Multimed Tools Appl (2013) 67:179–211
DOI 10.1007/s11042-011-0940-9

Survey on context-awareness in ubiquitous media

Daqiang Zhang · Hongyu Huang · Chin-Feng Lai · Xuedong Liang · Qin Zou · Minyi Guo

Published online: 29 November 2011

Abstract Context-awareness assists ubiquitous media applications in discovering the changeable contextual information and adapting their behaviors accordingly. A wide spectrum of context-aware schemes have been proposed over the last decade. However, most of them provide partial functionalities of context-awareness

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61103185, 61003247 and 61073118), the Start-up Foundation of Nanjing Normal University (Grant No. 2011119XGQ0072), and Natural Science Foundation of the Higher Education Institutions of Jiangsu Province, China (Grant No. 11KJB520009). This work is also supported by Major Program of National Natural Science Foundation of Jiangsu Province (Grant No. BK211005).

D. Zhang
School of Computer Science, Nanjing Normal University and Jiangsu Research Center of Information Security and Confidential Engineering, Nanjing, China
e-mail: dqzhang@ieee.org

H. Huang (✉)
College of Computer Science, Chongqing University, Chongqing, China
e-mail: hyhuang@cqu.edu.cn

C.-F. Lai
Institute of Computer Science and Information, National Ilan University, Ilan, Taiwan
e-mail: cinfon@ieee.org

X. Liang
Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada
e-mail: xuedongl@ece.ubc.ca

Q. Zou
School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China
e-mail: qzou@whu.edu.cn

M. Guo
Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China
e-mail: guo-my@cs.sjtu.edu.cn

in ubiquitous media applications. They are specified to a certain task and lack of systematic research on context-awareness. To this end, this survey aims at answering how close we are to developing context-aware applications in ubiquitous media in a systematic manner. This survey proposes a reference framework to identify key functionalities of context-awareness. Then, it investigates the state-of-the-art advances in every functionality of context-awareness. Finally, it points out potential directions in context-awareness research and tools for building and measuring context-aware ubiquitous media systems.

Keywords Ubiquitous media · Context-awareness · Survey

1 Introduction

Ubiquitous computing is a computing paradigm shift where technologies become virtually invisible in our lives [110]. It creates many new opportunities for mobile TV, video conferences and multimedia visualization. For example, in [23, 28], a wireless multimedia system was built by employing sensors with both mobility and multimedia capabilities. The system enabled interactive multimedia services and made adaptive routings according to the contexts, e.g., user geographical location, bandwidth constraint and energy limitation. The goal of ubiquitous media is to build an intelligent media environment embedding computation and communication such that users share media services without awareness of underlying technologies. This intelligence in ubiquitous media is mainly achieved by context-awareness, an enabling technology that assists ubiquitous applications in timely sensing contextual information and adapting to the changeable contexts [99]. Contexts refer to the pieces of interesting environmental information, e.g., user id and location [87, 117].

Research on context-awareness in multimedia has grown dramatically since ubiquitous computing came into being in 1999. From the start, context-aware research was driven by integrated sensing, hardware miniaturization and object localization in smart spaces (e.g., meeting rooms, cars, campus and houses). Figure 1 illustrates a context-aware media infrastructure at the Museum of Nature and Human Activities in Hyogo [93], where four spots are specified for the animal specimen exhibition shown in Fig. 2. Each visitor has an pendant containing an active RFID tag. When a visitor stands in front of a LCD screen, the navigational system will detect the context of the visitor and illustrates course annotations together video introduction.

However, context-awareness requires more—the field requires an interdisciplinary approach, cutting across programming paradigms, operating systems, embedded systems, networking and many application areas. The challenges imposed by the breadth of the field have been enormous. These arise a sharp gap between the high-level requirements from ubiquitous media applications and the operation complexity in handling contexts from the environment. The application requirements cover various aspects, such as high adaptability, flexibility and intelligence. Owing to constrained resources, heterogeneous devices, and user frequent movement, handling contexts would be extremely tedious and error-prone. Along with the rapid evolution in ubiquitous computing, the gap becomes increasingly obvious.

A variety of projects in real-world applications have shed light on bridging the gap and removing the development impediments in designing ubiquitous media

Fig. 1 Context-aware active media at the Museum of nature and human activities in Hyogo, where the museum can intelligently navigate visitors

applications. These projects include smart tabs, pads and boards [111], Oxygen [35], active badge [109], Aura [45], Gaia [76], One.World [47], Pico [65], Easyliving [15], CoolTown [6], DataTiles [88] and ambient intelligence project [40]. However, they fail to explicate logically organizing context-awareness and rapidly building context-aware applications with appropriate designing and developing techniques in the following aspects. First, they provide transparent abstraction by hiding contextual information, but ubiquitous media applications require handling contexts as much explicitly as possible [95, 119]. Second, they partially support context-awareness. They are mainly concerned with how to continuously satisfy the preferences of individuals in the presence of mobility and heterogeneity. In contrast, ubiquitous media applications intend to cover the requirements of scenarios and applications. Thus, mobility and heterogeneity should be handled in a different manner. For

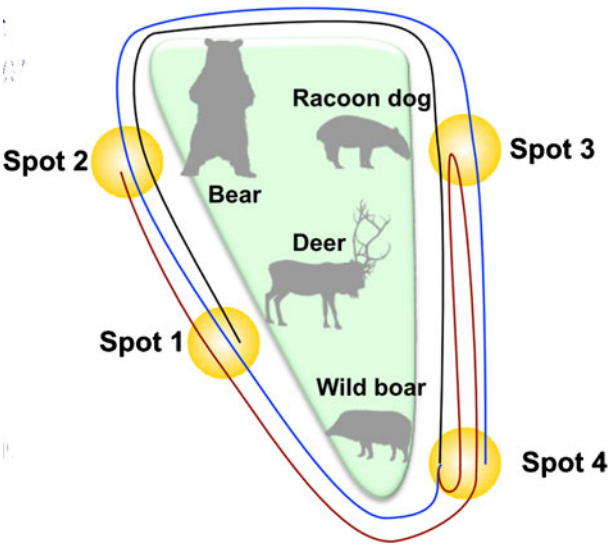


Fig. 2 Four spots in the exhibition room of animal specimens. The media infrastructure dynamically navigates their visitors among spots

instance, ubiquitous media applications will be fault-tolerant and flexible no matter how often users switch on/off a specific scenario. Third, ubiquitous media applications arise several challenging points, e.g., media studies concerns, tangible media design concerns and human-computer interaction [44, 60]. This definitely increases the complexity of implementing context-awareness in ubiquitous media applications.

To this end, this survey studies context-awareness in a systematic study. It explores the connotations and the way to achieve context-awareness. The distinct contributions of this paper are three-fold.

- It proposes a reference framework by which context-awareness in ubiquitous media is decomposed, analyzed and compared from programming abstraction, service and runtime support. This framework facilitates insights into the strengths and weaknesses of different systems and comparisons with a unified model.
- It classifies the context-aware efforts according to the functionalities and services contained in the proposed model. It also discusses the characteristics of these efforts by the classification.
- On the basis of widely investigating existing efforts, this paper identifies the open problems in context-awareness, and points out the future research directions.

The remaining of this survey is organized as follows. Section 2 introduces preliminary knowledge about contexts and context-awareness in ubiquitous media applications. Section 3 introduces a reference framework to analyze functionalities and identifies critical services to be provided by context-awareness. Section 4 overviews the state-of-the-art schemes and techniques corresponding to every functionality and service of context-awareness. Section 5 looks into the advances of runtime support in context-aware projects. Section 6 discusses challenging issues and points out future research directions. Finally, Section 7 concludes this survey.

2 Preliminary

This section firstly introduces what contexts are and then overviews how context-awareness is achieved in most existing ubiquitous projects.

2.1 Contexts

Generally, contexts vary with environment and users such that it is challenging to define them. We introduce contexts according to the context evolution.

- Contexts initially perceived are related to location. Contexts are mainly exploited to develop various location-based tourist services and systems [1, 37, 39, 83, 98].
- Contexts are enriched to incorporate other user information [21, 36, 79, 91]. User emotional state, attention focus and identities of nearby people are such contexts.
- Contexts are further widened to involve environment information [12, 50, 100, 104]. This is because the advent of going into the ubiquitous computing has stimulated broad and contrasting interpretations on contexts.

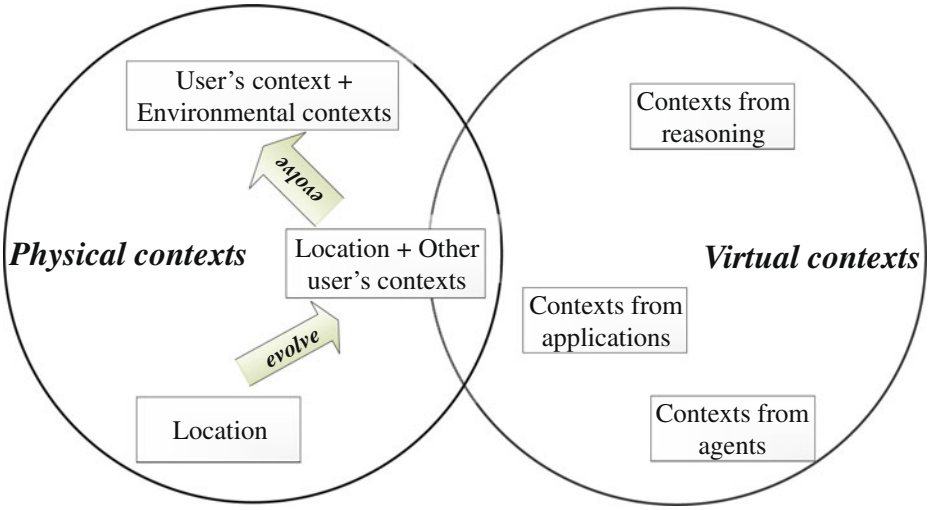


Fig. 3 Context classification, which illustrates our view on contexts into two categories

In this survey, we share the same view with [20, 56] that contexts refer to pieces of information that capture the characteristics of ubiquitous computing. We classify contexts into physical and virtual contexts based on context sources. Figure 3 illustrates our view on contexts, indicating the evolution process of physical contexts and the contents of virtual contexts.

- Physical contexts refer to contexts that can be aggregated by sensing devices. This kind of contexts involves 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, including user preferences, business processes, goals and tasks. They enable ubiquitous media applications to be much more adaptive by further understand objects' contexts [16, 43].

2.2 Context-awareness in ubiquitous media

Context-awareness is a mechanism that assists ubiquitous media applications in adapting their behaviors to the evolving contexts [22, 67]. Suppose a call comes when Alice is watching Kongfu Panda in her smart bedding room. According to the call urgency and the caller's relationship with Alice, this smart space adapts its behavior correspondingly, i.e., reject or accept the call. We investigate this scenario and find that most ubiquitous media applications share a similar way to achieving context-awareness. First, they acquire physical and virtual contexts about Alice, including her identification, location, phone, and social network. Then, they exploits these contexts to determine what strategy should be taken when contexts keep evolving.

We abstract this scenario and get findings that context-awareness consists of three layers—sensor, context-aware and application layers, which is shown in Fig. 4.

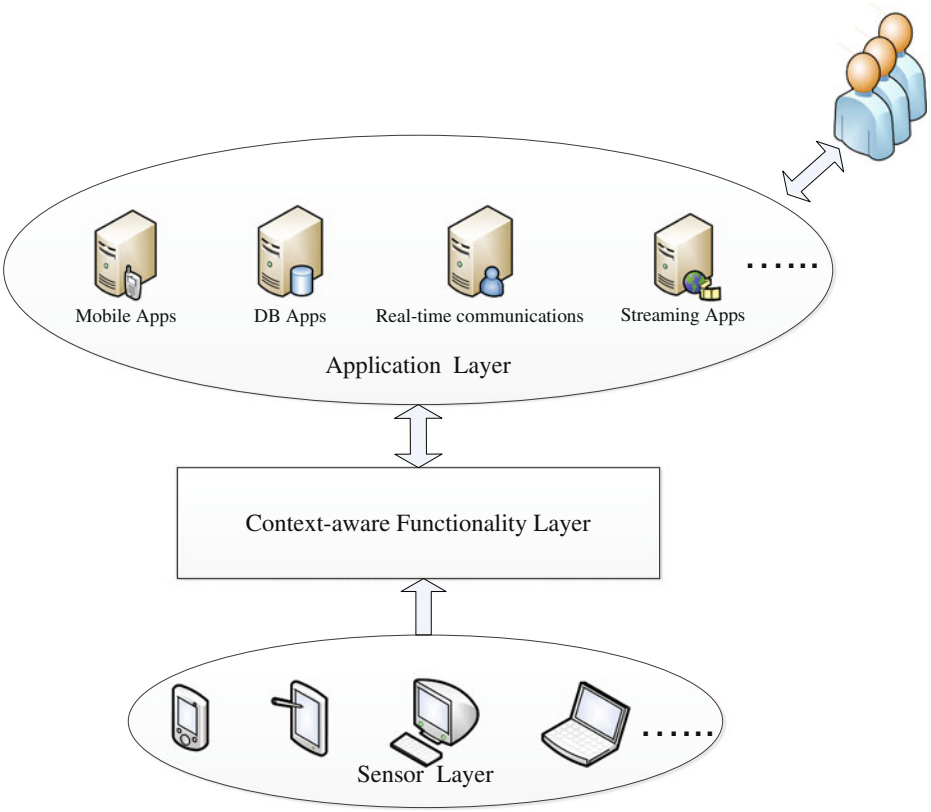


Fig. 4 The way of achieving context-awareness in most ubiquitous computing projects, consisting of three layers—sensor, context-aware and application layers. Sensor layer acquires contexts from context sources, context-aware layer encapsulates context-aware functionalities and leverages raw contexts to implement these functionalities as context services, and applications access the provided services to achieve their respective functionalities

Ubiquitous media is saturated with computing and communication capabilities so that ubiquitous media applications exploits these capacities to capture contexts. Then, applications deal with raw contexts in context-awareness modules, e.g., context preprocessing and reasoning. Finally, applications customize their behaviors according to the changeable contexts.

Note that most projects are bound up with specific tasks, and aim at hiding environment complexity by isolating applications from explicit schedule and management of devices, memory, protocols and heterogeneous problems related to architectures, operating systems, network technologies and programming languages. They satisfy not all users but the individuals, indicating that they may not achieve global optimization in the presence of mobility and heterogeneity. Meanwhile, they lack systematic research upon context-awareness such that they provide neither a series of rules and guidance to facilitate the development of general context-aware functionalities, nor reusable modules and libraries.

3 Reference framework

To identify the crucial functionalities in context-awareness, we propose a context-aware reference framework in this section. We firstly overview this framework in a systematic manner, and then introduce its every component. Throughout this survey, we interchangeably use 'node' and 'user' in the contexts without ambiguity.

3.1 Model overview

We classify context-aware in ubiquitous computing into three dimensions—programming abstraction, services and runtime support. To simplify context operations and facilitate the composition and migration of context-aware services, programming abstraction encapsulates and decouples contexts. It also defines service interfaces for application developers. Services provide implementation to achieve the abstraction. Whereas runtime support serves embedded operating systems as an extension with dynamically supporting the self-adaptation of services. Note that the reference framework assumes that QoS is encapsulated in every functionality of these dimensions.

By investigating the requirements and characteristics of context-awareness, we propose a reference framework shown in Fig. 5. This framework explicates the structure and the relationships of programming abstraction, services and runtime support. The goal of the framework is not only introducing key technologies, but identifying a set of functionalities and services that simplify the construction of new context-aware applications. These three dimensions provide elementary viewpoints

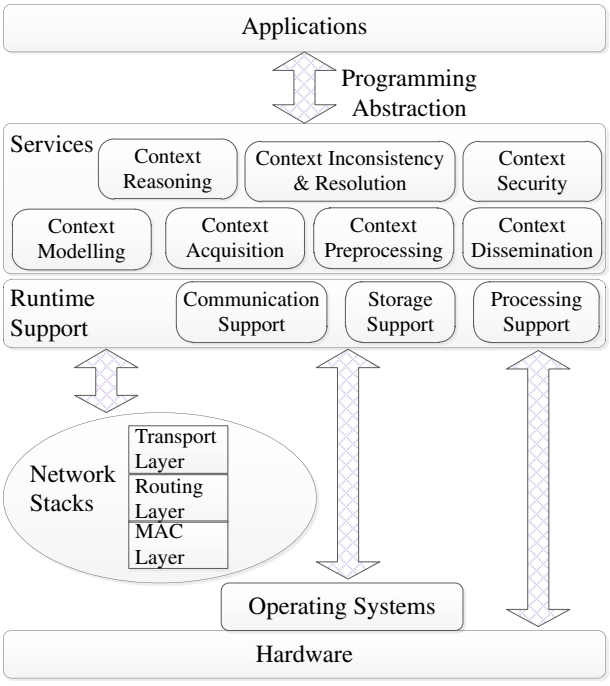


Fig. 5 The context-aware reference framework in ubiquitous computing, which is built on top of hardware, operating systems and networks

to envision context-aware applications, and thus encompass various functionalities. All functionalities may not be implemented in a context-aware module. They may be achieved in various manners or even being embedded. They are usually deployed to sensor nodes, sink nodes, and hand-held devices that may locate in widely different areas. Thus, nodes are capable of cooperating each other to achieve context-awareness.

3.2 Programming abstraction

Programming is an essential part in context-awareness, which forms the basis of implementing context-aware services. It separates the design and development of context-awareness from operations in the underlying ubiquitous infrastructures. It encapsulates the description, design and access of services into three components—abstraction level, programming paradigm and interfaces.

3.2.1 Abstraction level

It denotes that how application developers view the context-aware systems, consisting of two levels: node-level and system-level abstractions. The former abstracts a ubiquitous system as a distributed system including a range of cameras, sensors, hand-held devices, projectors and embedded devices. This level abstraction assists developers in programming individual devices belonging to the sensor layer for their action and cooperation [68, 70, 71]. The latter abstracts all context-aware functionalities as a single virtual module and provides a small set of programming primitives. It enables program developers to develop a single centralized program (global behavior) into subprograms that can be executed on local nodes (nodal behavior). Thus, it makes developers keep away from low-level operations, such as remote code data access, cache management, and node communication and cooperation.

In general, the former abstraction exploits the characteristics of nodes and hence is energy-efficient and saves many communication and cooperation overheads. Meanwhile, this level abstraction requires that developers be with adequate knowledge of nodes regarding hardware and software. Whereas the latter abstraction makes developers deal with actual applications rather than solving multifarious and error-prone issues in nodes. It allows developers to timely capture context changes and to handle them in a much more straightforward way than before. It also enhances code reuse to solve commonly encountered problems [49, 112].

3.2.2 Programming paradigm

It refers to a style of programming regarding how solutions to problems are to be formulated in a programming language, reflecting the view that programmers have of the execution of programs. Programming paradigm is dependent on applications. Given that contexts frequently evolve with individuals and situations in sensor-driven pervasive computing environment [66], we assort contexts into static and dynamic features. Context static features are captured by context acquisition through continuous, event-driven or query-based paradigms, while context dynamic features are gathered by diverse programming paradigms according to concrete requirements of applications. Common programming paradigms include Pub/Sub, message-passing

and concurrent paradigms. As a matter of fact, there are a huge number of programming languages, but fewer paradigms. Each programming language may realize at least one paradigm. Each paradigm is composed of a set of programming concepts, organized into a core as paradigm kernel language.

3.2.3 Interfaces

An interface is an abstraction that an entity provides itself to the outsider, while its type is a tagged constraint on programming interfaces. An example is that object-oriented languages standardize interfaces and their types between providers and requesters. Thus, requesters can handle messages by requiring providers to conform to this standard. This also enables requesters to select providers from a number of candidates, and providers to be extended by being able to accommodate future suppliers that comply with the standard.

There are two types of interfaces—declarative and imperative interfaces. The former assists declarative languages in query operation and file configuration with much resource consumption. It is known as SQL and XML in query operations. The latter helps imperative languages interact with networks and operating systems. It's representatives are C and Fortran based interfaces.

3.3 Services

Services are supposed to meet a significant requirement that context-aware applications should be rapidly built. These applications continuously adapt to highly complex and dynamic environment and functions in the presence of user movement and heterogeneous devices with limited communication and computation capabilities. However, most existing schemes are not qualified for constructing such services because they work in the form of objects and hide the operations on contexts [47, 95, 119]. Thus, users are often forced to adapt, which is antithetical to the vision of ubiquitous computing.

In order to satisfy this requirement, the reference framework proposes a set of context-aware functionalities that are abstracted in context modeling, acquisition, preprocessing, inconsistency detection and resolution, and reasoning. Thus, context-aware applications derive services from these abstractions. These abstractions are summarized in Table 1, embodying the core services that context-aware applications should possess.

Table 1 Functionalities of context-awareness in ubiquitous computing

Functionality	Definition
Modeling	A pattern designed to represent the object "context"
Acquisition	Gathering contexts from context sources
Preprocessing	Manipulating raw contextual information into a format that will be easily and effectively handled by applications, users or processing procedure
Inconsistency detection and resolution	Checking and resolving context inconsistency
Reasoning	Inferring implicit contexts from existing contexts

3.4 Runtime support

Runtime support aims at building execution environment where ubiquitous applications run smoothly with heterogeneous devices, networks, topologies and frequent movement of nodes. It extends embedded systems in terms of task scheduling, process communication, management of memory, power and resources. It is indispensable to ubiquitous applications, which involves many devices and nodes with limited capabilities, power and resources.

Runtime support can be achieved in several ways. It may be implemented as a virtual machine over a specific operating system in the sensor layer [108]. In Mate [68], it is built on top of open-source operating system TinyOS [69].

Thus far we have discussed all functionalities as well as services of the reference model. In the remaining of this survey, we focus on investigating how existing projects implement the functionalities involved in the proposed model, and then identifying challenging problems in context-awareness research community as well as future directions.

4 Context-awareness evolution in ubiquitous computing

This section discusses the existing context-aware techniques in ubiquitous applications. It gives a detailed coverage for most of the recently presented techniques focusing primarily on the functionalities involved in the reference framework. Section 4.1 introduces current context modeling techniques. Section 4.2 describes the existing efforts on context acquisition. Section 4.3 investigates the context processing techniques. Section 4.4 overviews the work on context inconsistency detection and resolution. Section 4.5 describes the latest context reasoning research.

4.1 Context modeling

Context model is a pattern to represent the object "context". Many context models have been developed over the last decade ranging from early simple to sophisticated complex models. We summarize context models based on the data structure used for expression and exchanging.

4.1.1 Key-value context model

Key-value model employs key-value pairs to enumerate attributes and their values describing contextual information, which is recognized as the simplest context model. It has achieved widespread success in early distributed frameworks to represent location information or service functionalities e.g., [99] and [92, 106]. Then, context discovery is applied by checking the matches of key-value pairs.

Key-value pairs are unable to represent complex contexts, although it is easy to manage. Consequently, it gets increasingly less used in the late context-aware research.

4.1.2 Markup context model

Markup model represents contexts by a set of symbols and annotations inserted in a text document to control its structure, formatting or relationships among its parts. These symbols and annotations may spring from the makeup languages with built-in semantics mechanisms (e.g., HTML), or the makeup languages without semantics (e.g., XML). They can be interpreted by devices to control how contexts should be organized and exchanged.

Two of the most popular markup context models are User Agent Profile (UAProf) standard [113] that captures capabilities and preferences for wireless devices, and Composite Capabilities/Preferences Profile (CCPP) that describes device capabilities and user preferences and instructs the adaptation of content presented to devices [114]. UAProf and CCPP as well as their derivatives are seriously limited by the pre-defined hierarchical structure and restricted overriding mechanisms. Besides CCPP and UAProf, there are several other markup context models proposed, such as Context Extension [58], Comprehensive Structured Context Profiles (CSCP) [53], ConteXtML [90], Centaurs Capability Markup Language [62], and the note-tags of the stick-e notes [14]. Nevertheless, these markup context models suffer from several problems in capturing context relationships, dependencies, timeliness, inconsistency checking, reasoning, and uncertainty removing from contexts [8, 99].

4.1.3 Graphical context model

Graphical model mainly encompasses the models based on the Unified Modeling Language (UML), Entity-Relationship Model (ERM) and Object-Role Model (ORM) [51]. UML is a standardized general-purpose modeling language and is suitable to model contexts. ERM and ORM are powerful methods for designing and querying databases at the conceptual level. They are adopted in [54].

Some variants of ERM and ORM have been devised subsequently in representing contexts. For instance, Context Modeling Language (CML) springing from ORM was developed and enriched for conceptual modeling of databases [52, 53]. It extends ORM in capturing different classes and sources of contexts, and introducing quality metadata to express imperfect contexts.

In summary, graphical context model is capable of expressing contextual information by graphical diagrams. It also facilitates the analysis, design and discovering relationship of contexts at the concept level. However, they are ineffective in the practical interchange, which is attributable to a fact that they are lack of sufficient detail to facilitate reliable interchange of their notations between modeling tools and applications.

4.1.4 Object-oriented context model

This kind of context models, e.g., Cue [96] and Active Object [30] incorporates encapsulation, inheritance and reusability of object-oriented into context representation. In most cases, this model technique encapsulates the processing of contexts at the object level, and allows instances to access contexts by inheritance mechanism.

However, object-oriented context models make high demands of developers that they must systematically analyze and design the entire context taxonomy. In addition, object-oriented models are complex and may not be supported in hand-held device with limited computation capabilities.

4.1.5 Logic-based context model

Logic is the study of reasoning that aims at analyzing and representing facts from a set of other facts. It has attracted unceasingly attention from academia and industry in ubiquitous computing that is characterized by complex, dynamic and incomplete contexts. A variety of logic-based context models have been proposed. Pioneering efforts [46, 78] have applied logic to represent contexts.

A typical logic-based context model is introduced in Cabot project [20, 115, 116], which is designed for checking and resolving context inconsistency. Contexts are modeled by first-order logic as a five-tuple, i.e., $context = (category, fact, restriction, timestamp)$.

4.1.6 Ontology-based context models

Resource Description Framework (RDF) and Ontologies incorporate semantics into XML-based representation. They are supported by many powerful software tools such as Stanford Protégé [102] and IBM Integrated Ontology Development Toolkit [57]. Many RDF-based and ontology-based context models have been proposed [55, 107].

Ontologies and RDF are promising techniques to model contexts because of their formal expressiveness and the ability of inferring contexts. However, it is hard to construct complete ontologies and differ ambiguity of ontologies [89]. Moreover, they are also seriously limited by inexact reasoning.

4.1.7 Multidisciplinary context model

This model is proposed in [12] and refined in [13]. It is a hybrid model, capturing context relationship and interpretations from psychology, computer science, and linguistics. It encompasses the information concerning applications, users, and environment.

This model is a conceptual model that does not exactly specify how to represent contexts. Moreover, it is confronted with the inter-operability problem.

4.1.8 Domain-focused context model

This kind of context models is tailored to application domains, and thus considerably improves the functionalities of context-aware applications. This model represents contexts as a four-tuple (who, $what$, $where$, $when$), and supports query and deletion operations [18]. Thus, such representation is suitable for very expressive, and flexible data usages.

Modeling contexts by a universal context model still remains open due to the complex, dynamic and incomplete characteristics of contexts. Thus far, this survey has investigated how contexts are modeled in existing context modeling techniques. In the next section will investigate the advances in context acquisition. Note that we overview existing ubiquitous projects in terms of these functionalities of the proposed reference model shown in Fig. 5.

4.2 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 [27].

In recent years, owing to its low cost, wide availability and location sensing functionality, RFID has attracted increasing attention in ubiquitous agents [24], internet of things [124], ubiquitous e-healthcare [25]. It consists of three key elements—readers that are silicon-based radio transceivers, which interrogate and interact with tags by electromagnetic waves, tags that are tiny radio clients, which store unique identification, and systems that provide a way to user RFID data [25, 125]. Table 3 shows the component of RFID systems. RFID devices acquire node real-time location, thus being widely used in applications.

4.2.1 Context acquisition paradigms

In the sensor layer, there are two primary paradigms to gather contexts: event-driven and query-based. 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. In general, subscribers know neither where publishers are nor what they provide. But they can notify their interests to publishers mainly by web services or brokers or overlay network protocols, and merely receive messages that of their interests. Publishers or a pub service maintain a list of subscription so that they are able to deliver right message to right subscribers when an event is triggered (e.g., newly released message from publishers).

In contrast, the other paradigm for gathering contexts is query-based paradigm, which allows database-style query operations to be executed among sensing nodes and sink nodes through a declarative, SQL-like but a distributed interface [72]. When an application requires a certain kinds of context, it raise 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 [74], SensorWare [11] and CACQ [73].

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 [10].

Table 2 Some sensing devices and their capturing contexts

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	

*This table lists some common contexts captured by sensing devices

4.2.2 What kinds of contexts are acquired

Ubiquitous media applications usually involve contextual information about physical environment and users. This contextual information chiefly includes light and vision, audio, movement and acceleration, location and position, magnetic field and orientation, proximity, touch and user interaction, temperature, humidity and air pressure, weight and power, emotion. In order to capture contexts, various enabling techniques have been employed, such as GPS and Motes [25, 26, 29], shown in Table 2.

Note that enabling techniques, e.g., image recognition, machine learning and data mining, can capture virtual contexts that may not be aggregated by physical sensors . They may collect data loggers or images and then analyze user preferences, emotions and satisfactions [32, 80].

On the other side, we look into existing research on context acquisition, and get findings that most context acquisition efforts emphasize on acquiring location that is the most important contextual information. An example is load sensing [97], which is a mature and robust technology. Applications equip all objects' surfaces in a specific scenario with sensing capabilities, and thus they can accurately and reliably identify objects' positions (Table 3).

Table 3 Some sensing devices and their capturing contexts

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

4.2.3 How contexts are acquired

Data acquired by sensors is raw data and evolves into contexts when it is associated with interpretation or physical significance. For example, environmental contexts are connected with air and sound data, including air temperature, humidity, pressure, ambient noise and talking voice. Only all data is captured and interpreted, environmental contexts are aggregated.

Users or applications need to specify the relationships between sensor readings and contexts expected to be acquired. However, the types of contexts vary with applications, indicating that an application may require contexts that may be totally different from other applications. As a result, applications need to explicitly associate sensor readings and contexts, regardlessly of manually or automatically.

Figure 6 shows an example of the process of acquiring contexts on Muffin [118], which aims at detecting user behavior—walking or running or not. It firstly employs skin resistance sensors, accelerometer sensors, and ultra-range finder sensors to collect raw data. Then, it checks the attributes and values of these data to get low-level contexts, e.g., checking whether the value of the ultra-range finder sensor falls into the range of 0 and 100 to obtain the current location of the user. Finally, by incorporating other checking results and comprehensive analysis, it is able to aggregate the information about user current behavior.

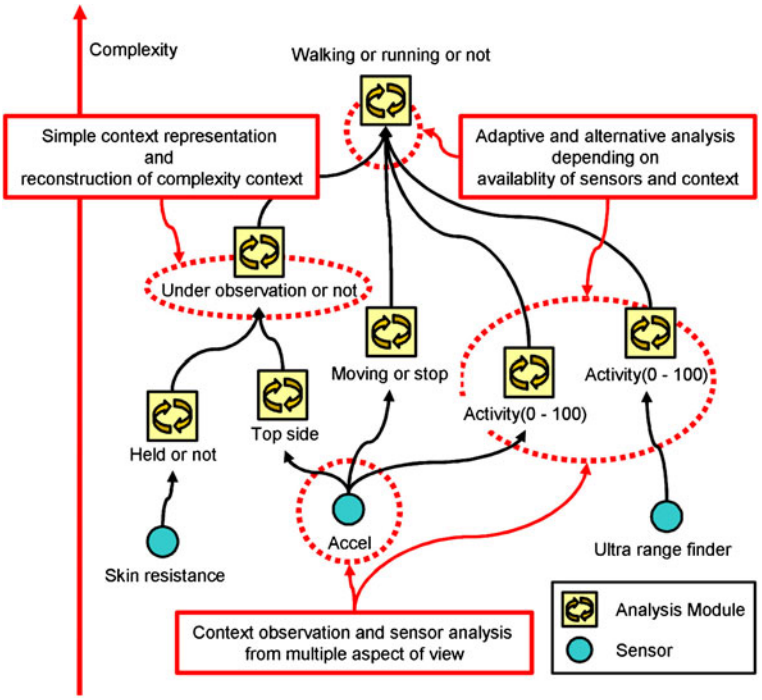


Fig. 6 An example of the process of context acquisition on Muffin [118], which detects user behavior by capturing sensor readings and corresponding interpretation

In summary, existing projects have conducted much research on acquiring contexts, and accumulated various experience in context acquisition. This experience indicates that contexts are mainly acquired from the sensor layer, and the most important context is location context. The proliferation of wireless communication, image recognition, machine learning, networks and software engineering has increasingly enabled existing techniques to capture more types of contexts. Projects ongoing and to be taken are supposed to leverage on more contextual information, and conduct comprehensive analysis of contexts to make ubiquitous media applications be further context-aware. Once contexts acquisition is finished, context-aware applications enter into the preprocessing step.

4.3 Context preprocessing

Contexts are often noisy and incomplete in ubiquitous media [61]. RFID readers capture 60–70% of the tags in their vicinity [24, 125]. Sensors deployed at the Intel Research Lab in Berkeley, on average delivered 42% of the data it was asked to report [59]. 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 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. We classify context processing techniques into two types: handling contextual information in the sensor layer or in the context layer. We omit the discussion of the first type that is not the focus of this paper.

4.3.1 Context preprocessing paradigms

Generally, there are three context preprocessing designs for contexts sensed by sensing devices—centralized, distributed and hybrid designs. In centralized designs, a central node or a centralized module is required to preprocess contexts captured and delivered by sensing contexts. In contrast, distributed paradigm allows sensors to filter and preprocess contextual information by aggregation nodes. 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 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. Thus, real-time applications usually adopt the event-driven paradigm. While 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. Meanwhile, 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.

4.3.2 Existing context preprocessing efforts

In the sensor layer, a variety of schemes have been proposed to preprocess raw contexts [41, 120]. 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 [42]. 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. Note that context inconsistency detection and resolution is exclude, which is discussed in the next section.

4.4 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. To remove and solve context inconsistency, various schemes have been studied in recent years. Figure 7 gives two example of context consistency conflicts. One is that two locations for a user at the same time is unreasonable, the other is that the speed change scope (less than 150%) is out of the scope that the system allows, where L_k is abnormal.

In this section, we firstly investigate context inconsistency detection schemes and then introduce corresponding inconsistency resolution schemes.

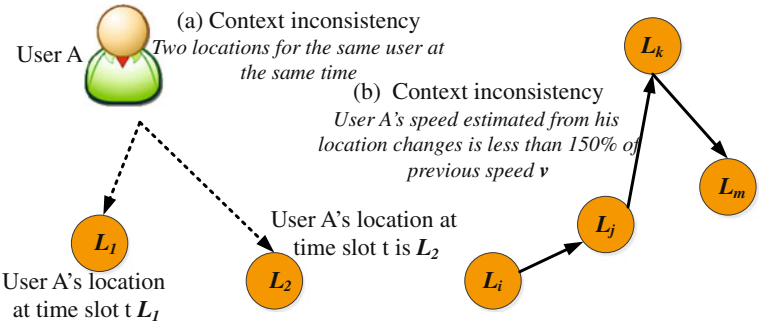


Fig. 7 Two examples of context inconsistency

4.4.1 Context inconsistency detection

The majority of existing context-aware efforts are concerned with frameworks that support context abstraction, reasoning and querying. Actually, 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 [45], CARISMA [17], CASF [94] and Context Toolkit [38] 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.
- Ontology-based context inconsistency checking schemes, which check context inconsistency by attributes, axioms and assertions [107]. 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, which check context inconsistency by tuple constraints. In [115], a context-aware middleware modeled context constraints by tuples shown in Section 4.1.5, and checked context consistency by semantic matching and inconsistency triggers among elements in tuples. This work is extended in [116], 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 [56] that checks context inconsistency by *happen-before* relationship among contexts in asynchronous environment.

4.4.2 Context inconsistency resolution

Regarding context inconsistency resolution, few schemes have been proposed. According to the manner that solves context conflicts, we classify them into user interference [86] and QoC-based (quality of context) categories [20, 77]

- User interference schemes, which 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, which 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. What follows in the context-aware reference model is context reasoning that is to be investigated in the next section.

4.5 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. While in a distributed manner, nodes cooperate and make reasoning, e.g., context reasoning based on peer-to-peer ubiquitous networks [48]. We summarize a taxonomy of context reasoning in Fig. 8.

4.5.1 Exact context reasoning

It consists of Bayesian network [75], case-based [63], logic-based [85], ontology-based [107], and rule-based [9] reasoning techniques. We investigate exact context reasoning by following the taxonomy.

- 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.
- 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

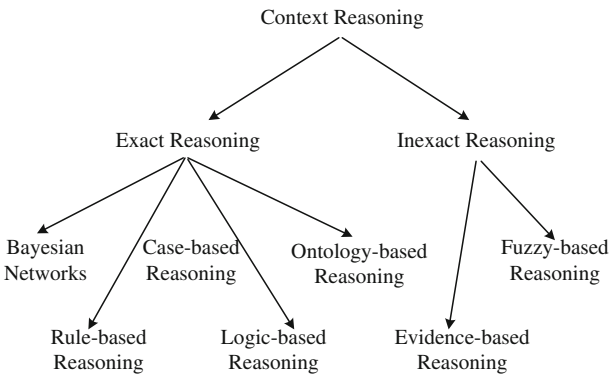


Fig. 8 The taxonomy of context reasoning

ubiquitous media. Moreover, most users resort to domain experts for ontology knowledge, which incurs extra human costs and hence restricts the application of ontology.

4.5.2 Inexact context reasoning

Fuzzy reasoning includes evidence-based [123] and fuzzy-based reasoning techniques [2, 33].

- Evidence theory, also known as Dempster-Shafer theory [34, 101], 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. Although [123] 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 [122]. This reasoning technique provides a manner of combining captured data with expert knowledge.

In summary, 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 about their effectiveness by probability community who prefer statistics, conventional probability and two-valued logic. In fact, inexact reasoning can be used as a way to inferring possible contexts in most applications.

4.6 Investigation of existing context-aware multimedia systems

Thus far, we have discussed most functionalities of context-aware services in the reference framework shown in Fig. 5. We aim to investigate how existing efforts achieve what degree of context-awareness. We selected Snap2Play [121], ConPhone [84], CAMP [3], CACH [82] and CMMM [31] as baselines.

Table 4 reports the comparison of selected context-aware media infrastructure. This implies that location is the most favorable context, and key-value context model is simple but yet powerful. This also shows that context reasoning and inconsistency functionalities are not available in most multimedia systems.

Table 4 Some sensing devices and their capturing contexts

Systems	Aggregation	Model	Preprocessing	Reasoning	Inconsistency
Snap2Play	Location	Key-value	Pilot test	X	X
ConPhone	Location	Key-value	X	Rule-based	X
CAMP	Scene	CDI/SMIL	Optimizer	X	X
CACH	Hospital environment ex. location, role	XML	X	Rule-based	X
CMMM	Location User preference	Key-value XML	MySQL	Rule-based	Yes

Table 5 Some sensing devices and their capturing contexts

Devices	Aggregation	Preprocessing	Communication
SmartTVs	ccd camera,microphone	Hand-gesture recognition	TV
	TV wireless network	Voice recognition	Wifi
sPhones	gps,microphone	Speech recognition	Wifi, 3G
	Camera	Hand-writing recognition	
Kinect	ccd camera, laser camera	Body-gesture recognition	Computer
	Microphone	Speech recognition	
Xtion	ccd camera, laser camera	Body-gesture recognition	
	Microphone	Speech recognition	
iPad2	Touch pad, camera, microphone bluetooth, Wifi	Proprietary software	Wifi, Internet
MyFi	Voice recognition	XM satellite radio station	
	Spatial information parser	Wireless FM transmitter	

Note that there are a number of context-aware multimedia systems that are customized for hand-held devices. We also investigated the pros. and cons. of context-awareness (particularly context aggregation, preprocessing and communication) in these types of applications. We extracted SmartTVs (http://www.gesturetek.com/newscenter/media.php?media=46), sPhones (http://www.apple.com/iphone/features/), Kinect (http://www.xbox.com/en-US/kinect), Xtion (http://us.estore.asus.com/), iPad2 (http://www.apple.com/ipad/), MyFi Radio (http://shopdelphi.com/consumers/support/faqs/myfi/).

The comparison result is given in Table 5, where context-awareness is supported in a lightweight manner. Owing to the limitation of communication and computation capabilities, the majority of hand-held devices do not provide the functionalities of context modeling, preprocessing, reasoning, and inconsistency processing.

5 Runtime support

Runtime support aims at building an execution environment for ubiquitous media applications by extending functions from underlying platforms in term of processing, communication and storage. It is indispensable to most ubiquitous devices and applications that are characterized by constrained resource, limited computation and communication capabilities.

Runtime support can be achieved in the sensor layer or the context layer. It is often implemented as a service or a virtual machine that embeds its functionalities into a middlewares-like module over a specific operating system.

- In the sensor layer, runtime support provides ubiquitous media applications with code interpretation, energy control, memory management, process handling, task scheduling, and thread concurrence and synchronization. Magnet [5] and LIME [81] are initial research on runtime support in the sensor layer.
- In contrast, runtime support for context-awareness is not yet fully discussed. Runtime support mainly involves some functionalities of handling and managing contexts in terms of device configuration, context storage, and assisting applications in adaption. Existing efforts have addressed many issues in context-awareness, but they do not provide adequate runtime support in the context

layer. For example, CARMEN [7] was an adaptive context discovery middleware that provided runtime support in discovering and acquiring contexts when contexts change. However, it is incapable of automatically configuring services in response to context changes without any intervention.

6 Challenging issues

Due to the complexity and high-level abstraction of context-awareness, many issues are still open and deserve further research, especially in the following aspects.

6.1 Programming abstraction

Programming abstraction substantially affects the development and application of context-awareness. It simplifies the programming effort and improves development efficiency, and provides a universal interface to interact with heterogeneous environment. It is dependent on a specific toolkit and a middleware.

- **Designing a programming abstraction for context-aware applications requires systematic design, rather than piece design**. As discussed in Section 3.2, programming abstraction consists of abstraction levels, interfaces and programming paradigms. Meanwhile, it is highly associated with context modeling, storage and underlying systems. We observe repeated activities from nearly sixty context-aware projects in almost all steps of developing context-aware media applications, i.e., from the context representation, acquisition to context exploitation. To our knowledge, only few ontologies and common libraries may be reused. Finally, media stream cannot be suddenly operated (e.g., back 5 min) [19]. This requires a new programming methodology to deal with.
- **Developing context-aware media applications need a bottom-up, decentralized approach**. Most existing context-aware systems are based on a top-down, centralized model where a central controlling entity facilitates and coordinates the functional operations of component devices. As technology advances, devices will be equipped with strong computation and communication capabilities. Therefore, the increasing capability of the component devices can be used to alleviate the role and load of the central controlling entity for achieving context-awareness.
- **Creating an infrastructure for rapidly building context-aware media applications necessitates a new manner to handling contexts.** Ubiquitous computing is characterized by transparent operation and reusability, which requires that applications should deal with contexts explicitly. Nevertheless, conventional schemes often hide context processing and thus cannot satisfy the requirement. Moreover, programming abstraction for context-aware applications is supposed to concentrate much on handling device heterogeneity, user mobility and asynchronous communication. To be specific, we enumerate some examples to reflect what should be paid great attention to: “what kind of abstraction is the best for context exchange among multi-organizational processes”, “how existing programming languages incorporate more aspects of contexts (e.g., annotation and metadata)”, “is it possible to set up a global share space for media service provisioning in

the cloud", and "when the message-passing programming paradigm rather than remote procedure call is used".

6.2 Context modeling

It is challenging to develop applicable and flexible ubiquitous media applications that covers various contexts in ubiquitous computing environment. Moreover, how to evaluate context models as good remains open. Ontologies have been recommended as the most expressive model based on six requirements in [64, 103]. However, these requirements are hard to evaluate, e.g., expressiveness and general rules. We share and extend the idea with [4] regarding context modeling.

- **Context model should at least contain context type and value information**. Meanwhile, it incorporates timestamp, source and confidence information of contexts as much as possible.
- **Context model may involve significant and compulsory parts**. These two parts play a significant role in reducing the complexity of removing context inconsistency, inference and resolution.

6.3 Context acquisition

The investigation in Section 4.2 shows that location is the most common context that are acquired by existing context-aware efforts. We consider two directions in context acquisition.

- **Acquiring news types of contexts that may enable applications to be more adaptive to changeable contexts**. Actually, there are a wide range of other types of contexts. Furthermore, exploiting new techniques that can capture more types of contexts to acquire contexts is a direction. These techniques keep evolving with the advance of communication, sensing and computation.
- **Gathering user's contexts from user behavior and communities is on the way.** This direction is not unique to context-awareness, and has been extensively researched in related disciplines (e.g., artificial intelligence, machine learning and software engineering). Given that sensors are easily error-prone, employing multiple sensors to monitor a specific object is attractive that may remarkably improve the accuracy of captured contexts. This also initiates non-trivial efforts on dynamically selecting sensors and intelligently fusing their readings.

6.4 Context reasoning

What will follow in context reasoning may fall into the change of manners in which it works, and the combination of exact and inexact reasoning techniques.

- **Large-scale context-aware applications (e.g., mobile transportation systems) require inferring contexts in a distributed manner**. Most context reasoning schemes rely on a central structure, which does not always hold in ubiquitous computing environment that is characterized by user movement and heterogeneous devices. Meanwhile, they are conducted on the limited scale of scenarios (e.g., smart rooms and labs).

- **Adaptive applications may benefit from the synthesis of exact and inexact reasoning**. Context-awareness includes exact, inexact or mixture requirements of context reasoning. To achieve adaptation, applications use or combine exact and inexact reasoning techniques correspondingly.

6.5 Context inconsistency detection and resolution

Owing to the noisy contexts, detecting and resolving context inconsistency is fundamental. It has drawn increasing attention in research community regarding two aspects.

- **Schemes for context inconsistency detection and resolution in a distributed manner are desirable for scenarios with user frequent movement and heterogeneous devices.** Most existing efforts work under an assumption that middleware or a central structure is available. As discussed in [20, 56], this assumption can not be always held, particularly in the peer-peer or urgent scenarios.
- **Incorporating semantics for checking context inconsistency and resolution may become a hot area**. The semantics of contexts refers to the information about specified objects. It helps ubiquitous media applications exactly understanding contexts and then make adaptation accordingly. Processing context semantics without ambiguity is challenging, but it is still a direction deserving to explore.

6.6 Runtime support

Section 5 shows that runtime support has been well-studied in the sensor layer and rarely discussed in the context layer. Given that many ubiquitous devices have limited communication and computation capabilities, runtime support should support context storage, task scheduling and process handling and concurrency.

Ubiquitous computing involves many nodes, which necessitates the efficient context exchange. Meanwhile, contexts keep evolving so that runtime should take context storage into consideration for rapidly context filtering and retrieving. Another possible direction is encapsulating runtime support into the cloud and hence users interact with systems with less involvement (e.g., no need to install and configure runtime support services). Note that we do not discuss the context security and privacy that is also a part of runtime support owing to its complexity and dynamics. For example, most security protocols fail in RFID devices because tags are seriously limited by communication and computation capabilities, and memory constraints. One way is to attaching access control to RFID-based systems [105].

7 Conclusion

Context-awareness in ubiquitous computing is an active research area, strongly driven by internal characteristics of ubiquitous computing. It makes ubiquitous media applications adaptive to the evolving contexts and to be intelligently tailored to user preferences.

In this paper, we have systematically investigated context-awareness in ubiquitous media. To identify key functionalities and services of context-awareness, we have proposed a reference framework from the systematic viewpoint—programming

abstraction, services and runtime support. We have further classified these into several sub-viewpoints. We extract functionalities from services as context modeling, preprocessing, reasoning, inconsistency and detection and resolution. Then, we have presented an overview of the state-of-the-art of existing context-awareness efforts by following the proposed reference model. Finally, we have identified possible challenging problems, and given detailed discussions on future research directions.

Acknowledgements We would like to thank Dr. Jingyu Zhou for his constructive suggestions. We also thank Xi Ma and Wenyin Wang for their proofread.

References

1. Abowd G, Atkeson C, Hong J, Long S, Kooper R, Pinkerton M (1997) Cyberguide: a mobile context aware tour guide. Wirel Netw 3(5):421–433
2. Anagnostopoulos CB, Pasias P, Hadjiefthymiades S (2007) A framework for imprecise context reasoning. International conference on pervasive services, pp 181–184
3. Asadi M, Dufourd J-C (2005) Context-aware semantic adaptation of multimedia presentations. IEEE international conference on multimedia and expo, pp 362–365
4. Baldauf M, Dustdar S, Rosenberg F (2007) A survey on context-aware systems. IJAHUC 2(4):263–277
5. Barr R et al (2002) On the need for system-level support for ad-hoc and sensor networks. Oper Syst Rev 36(2):15
6. Barton J, Kindberg T (2001) The cooltown user experience. HP Laboratories HPL, Tech Rep
7. Bellavista P, Corradi A, Montanari R, Stefanelli C (2003) Context-aware middleware for resource management in the wireless internet. IEEE Trans Softw Eng 29(12):1086–1099
8. Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, Riboni D (2010) A survey of context modelling and reasoning techniques. PMC 6(2):161–180. doi: 10.1016/j.pmcj.2009.06.002
9. Bikakis A, Patkos T, Antoniou G, Plexousakis D (2007) A survey of semantics-based approaches for context reasoning in ambient intelligence. In: Proceedings of the workshop artificial intelligence methods for ambient intelligence. Springer, London, pp 15–24
10. Bonnet P, Gehrke J, Seshadri P (2001) Towards sensor database systems. In: Proceedings of the 2nd international conference on mobile data management. Springer-Verlag, London, pp 3–14
11. Boulis A, Han C, Srivastava M (2003) Design and implementation of a framework for efficient and programmable sensor networks. In: Proceedings of the 1st international conference on mobile systems, applications and services. ACM, New York, pp 187–200
12. Bradley N, Dunlop M (2003) Towards a multidisciplinary user-centric design framework for context-aware applications. In: Proceedings of the 1st UK-UbiNet workshop. Springer-Verlag, London, pp 25–26
13. Bradley NA, Dunlop MD (2009) Toward a multidisciplinary model of context to support context-aware computing. Hum-Comput Interact 20(4):403–446
14. Brown P, Bovey J, Chen X (1997) Context-aware applications: from the laboratory to the marketplace. IEEE Pers Commun 4:58–64
15. Brumitt B, Meyers B, Krumm J, Kern A, Shafer SA (2000) Easyliving: technologies for intelligent environments. In: Proceedings of the 2nd international symp. on handheld and ubiquitous computing. Springer-Verlag, London, pp 12–29
16. Budzik J, Hammond KJ (2000) User interactions with everyday applications as context for just-in-time information access. In: Proceedings of the 5th international conference on intelligent user interfaces. ACM, New York, pp 44–51
17. Capra L, Emmerich W, Mascolo C (2003) Carisma: context-aware reflective middleware system for mobile applications. IEEE Trans Softw Eng 29(10):929–945
18. Castelli G, Rosi A, Mamei M, Zambonelli F (2007) A simple model and infrastructure for context-aware browsing of the world. In: Proceedings of the 5th IEEE international conference on pervasive computing and communications. IEEE, White Plains, pp 229–238
19. Chang S, Vetro A (2005) Video adaptation: concepts, technologies, and open issues. Proc IEEE 93(1):148–158

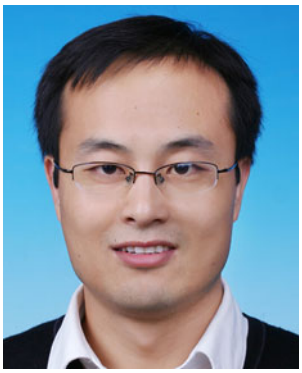
20. Chang X, Cheung SC, Chan WK, Chunyang Y (2008) Heuristics-based strategies for resolving context inconsistencies in pervasive computing applications. In: Proceedings of the 28th international conference on distributed computing systems. IEEE Computer Society, Washington, pp 713–721
21. Chen G, Kotz D (2000) A survey of context-aware mobile computing research. Department of Computer Science, Dartmouth College, Tech Rep
22. Chen M, Shu L (2009) Context-aware multimedia services in ambient intelligent environments. IEEE COMSOC MMTC E-Letter
23. Chen M, Leung VCM, Mao S, Yuan Y (2007) Directional geographical routing for real-time video communications in wireless sensor networks. Comput Commun 30:3368–3383
24. Chen M, Gonzalez S, Zhang Q, Leung V (2010) Code-centric RFID system based on software agent intelligence. IEEE Intell Syst 25(2):12–19
25. Chen M, Gonzalez S, Leung V, Zhang Q, Li M (2010) A 2G-RFID based e-healthcare system. IEEE Wirel Commun 17(1):37–43
26. Chen M, Huang R, Zhang Y, Chao H-C (2010) A smart RFID system. In: Proceedings of mobiWorld. IEEE, Beijing
27. Chen M, Gonzalez S, Vasilakos A, Cao H, Leung VCM (2011) Body area networks: a survey. ACM/Springer Mob Netw Appl 16(2):171–193
28. Chen M, Guizani M, Jo M (2011) Mobile multimedia sensor networks: architecture and routing. In: IEEE conference on computer communications workshops (INFOCOM WKSHPS), 2011. Shanghai, China, pp 409–412
29. Chen M, Guizani M, Jo M (2011) Mobile multimedia sensor networks: architecture and routing. In: Proceedings of MobiWorld. IEEE
30. Cheverst K, Mitchell K, Davies N (1999) Design of an object model for a context sensitive tourist guide. Comput Graph 23(6):883–891
31. Davidyuk O, Riekki J, Rautio V-M, Sun J (2004) Context-aware middleware for mobile multimedia applications. In: Proceedings of the 3rd international conference on mobile and ubiquitous multimedia, ser. MUM '04. ACM, New York, pp 213–220
32. Davis M, King S, Good N, Sarvas R (2004) From context to content: leveraging context to infer media metadata. In: Proceedings of the 12th annual ACM international conference on multimedia. ACM, New York, pp 188–195
33. Delir Haghighi P, Krishnaswamy S, Zaslavsky A, Gaber MM (2008) Reasoning about context in uncertain pervasive computing environments. In: Proceedings of the 3rd European conference on smart sensing and context. Springer-Verlag, Berlin, pp 112–125
34. Dempster AP (1968) A generalization of bayesian inference. J R Stat Soc Ser B 30:205–247
35. Dertouzos M (1999) The oxygen project. Sci Am 282(3):52–63
36. Dey AK (1998) Context-aware computing: the CyberDesk project. In: Proceedings of the AAAI spring symp. on intelligent environments. AAAI Press, Madison, pp 51–54
37. Dey AK (2001) Understanding and using context. PUC 5(1):4–7
38. Dey AK, Abowd GD, Salber D (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. HCI 16(2):97–166
39. Driver C, Clarke S (2004) Hermes: generic designs for mobile, context-aware trails-based applications. In: Proceedings of the workshop on context awareness at mobisys. ACM, Boston
40. Ducatel K, Bogdanowicz M, Scapolo F, Leijten J, Burgelman J (2001) Scenarios for ambient intelligence in 2010. The IST Advisory Group, Technical Report
41. Elnahrawy E, Nath B (2003) Cleaning and querying noisy sensors. In: Proceedings of the 2nd ACM international conference on wireless sensor networks and applications. ACM, New York, pp 78–87
42. Eo S-H, Zha W, You B-S, Lee D-W, Bae H-Y (2007) Intelligent context-awareness system using improved self-adaptive back propagation algorithm, vol 4761, ch 9. Springer, Berlin, pp 329–338
43. Finkelstein L, Gabrilovich E, Matias Y, Rivlin E, Solan Z, Wolfman G, Ruppin E (2001) Placing search in context: the concept revisited. In: Proceedings of the 10th international conference on World Wide Web. ACM, Hong Kong, pp 406–414
44. Fleury A, Pedersen J, Larsen L (2011) Evaluating ubiquitous media usability challenges: content transfer and channel switching delays. Design, user experience, and usability: theory, methods, tools and practice, pp 404–413
45. Garlan D, Siewiorek DP, Smailagic A, Steenkiste P (2002) Project Aura: toward distraction-free pervasive computing. IEEE Pervasive Computing 1(2):22–31
46. Ghidini C, Giunchiglia F (2001) Local models semantics, or contextual reasoning = locality + compatibility. Artif Intell 127(2):221–259

47. Grimm R, Davis J, Lemar E, Macbeth A, Swanson S, Anderson T, Bershad B, Borriello G, Gribble S, Wetherall D (2004) System support for pervasive applications. TOCS: ACM Trans Comput Syst 22(4):421–486
48. Gu T, Pung HK, Zhang D (2008) Peer-to-Peer context reasoning in pervasive computing environments. In: Proceedings of 6th annual IEEE international conference on pervasive computing and communications. IEEE, Hong Kong, pp 406–411
49. Gummadi R, Kothari N, Govindan R, Millstein T (2005) Kairos: a macro-programming system for wireless sensor networks. In: Proceedings of the 20th ACM symp. on operating systems principles. ACM, New York, pp 1–2
50. Gustavsen R (2002) Condor: an application framework for mobility-based context-aware applications. In: Proceedings of the workshop on concepts and models for ubiquitous computing held in conjunction with the 4th international conference on ubiquitous computing. Springer, Goteborg
51. Halpin T (2009) Object-role modeling, ser. encyclopedia of database systems. Springer, Berlin, ch 3, pp 1–6
52. Henricksen K, Indulska J (2005) Developing context-aware pervasive computing applications: models and approach. PMC 2(1):37–64
53. Henricksen K, Indulska J, Rakotonirainy A (2002) Modeling context information in pervasive computing systems. In: Proceedings of the 1st international conference on pervasive computing. Springer, London, pp 167–180
54. Henricksen K, Indulska J, McFadden T (2005) Modelling context information with orm. In: On the move to meaningful internet systems 2005: OTM workshops, vol 3762. Springer, Berlin, pp 626–635
55. Hu B, Hu B, Wan J, Dennis M, Chen H-H, Li L, Zhou Q (2009) Ontology-based ubiquitous monitoring and treatment against depression. Wirel Commun Mob Comput 999(9):22–38
56. 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
57. IBM China Research Lab (2007) Integrated ontology development toolkit. http://www.alphaworks.ibm.com/tech/semanticstk. Accessed 10 October 2011
58. Indulska J, Robinson R, Rakotonirainy A, Henricksen K (2003) Experiences in using CC/PP in context-aware systems. In: Proceedings of the 4th international conference on mobile data management. Springer, London, pp 247–261
59. Intel (2004) Intel lab data. http://berkeley.intel-research.net/labdata/. Accessed 10 October 2011
60. Jacucci C, Jacucci G, Wagner I, Psik T (2005) A manifesto for the performative development of ubiquitous media. In: Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility ser. CC '05. ACM, New York, pp 19–28
61. Jeffery S, Garofalakis M, Franklin M (2006) Adaptive cleaning for RFID data streams. In: Proceedings of the 32nd international conference on very large data bases. VLDB Endowment, Seoul, pp 163–174
62. Kagal L, Korolev V, Chen H, Joshi A, Finin T (2001) Project centaurus: a framework for indoor services mobile services. In: Proceedings of the 21st international conference on distributed computing systems workshops. IEEE, Phoenix, pp 195–201
63. Kofod Petersen, A, Mikalsen M (2005) Context: representation and reasoning: representing and reasoning about context in a mobile environment. Rev Intell Artif 19(3):479–498
64. Korpipää P, Mäntyjärvi J (2003) An ontology for mobile device sensor-based context awareness. In: Proceedings of the context, vol 2680, pp 451–459
65. Kumar M, Shirazi B, Das S, Sung B, Levine D, Singhal M (2003) Pico: a middleware framework for pervasive computing. IEEE Pervasive Computing 2(3):72–79
66. Lai R, Chen M (2011) Mobile multimedia technology for internet of things. E-Letter 6(9): 6743–6759
67. Lei Z, Georganas N (2001) Context-based media adaptation in pervasive computing. In: Canadian conference on electrical and computer engineering, 2001, vol 2. IEEE, Toronto, pp 913–918
68. Levis P, Culler D (2002) Maté: a tiny virtual machine for sensor networks. In: Proceedings of the 10th annual conference on architectural support for programming languages and operating systems. ACM, San Jose, pp 85–95
69. Levis P, Lee N, Welsh M, Culler D (2003) Tossim: accurate and scalable simulation of entire tinyos applications. In: Proceedings of the 1st international conference on embedded networked sensor systems. ACM, Los Angeles, pp 126–137

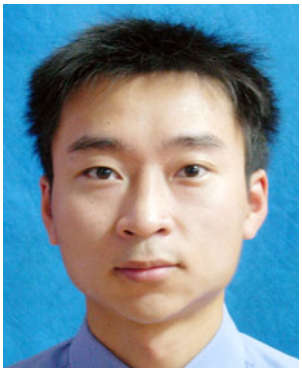
70. Levis P, Gay D, Culler D (2004) Bridging the gap: programming sensor networks with application specific virtual machines. In: Proceedings of the 6th symp. on operating systems design and implementation. ACM, San Francisco, pp 273–288
71. Liu T, Martonosi M (2003) Impala: a middleware system for managing autonomic, parallel sensor systems. In: Proceedings of the 9th ACM SIGPLAN symp. on principles and practice of parallel programming. ACM, New York, pp 107–118
72. Madden S, Franklin MJ, Hellerstein JM, Hong W (2002) Tag: a tiny aggregation service for ad-hoc sensor networks. In: Proceedings of the 5th symp. on operating systems design and implementation, vol 36. ACM, New York, pp 131–146
73. Madden S, Shah M, Hellerstein J, Raman V (2002) Continuously adaptive continuous queries over streams. In: Proceedings of the 2002 ACM SIGMOD international conference on management of data. ACM, New York, pp 49–60
74. Madden S, Franklin M, Hellerstein J, Hong W (2005) TinyDB: an acquisitional query processing system for sensor networks. ACM Trans Database Syst 30(1):122–173
75. Mamei M, Nagpal R (2007) Macro programming through bayesian networks: distributed inference and anomaly detection. In: Proceedings of the 5th annual IEEE international conference on pervasive computing and communications. IEEE, Washington, pp 87–96
76. Manuel R, Christopher H, Renato C, Anand R, Roy HC, Klara N (2002) A middleware infrastructure for active spaces. IEEE Pervasive Computing 1(4):74–83, 612880
77. Manzoor A, Truong H, Dustdar S (2009) Using quality of context to resolve conflicts in context-aware systems, ser. lecture notes in computer science, vol 5786/2009, ch 4. Springer, Berlin, pp 144–155
78. McCarthy J, Buvac S (1998) Formalizing context (expanded notes). CNL 81:13–50
79. Moran T, Dourish P (2001) Introduction to this special issue on context-aware computing. HCI: Hum-Comput Interact 16(2):87–95
80. Moran S, Nakata K (2009) The behavioural implications of ubiquitous monitoring. In: Proceedings of the IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology. IEEE, Washington, pp 327–330
81. Murphy AL, Picco GP, Roman G-C (2006) Lime: a coordination model and middleware supporting mobility of hosts and agents. ACM Trans Softw Eng Methodol 15(3):279–328
82. Muñoz MA, Rodríguez M, Favela J, Martinez-Garcia AI, González VM (2003) Context-aware mobile communication in hospitals. Computer 36:38–46
83. Paganelli F, Bianchi G, Giuli D (2007) A context model for context-aware system design towards the ambient intelligence vision: experiences in the etourism domain. In: Universal access in ambient intelligence environments, vol 4397. Springer-Verlag, Papadopouli, pp 173–191
84. Raento M, Oulasvirta A, Petit R, Toivonen H (2005) Contextphone: a prototyping platform for context-aware mobile applications. IEEE Pervasive Computing 4:51–59
85. Ranganathan A, Campbell RH (2003) An infrastructure for context-awareness based on first order logic. PUC 7(6):353–364
86. Ranganathan A, Campbell R, Ravi A, Mahajan A (2002) Conchat: a context-aware chat program. IEEE Pervasive Computing 1(3):51–57
87. Ranganathan A, Al-Muhtadi J, Campbell RH (2004) Reasoning about uncertain contexts in pervasive computing environments. IEEE Pervasive Computing 3(2):62–70
88. Rekimoto J, Ullmer B, Oba H (2001) Datatiles: a modular platform for mixed physical and graphical interactions. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, Seattle, pp 269–276
89. Rosemann M, Green P, Indulska M (2004) A reference methodology for conducting ontological analyses. In: Proceedings of 23rd international conference on conceptual modeling. Springer, Berlin, pp 110–121
90. Ryan N (1999) ConteXtML: exchanging contexual information between a mobile client and the fieldnote server. http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html
91. Ryan N, Pascoe J, Morse D (1999) Enhanced reality fieldwork: the context aware archaeological assistant. BAR International SERIES 750:269–274
92. Samulowitz M, Michahelles F, Linnhoff-Popien C (2001) Capeus: an architecture for context-aware selection and execution of services. In: Zieliński K, Geihs K, Laurentowski A (eds) New developments in distributed applications and interoperable systems. Kluwer, Deventer, pp 23–40
93. Satoh I (2006) Location-based services in ubiquitous computing environments. Int J Digit Libr 6(3):280–291

94. Satoh I (2009) A context-aware service framework for large-scale ambient computing environments. In: Proceedings of the international conference on pervasive services. Springer, London, pp 199–208
95. Satyanarayanan M (2001) Pervasive computing: vision and challenges. IEEE Pers Commun (see also IEEE Wirel Commun) 8(4):10–17
96. Schmidt A, Adoo KA, Takaluoma A, Tuomela U, Laerhoven KV, Velde WVD (1999) Advanced interaction in context. In: Proceedings of 1st international symp. on handheld and ubiquitous computing. Springer, London, pp 89–101
97. Schmidt A, Strohbach M, Van Laerhoven K, Friday A, Gellersen H, Kubach U (2008) Context acquisition based on load sensing. In: Proceedings of the 4th international conference on ubiquitous computing. ACM, Seoul, pp 333–350
98. Schwinger W, Ch G, Pröll B, Retschitzegger W, Schauerhuber A (2005) Context-awareness in mobile tourism guides: a comprehensive survey. Johannes Kepler University, Austria, Rapport Technique 1
99. Schilit B, Adams N, Want R et al (1994) Context-aware computing applications. In: Proceedings of the workshop on mobile computing systems and applications. ACM, Santa Cruz, pp 85–90
100. Selkar T, Burlesson W (2000) Context-aware design and interaction in computer systems. IBM Syst J 39(3–4):880–891
101. Shafer G (1976) A mathematical theory of evidence. Princeton University Press, Princeton
102. Stanford Center for Biomedical Informatics Research (2011) Protégé. http://protege.stanford.edu/. Accessed 10 October 2011
103. Strang T, Popien C (2004) A context modeling survey. In: Proceedings of the workshop on advanced context modelling, reasoning and management. In: Davies N, Mynatt ED, Siio I (eds) Lecture notes in computer science, vol 3205. Springer, Nottingham
104. Tadj C, Ngantchaha G (2006) Context handling in a pervasive computing system framework. In: Proceedings of the 3rd international conference on mobile technology, applications and systems. ACM, Bangkok, pp 13–18
105. Tang W, Chen M, Ni J, Yang X (2011) Security enhancement mechanism based on contextual authentication and role analysis for 2G-RFID systems. Sensors 11(7):6743–6759
106. Truong H, Dustdar S (2009) A survey on context-aware web service systems. IJWIS 5(1):5–31
107. Wang XH, Zhang DQ, Gu T, Pung HK (2004) Ontology based context modeling and reasoning using owl. In: Proceedings of the 2nd IEEE annual conference on pervasive computing and communications workshops. IEEE, Orlando, pp 18–22
108. Wang MM, Cao JN, Li J, Dasi SK (2008) Middleware for wireless sensor networks: a survey. J Comput Sci Technol 23(3):305–326
109. Want R, Hopper A, Falcao V, Gibbons J (1992) The active badge location system. TOIS: ACM Transactions on Information Systems 10(1):102
110. Weiser M (1991) The computer for the twenty-first century. Sci Am 265(3):94–104
111. Weiser M (1993) Some computer science issues in ubiquitous computing. Commun ACM 36(7):75–84
112. Welsh M, Mainland G (2004) Programming sensor networks using abstract regions. In: Proceedings of the 1st symp. networked systems design and implementation. USENIX Association, Berkeley, pp 3–3
113. WWW Consortium (2001) User agent profile (uaprof). http://www1.wapforum.org/tech/terms.asp?doc=WAP-248-UAProf-20011020-a.pdf. Accessed 10 October 2011
114. WWW Consortium (2007) Composite capability/preference profiles: structure and vocabularies 2.0—W3C working draft. http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/. Accessed 10 October 2011
115. 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
116. 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
117. Xu C, Cheung SC, Chan WK, Ye C (2010) Partial constraint checking for context consistency in pervasive computing. ACM Trans Softw Eng Methodol 19(3):1–61
118. Yamabe T, Takagi A, Nakajima T (2005) Citron: a context information acquisition framework for personal devices. In: Proceedings of the 11th IEEE international conference on embedded and real-time computing systems and applications. Hong Kong, pp 489–495

119. Ye J, Dobson S, Nixon P (2008) An overview of pervasive computing systems. Augmented Materials and Smart Objects: Building Ambient Intelligence Through Microsystems Technology 18(1):3–17
120. Yongzhen Z, Lei C, Wang XS, Jie L (2007) A weighted moving average-based approach for cleaning sensor data. In: Proceedings of the 27th international conference on distributed computing systems. IEEE, Toronto, pp 38–48
121. You Y, Chin TJ, Lim JH, Chevallet J-P, Coutrix C, Nigay L (2008) Deploying and evaluating a mixed reality mobile treasure hunt: snap2play. In: Proceedings of the 10th international conference on human computer interaction with mobile devices and services, ser. MobileHCI '08. ACM, New York, pp 335–338
122. Zadeh L (1973) Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans Syst Man Cybern 3(1):28–44
123. Zhang D, Guo M, Zhou J, Kang D, Cao J (2010) Context reasoning using extended evidence theory in pervasive computing environments. Future Gener Comput Syst 26(2):207–216
124. Zhang D, Yang LT, Huang H (2011) Searching in internet of things: vision and challenges. In: Proc. of the 9th IEEE international symposium on parallel and distributed processing with applications. Busan, Korea, pp 201–206
125. Zhang D, Zhou J, Guo M, Cao J, Li T (2011) Tasa: tag-free activity sensing using RFID tag arrays. IEEE Trans Parallel Distrib Syst 22:558–570



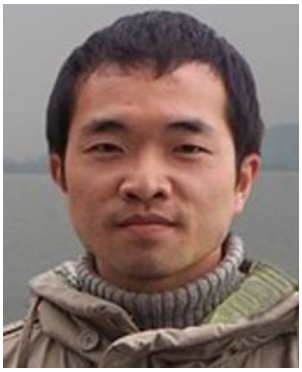
Daqiang Zhang received his Ph.D. degree in Computer Science at Shanghai Jiao Tong University in 2010. He is now with the college of Computer Science at Nanjing Normal University. His research focuses on ubiquitous computing, sensor networks and multimedia.



Hongyu Huang received his Ph.D. degree in Computer Science at Shanghai Jiao Tong University in 2009. He is now with the college of computer science of Chongqing University, Chongqing, China. His research interests include wireless sensor networks and vehicular ad hoc networks.

Chin-Feng Lai who born in Taiwan in 1980, is now an assistant professor at Institute of Computer Science and Information Engineering, National Ilan University since 2011. He received the Ph.D. degree in Department of Engineering Science from the National Cheng Kung University, Taiwan, in 2008. His research interests include multimedia communications, sensor-based healthcare and embedded systems. After receiving the Ph.D. degree, he has authored/co-authored over 60 refereed papers in journals, conference, and workshop proceedings about his research areas within two year. He was selected an honorary member of the Phi Tau Phi Scholastic Honor Society of the Republic of China in 2009 and TC member of Multimedia Systems & Applications Technical Committee (MSATC), IEEE Circuits and Systems Society, 2009. He is also a project leader of several international, industrially-funded multimedia projects. Now he is making efforts to publish his latest research in the IEEE Transactions on Multimedia and the IEEE Transactions on Circuit and System on Video Technology. He is also a member of the IEEE as well as IEEE Circuits and Systems Society and IEEE Communication Society.

Xuedong Liang is working as a Postdoctoral Research Fellow at Data Communications Group, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada. He received his PhD degree from University of Oslo, Oslo, Norway, in the area of Wireless Communications and Networking.



Qin Zou received his BE in information engineering in 2004, and MS in geographic information science in 2006, both from Wuhan University, China. From 2006 to 2007, he worked as a software engineer in Autodesk Lmt., Shanghai. After that, he joined Zoyon Image Processing Group as a research engineer. Currently, Qin Zou pursues his Ph.D. degree in photogrammetry and remote sensing (computer vision) at Wuhan University. From October 2010 to present, Qin Zou is a visiting student at Computer Vision Lab, University of South Carolina. His research activities involve perceptual grouping, 3D structure reconstruction, and shape matching, etc.

Minyi Guo received the B.Sc. and ME degrees in computer science from Nanjing University, China, in 1982 and 1986 respectively; and the PhD degree in computer science in 1998 from the University of Tsukuba, Japan. He is currently a distinguished professor and head of the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU), China. Before joined SJTU, Dr. Guo had been a professor and department chair of school of computer science and engineering, University of Aizu, Japan. Dr. Guo received the national science fund for distinguished young scholars from NSFC in 2007.

His present research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, and bioinformatics. He has more than 230 publications in major journals and international conferences in these areas, including the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Nanobioscience, the ACM Transactions on Autonomous and Adaptive Systems, the Journal of Parallel and Distributed Computing, INFOCOM, IPDPS, ICS, CASES, ICPP, WWW, PODC, etc. He received 5 best paper awards from international conferences. Dr. Guo is a senior member of IEEE, member of ACM, IEICE IPSJ, and CCF.