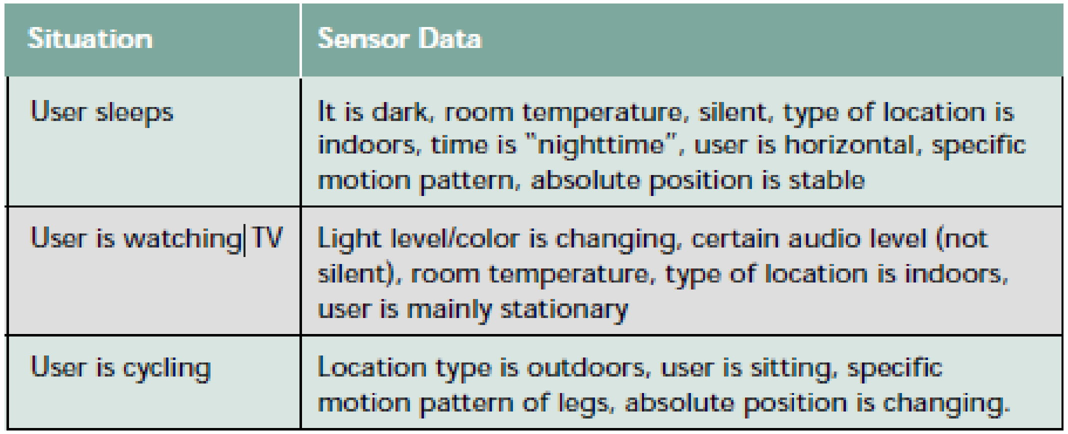
## Sensor Data is Related to a Situation (1/2)

- The basic assumption of the concept of sensor-based context-awareness is:

*"In a certain situation, specific sensor data is frequently similar to sensor data of the same situation at different times."*

- When implementing context awareness the idea is to use the sensor data and predict the current situation or even to forecast a situation
- The design of a system that is context-aware includes consideration of what types of contexts may be useful to recognize and what sensing technology is needed to do this

# Sensor Data is Related to a Situation (2/2)



| Situation | Sensor Data |
|---|---|
| User sleeps | It is dark, room temperature, silent, type of location is indoors, time is "nighttime", user is horizontal, specific motion pattern, absolute position is stable |
| User is watching TV | Light level/color is changing, certain audio level (not silent), room temperature, type of location is indoors, user is mainly stationary |
| User is cycling | Location type is outdoors, user is sitting, specific motion pattern of legs, absolute position is changing. |

Real-world situations related to sensor data

- When looking at what type of sensors can provide significant information to implement context-aware systems, keep in mind that the sensors just give some information about the situation
  - but these can be mapped to contexts, using specific methods (e.g., AI, logic, statistics).

## Context preprocessing introduction (1/2)

- Contexts are often noisy and incomplete in ubiquitous media
  - RFID readers capture 60–70% of the tags in their vicinity
  - Sensors deployed at the Intel Research Lab in Berkeley, on average delivered 42% of the data it was asked to report
- This is because sensed data and their interpretation as contexts are prone to errors
- The sources of errors consist of, but are not limited to internal components, inaccurate measurement, and noise from external environment

## Context preprocessing introduction (2/2)

- Context preprocessing is a context-aware functionality that is in charge of handling raw contextual information
- This functionality significantly improves the context quality, particularly when contextual information is coarse-grained
- Context processing techniques is classified into two types:

1. handling contextual information in the sensor layer
2. handling contextual information in the context layer

- The second type will be examined further

## Context preprocessing paradigms (1/2)

- Generally, there are three context preprocessing designs for contexts sensed by sensing devices:

1. In centralized designs, a central node or a centralized module is required to preprocess contexts captured and delivered by sensing contexts
2. In the distributed paradigm allows sensors to filter and preprocess contextual information by aggregation nodes
3. The hybrid design aims at combining the strengths of the former two paradigms, in which several nodes are selected to aggregate their nearby contexts and then combine their results as final results and report them to requesters

## Context preprocessing paradigms (2/2)

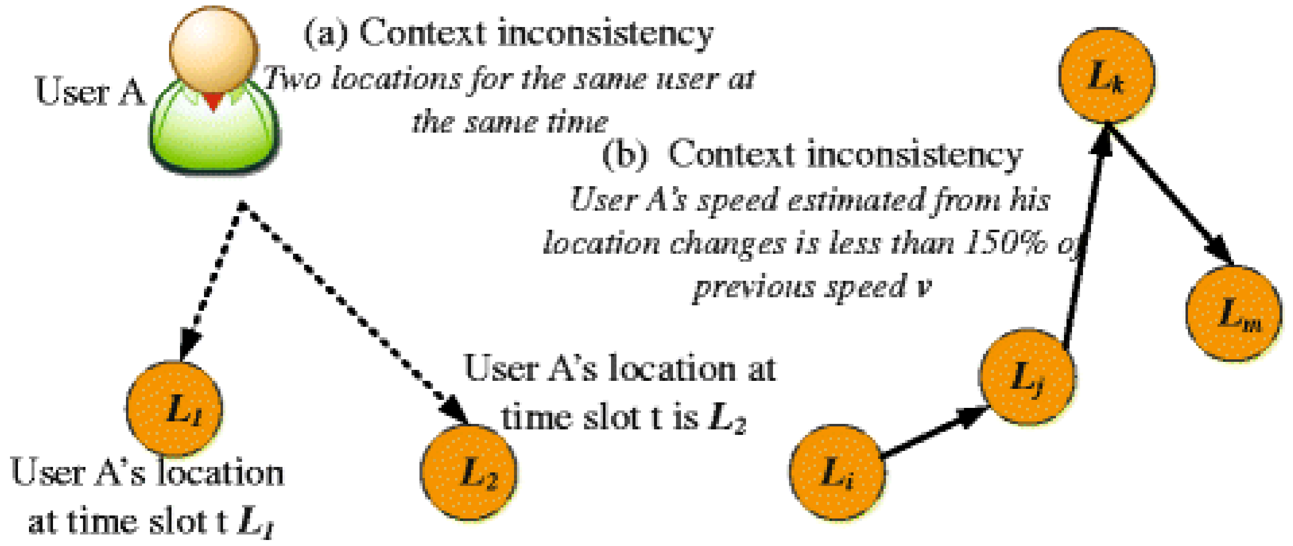
- Context preprocessing can be achieved by event-driven, message-passing or query-based communication paradigms
  - In real-time applications, aggregation nodes are supposed to timely preprocess raw contextual information from sensors, therefore real-time applications usually adopt the event-driven paradigm
  - in applications that are not urgent, e.g., the temperature checking, most sensor readings are redundant. These sensors may filter redundant readings and just notify aggregation nodes about significant contexts. These applications do not require sensors to keep delivering their readings as the real-time applications do. In this case, message-passing is suitable
  - Applications and users may raise some questions about certain contexts. Then, the aggregation nodes had better communicate with sensors by the query-based paradigm
  - On the other side, virtual contexts are similar in context preprocessing with concrete contexts that are collected by sensing devices. They can be handled in the same way as concrete contexts from devices

## Existing context preprocessing efforts

- In the sensor layer, a variety of schemes have been proposed to preprocess raw contexts
  - They mainly employ statistical and probabilistic techniques such as Bayesian network, Kalman filter and linear regression to clean contexts
  - They enable programmers to specify cleaning stages using high-level declarative queries, and transparently translate these queries into the low-level operations necessary to produce results
- On the contrary, preprocessing contexts in the context level is rarely discussed
  - Most context-aware work implicitly assumes that raw data is accurately interpreted as contexts. However, this assumption does not always hold. Context Query Language (CQL) was proposed for contexts aggregation and ontology integration
- To summarize, context preprocessing is mainly limited to processing raw data

## Context inconsistency detection and resolution

- Contexts are often noisy, which necessitates context inconsistency detection and resolution in pervasive computing that compasses various resource-limited devices


(a) Context inconsistency
*Two locations for the same user at the same time*
User A
User A's location at time slot t $L_1$
User A's location at time slot t is $L_2$
$L_1$
$L_2$
(b) Context inconsistency
*User A's speed estimated from his location changes is less than 150% of previous speed v*
$L_i$
$L_j$
$L_k$
$L_m$

## Context inconsistency detection (1/2)

- The majority of existing context-aware efforts are concerned with frameworks that support context abstraction, reasoning and querying. They provide partial support for checking context consistency, and fall into logic-based, ontology-based and tuple-based inconsistency checking schemes
- Logic-based context inconsistency checking schemes:
  - Aura, CARISMA, CASF and Context Toolkit have conducted initial research on checking context inconsistency by using logic rules
  - These projects mainly demonstrate that centralized architectures leverage the development of the functionality of context-aware applications
  - They also demonstrate that message-passing and event-driven paradigms are two appropriate paradigms in implementing context-awareness
  - Thus, **they fall short** of efficiently checking context inconsistency

# Context inconsistency detection (2/2)

- Ontology-based context inconsistency checking schemes:
  - Check context inconsistency by attributes, axioms and assertions
  - However, ontology-based schemes have limited capability of removing noise in contexts and dynamically reasoning out contexts
  - Moreover, they impose a prerequisite that all domain ontologies must be defined beforehand. This usually incurs extra consultant cost
- Tuple-based context inconsistency checking schemes:
  - Check context inconsistency by tuple constraints
  - In Xu et al (2005), a context-aware middleware modeled context constraints by tuples and checked context consistency by semantic matching and inconsistency triggers among elements in tuples.
  - This work is extended in Xu et al (2006), which can incrementally check inconsistency.
  - However, these schemes implicitly assume that the contexts being checked belong to the same snapshots of time
  - This assumption is eliminated in Huang et al (2009) that checks context inconsistency by *happen-before* relationship among contexts in asynchronous environment

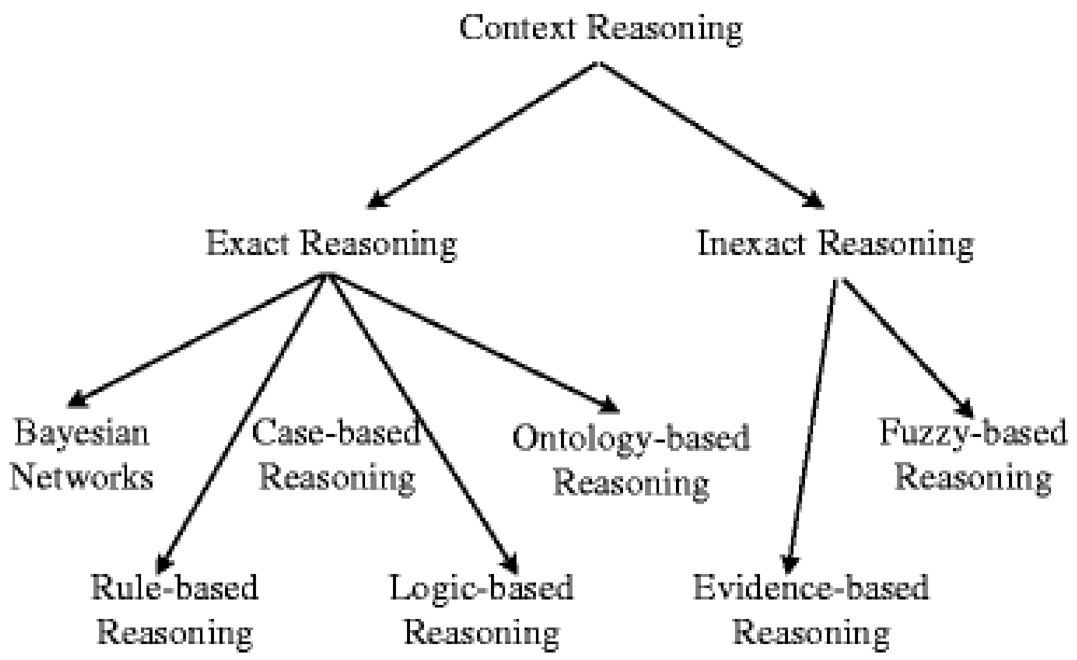
# Context inconsistency resolution

- Regarding context inconsistency resolution, few schemes have been proposed
- According to the manner that solves context conflicts, they are classified into user interference and QoC-based (quality of context) categories
- User interference schemes resort to users to manually solve context inconsistency
  - Users cannot make sure that they are able to find and resolve all context inconsistency
  - Thus this kinds of schemes is neither scalable nor reliable
- QoC-based schemes resolve context inconsistency by context quality
  - They introduce up-to-datedness, trust-worthiness, and other context quality policies according to the context QoC measurement e.g., source location and source state
  - Thus, these policies can resolve conflicts in rapidly evolving contexts. However, contexts are noisy and keeping evolving, which inevitably makes the inconsistency resolution harder
- Context inconsistency checking and resolution is one of the latest hot topics in context-awareness research community

## Context reasoning

- Context reasoning infers implicit contexts from existing contexts
- Many context reasoning schemes have been proposed to achieve context reasoning in a centralized manner or distributed manner
- In a centralized manner, a node makes use of captured contexts to infer implicit contexts by various reasoning techniques
- In a distributed manner, nodes cooperate and make reasoning, e.g., context reasoning based on peer-to-peer ubiquitous networks

## A taxonomy of context reasoning



## Exact context reasoning (1/2)

- It consists of Bayesian network, case-based, logic-based, ontology-based and rule-based reasoning techniques
- Bayesian network can be regarded as a canonical context reasoning technique, which represents contexts by graph and probability
  - However, it is seriously limited by two assumptions. One is that it requires exhaustive and exclusive hypotheses, and the other is its exponential computation overhead
- Case-based reasoning infers contexts based on the past cases
  - However, it falls short of accurately measuring the similarity among cases and automatically generating and analyzing cases
- Logic-based context reasoning is designed for exact reasoning
  - It is inefficient when it is confronted with incomplete and imprecise contexts in ubiquitous media. It is also criticized for its poor performance in fuzzy reasoning. In addition, it lacks of semantics into their representation

## Exact context reasoning (2/2)

- Rule-based reasoning shares the similar idea with logic-based schemes that it infers contexts by pre-defined rules
  - It remains an open issue for rule-based schemes that how to dynamically generates rules according to the varying contexts
- Ontology-based reasoning schemes incorporate semantics into context representation and reasoning
  - They implicitly suppose that all ontologies related to a specific domain are pre-defined
  - However, this supposition is not always true in ubiquitous media
  - Moreover, most users resort to domain experts for ontology knowledge, which incurs extra human costs and hence restricts the application of ontology

## Inexact context reasoning

- Fuzzy reasoning includes evidence-based and fuzzy-based reasoning techniques
- Evidence theory, also known as Dempster-Shafer theory, relaxes two assumptions held by Bayesian network and allows probability assignments to sets or intervals
  - It is capable of constructing the ground truth from pieces of information. Zhang et al (2010) has extended the evidence theory in its two detrimental problems: heavy computation overhead and Zadeh paradox
- To deal with imprecise and incomplete contexts, fuzzy context reasoning employs the fuzzy set theory, which depends on subjectivity in perceiving and representing concepts with member functions, rather than randomness in probability theory and statistics
  - This reasoning technique provides a manner of combining captured data with expert knowledge

## Summary of inexact reasoning schemes

- Inexact reasoning schemes cannot get the accurate implicit contexts and thus they may be unsuitable in ubiquitous media applications that require accurate contexts
- They are seriously doubted regarding their effectiveness by the probability community who prefer statistics, conventional probability and two-valued logic
- In fact, inexact reasoning can be used as a way of inferring possible contexts in most applications

## References

- Schmidt, A. (2000). Implicit human computer interaction through context. *Personal Technologies*, *4*(2-3), 191-199. *(Selected material available through the course website)*
- Zhang, D., Huang, H., Lai, C.F., Liang, X. Zou, Q., Guo, M. (2011). Survey on context- awareness in ubiquitous media. In Multimedia Tools and Applications, 67(1), pp. 179-211 *(Available through the course website)*
- Krumm, J. (Ed.). (2009). Ubiquitous computing fundamentals. CRC Press. ISBN: 978-1420093605
- Ferscha, A. (2009). Implicit Interaction. *The Universal Access Handbook.* CRC Press Taylor & Francis Group, ISBN: 978-0-8058-6280-5
- Schmidt, A., & Van Laerhoven, K. (2001). How to build smart appliances?.*Personal Communications, IEEE*, *8*(4), 66-71.
- Saha, D., & Mukherjee, A. (2003). Pervasive computing: a paradigm for the 21st century. Computer, 36(3), 25-31
- M. Weiser (1991). The computer for the twenty-first century. Scientific American, 165:94–104
- M. Weiser (1993). Hot topics: Ubiquitous computing. IEEE Computer, 26(10):71–72

## Additional references

- P. Krill (2000). IBM research envisions pervasive computing. InfoWorld
- A. Greenfield (2006). Everyware: The Dawning Age of Ubiquitous Computing. Peachpit Press
- Augusto, J. C. (2007). Ambient intelligence: the confluence of ubiquitous/pervasive computing and artificial intelligence. In Intelligent Computing Everywhere (pp. 213-234). Springer London.
- Zhao, R., & Wang, J. (2011). Visualizing the research on pervasive and ubiquitous computing. Scientometrics, 86(3), 593-612.
- Cook, D. J., Augusto, J. C., & Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, *5*(4), 277-298
- Xu C, Cheung SC (2005) Inconsistency detection and resolution for context-aware middleware support. In: ESEC/FSE-13: proceedings of the 10th European software engineering conference. ACM, Lisbon, pp 336–345
- Xu C, Cheung SC, Chan WK (2006) Incremental consistency checking for pervasive context. In: Proceedings of the 28th international conference on software engineering. IEEE, Shanghai, pp 292–301
- Huang Y, Ma X, Cao J, Tao X, Lu J (2009) Concurrent event detection for asynchronous consistency checking of pervasive context. In: Proceedings of the 8th IEEE international conference on pervasive computing and communications. IEEE, Dallas, pp 37–46
- A. K. Dey, G. D. Abowd, and D. Salber, “A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications,” Human-Computer Interaction, vol. 16 (2-4), pp. 97–166, 2001
- Ju, W., & Leifer, L. (2008). The design of implicit interactions: Making interactive systems less obnoxious. Design Issues, 24(3), 72-84
- A. Harter, A. Hopper, P. Steggles, A. Ward and P. Webster (1999). The anatomy of a context-aware application, in: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1999), Seattle, WA (ACM Press, August 1999) pp. 59–68.
- Steggles, P., & Gschwind, S. (2005). The Ubisense smart space platform. na.
- Stephanidis, C. (2012). Human Factors in Ambient Intelligence Environments. In G. Salvendy (Ed.), Handbook of Human Factors and Ergonomics (4th Edition), Chapter 49 (pp. 1354-1373). USA: John Wiley and Sons