Survey of Context Provisioning Middleware

Michael Knappmeyer, Saad Liaquat Kiani, Eike Steffen Reetz, Nigel Baker, and Ralf Tönjes

***Abstract*—In the scope of ubiquitous computing, one of the key issues is the awareness of context, which includes diverse aspects of the user's situation including his activities, physical surroundings, location, emotions and social relations, device and network characteristics and their interaction with each other. This contextual knowledge is typically acquired from physical, virtual or logical sensors. To overcome problems of heterogeneity and hide complexity, a significant number of middleware approaches have been proposed for systematic and coherent access to manifold context parameters. These frameworks deal particularly with context representation, context management and reasoning, i.e. deriving abstract knowledge from raw sensor data. This article surveys not only related work in these three categories but also the required evaluation principles.**

***Index Terms*—Middleware, Context Provisioning, Context Management, Context Representation, Evaluation, Simulation, Ubiquitous Computing.**

I. Introduction

UBIQUITOUS Computing (UbiComp) paraphrases the paradigm of hardware and software components being transparently interwoven by means of wireless communication. Value added computer intelligence resulting from the smart and autonomous networking of multiple devices has much more potential than that originating from a single, isolated device. A key objective of these systems is to significantly simplify Human Computer Interaction (HCI) by deploying sensors, processors and actuators in the fabric of everyday life, such that their presence and complexity is hidden from users [1]. UbiComp is commonly understood as the next wave of an evolution chain of computing paradigms, which have gone through the personal computing (second generation) and distributed computing (third generation) from the roots of mainframe computing. Figure 1 illustrates our view of this evolution and highlights some of the features pertinent to the realisation of UbiComp.

A. Context and Context-awareness

Context is information about a location, its environmental attributes (e.g. noise level, light intensity, temperature, and motion) and the people, devices, objects and software agents that it contains. Context may also include system capabilities, services offered and sought, the activities and tasks in which people and computing entities are engaged, and their

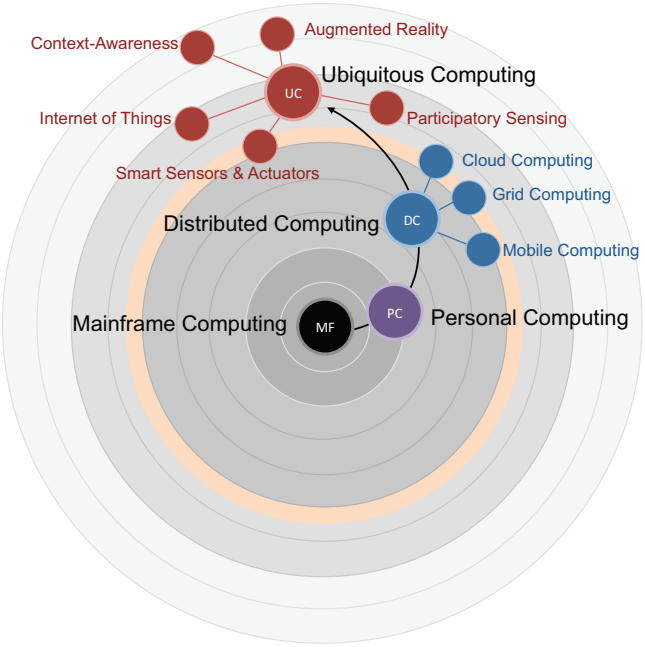

Fig. 1. Evolution Chain and Features of Ubiquitous Computing

situational roles, beliefs, and intentions. Context-awareness is one of the key enablers to facilitate proactive support of users in their current situation. Users do not have to define their situation explicitly by utilising time consuming and counter-intuitive input devices but it is implicitly recognised by the "smart" environment instead. The idea that computing devices can sense and react to stimuli from users' environment is labelled as context-aware computing.

User identity and location are the most prominent context parameters widely utilised in various location-based services. Beyond that, as defined by Dey [2], context may comprise *any* information relevant for describing the users' interaction with each other and with context-aware services and applications. In our digital world there is a large amount of distributed information available describing such interaction in a diverse way, e.g. the location of a person (GPS enabled smart phones), activity (phone-based gyroscope, accelerometer, foreground applications, digital calendar), social situation (location, time of the day, proximity to friends) and evolving preferences (self-configured and history-based digital profiles).

Context-awareness is an interdisciplinary field of research involving communication engineering and computer science, more precisely mobile communication, HCI (Human Computer Interaction) design, sensor data processing, feature extraction and artificial intelligence. In these communities a large set of application domains benefiting from contextual awareness has been proposed. Use cases of high potential

Manuscript received December 31, 2011; revised October 9, 2012.

M. Knappmeyer, E.S. Reetz and R. Tönjes are with the Faculty of Engineering and Computer Science, University of Applied Sciences Osnabrück, Germany. (e-mail: {m.knappmeyer, e.reetz, r.toenjes}@hs-osnabrueck.de).

S.L. Kiani and N. Baker are with the Faculty of Environment and Technology, University of the West of England, Bristol, UK (e-mail: saad2.liaquat@uwe.ac.uk).

Digital Object Identifier 10.1109/SURV.2013.010413.00207

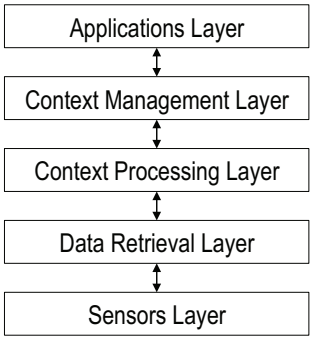

Fig. 2. Layered design of a context-aware middleware

include health care and well-being [3], e-learning and campus life [4], tourism and travelling [5], office and other business applications [6], advertising and e-commerce [7], entertainment [8], gaming [9] and social community applications [10]. Moreover, smart places – an emerging research field in itself – heavily rely on context-awareness. A smart place is a geographically bounded area providing smart interaction between computational devices and users located in the space. Smart offices, smart labs and smart homes are examples of smart places [11].

B. Context-aware Systems

The computing systems that are designed to provide context-aware services have to perform a variety of distinct functions. These include collection of raw data about the users and their environment, applying different reasoning techniques on such data to synthesise higher-level context information, storage of the context information in a retrievable and indexed format to make available when required, management and coordination of context and related information between different components of the systems, and to provide a platform for building, hosting or enabling context-aware applications and services.

A number of design approaches have been attempted for collective provisioning of these functions with the overall aim of making context information available about anything, any time, and anywhere. Context-aware systems are usually designed as middleware adopting a layered design – each functional layer hiding the details of the underlying layers (see Fig. 2). The primary benefit of this approach is the encapsulation of varying complexities of different functions. Each layer builds on the information made available by the layer below it, e.g. the Context Processing Layer uses data collected at the Data Acquisition Layer while the Applications Layer interacts with the Context Processing Layer to retrieve context and does not concern itself with the details of data acquisition or synthesis process.

With such a variety of functions to perform, the design, development, operation and evaluation of context-aware systems becomes a very complex task. Each functional layer in the middleware faces multidimensional challenges in contributing to the overall objective of context provisioning, e.g. with respect to data acquisition, to fully exploit awareness, context has to be acquired from heterogeneous sources, be processed and aggregated, as well as associated to users, devices and smart artefacts. Context is not only fetched from physical sensors but also from virtual and logical sensors, e.g. from databases or web services. This diversity and general applicability of context-awareness substantiate the need for architectural and functional support of ubiquitous services.

The design and operation of a context provisioning middleware is also laterally affected by the context model or representation scheme as it can influence the expressiveness and utility of the contextual information. In addition, current trends and innovations facilitate rapid development and deployment of new (smartphone) applications and services through the use of Service Creation Environments and specific Software Development Kits (SDK). Their context demands are unexpected and not easily predictable. The support of both existing context-aware applications/services and those emerging in the future requires functional scalability and gradual extendibility of the middleware with regard to new context types and new context processing capabilities. With their increasing processing, storage and communication capabilities, smartphones can be considered as personalised companions for interfacing with the digital world. Focussing on them as the main source of interaction allows for increasing the physical size of smart spaces to urban or even global extent. Therefore physical scalability is another prerequisite of a context provisioning middleware.

The complex operation of a context-aware system is illustrated as an autonomic communication cycle in Fig. 3. The cycle highlights the acquisition of raw data from sensors and user profiles, undergoing an aggregation/processing stage through application of various reasoning mechanisms, the resultant *information* providing a basis for decision making, which can support adaptation in context-based services and eventually serve as an input to the next cycle of (higher-level) context generation. In general, decision and adaptation are considered to be rather application/service specific and therefore tend to be realised by the actual service/application logic. The collection of sensor data and its analysis can be performed by a common middleware that is able to support a huge variety of application/service domains. Individual functions can be logically organised in different layers (cf. Fig. 2) of the context-aware middleware. The middleware as a whole thus acts a glue that binds these functions together to achieve the goal of context-provisioning. The role of a context-aware middleware is therefore both central and critical to the effectiveness of context-aware services in the realm of ubiquitous computing, without which the virtual and real worlds can not be bridged.

This article presents a retrospective view of the means by which existing context-aware systems carry out their functions using their constituent components for provision of contextual information and services. The comprehensive review and analyses provided in the following sections are focussed on how context-aware middleware systems undertake context modelling, management, provisioning and reasoning. Particularly, we analyse how such complex and interwoven systems, with the difficult task of bridging the virtual and real worlds, can be evaluated through multidisciplinary approaches.

C. Major Contributions

This article assembles the state of the art in the multifarious aspects of context-aware systems. The comprehensive review and analyses of these aspects contains significant contributions in the following categories:

The concept of context: Elaboration of the conceptual definition of context and context-awareness, including a view of how the definition of context has evolved over time, especially regarding human computer interaction.

Context modelling: A review of the context modelling and representation approaches, classifications of context models, the significance of context *meta data* and discussion on the features exposed by various context models that affect the utility and suitability of these models.

Context management: Identification of distinct functions carried out by context provisioning middleware and a review of existing classification of these systems. We present the basis of new classifications in terms of design, architecture and technology based parameters. A review of existing context provisioning middleware is presented, which serves as a basis for developing our novel three-tier classification based on the conceptual design model (layered, object-oriented, event-based, etc.) of a context provisioning middleware, the resulting architecture and implementation (central server, multiple-distributed servers, peer-to-peer, etc.) and an intersection of these two categories. Our classification presents a novel view for examining context provisioning middleware.

Context reasoning: We discuss the role of context reasoning as a critical function provided by context provisioning middleware, highlight the variety of multi-disciplinary techniques employed in context provisioning middleware and review the most prominent of these techniques. The context reasoning and processing approaches are categorised with respect to their requirements and features, and examples of their application domains are provided as well.

Evaluation of context middleware: A major contribution of this article is the detailed discussion on the evaluation strategies and mechanisms of context provisioning middleware. We identify the challenges faced in evaluating a multidisciplinary domain, review the approaches used in evaluation of contemporary context middleware as a whole and that of their individual functions and components, identify the recent trends and evolution of evaluation methodologies in these systems. A comparison of these evaluation methodologies, with respect to the target environment and requirements, is also presented along with examples from the state of the art.

Domain outlook: Finally, the domain of context provisioning is discussed in a broader perspective, highlighting the overlap and its interaction with other research disciplines. Based on lessons learnt and recent trends, we sketch the expected evolution and identify future research objectives.

Our objective is not only to present an in depth coverage of the approaches taken in fulfilling the critical functions of context provisioning middleware, but also to present a guide that comprehensively introduces the domain of context-aware systems to new researchers. To achieve this objective, it is essential to present the background and definitions of pertinent domain concepts, which we do so in Section II. The overall article structure is described in the following subsection.

D. Structure

We first of all elaborate the concepts of ubiquitous computing, context, and context-awareness as required background knowledge in Section II. Context modelling and representation are discussed from a classification perspective in Section III and the context management approaches adopted in existing systems are discussed and analysed in Section IV. Section V contains a discussion on the context reasoning and inference mechanisms. Section VI presents a detailed review and analysis on the evaluation techniques of these middleware systems. In this article decision taking is not covered, actuation and adaptation are only briefly addressed when discussing the evaluation (Section VI). The article finally concludes with a summary, presentation of future trends and discussion of open issues in Section VII.

II. Background and Definitions

A. Ubiquitous Computing

Ubiquitous Computing (UbiComp) evolved from mobile communication and relies on context-awareness as one of its key features (cf. Fig. 1). The term has been shaped by Mark Weiser [1]. Weiser's vision encompasses a new view and a paradigm shift with regard to the interaction of human beings with computational resources distributed pervasively across the environment and becoming part of the fabric of everyday life. Interaction includes both novel input mechanisms (e.g. gesture recognition based on accelerometers) and output devices (e.g. displays embedded in furniture). The use of processing resources is facilitated and refreshing as a "walk in the woods" [1]. Computers are pushed into the background. Ambient intelligence [13] arises from interweaving small embedded devices by means of mostly wireless communication. Overall, the HCI is simplified by autonomous adaptation to the users' situation and demands – and not vice versa. UbiComp systems aid in overcoming the problem of information overload.

When introduced, these UbiComp concepts appeared as a science fictional long-term target. However, due to the intermediate technological progress in increased processing, sensing, memory and communication capabilities, the vision has already become true to a certain extent. The size of handheld devices, environmental sensors and smart-its [14] has decreased continuously due to the advancements in solid-state technology. Numerous protocols and radio access technologies have been developed for supporting wireless communication in various ranges (e.g. Wi-Fi, IEEE 802.11; Zigbee, IEEE 802.15.4; Bluetooth, IEEE 802.15.1; 3GPP 3G/LTE). Today, smartphones, wireless sensors and networked desktop computers can easily be interwoven by always-on connectivity. However, there are a number of barriers to be overcome before truly context-aware applications and services can be deployed and made widely available.

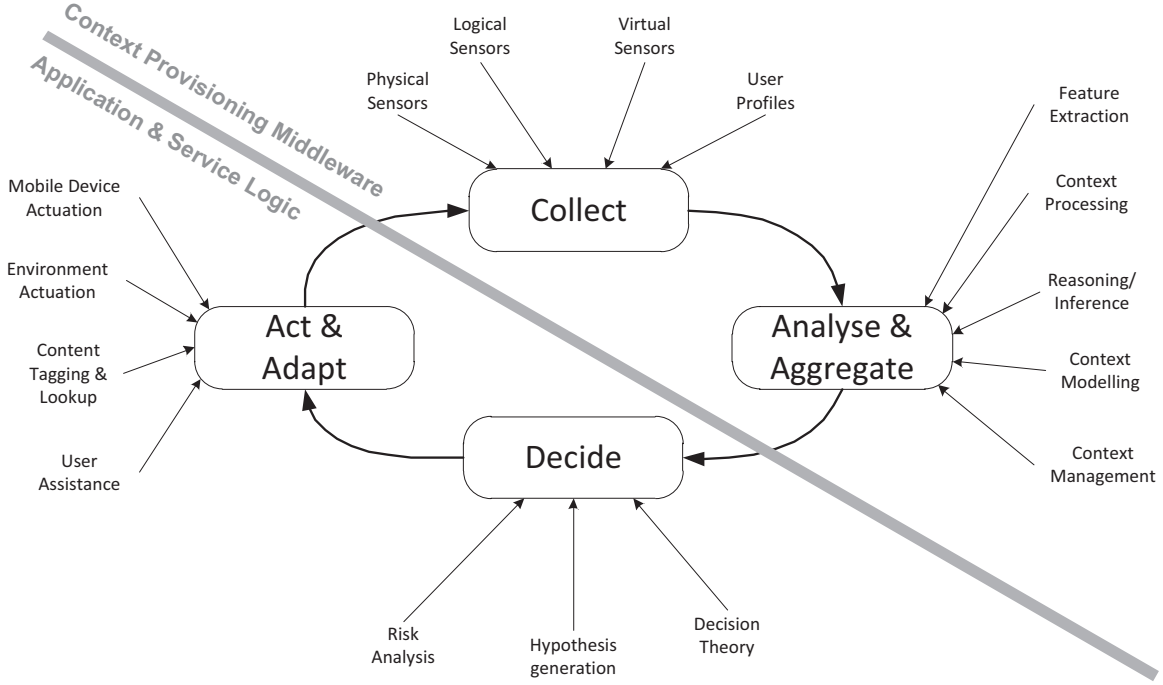

Fig. 3. The Context-Aware System Cycle (adapted from [12])

B. Context

According to Dey [2], context is any information that can be used to characterise the situation of an entity (person, place, physical or computational object) that is considered relevant to the interaction between entity and application [2]. This definition is by far the most cited in the literature. Zimmermann proposes five fundamental categories of context information: time, location, activity, relations and individuality [15].

Regarding the definition of a *situation* in context-aware systems, different views exist in the research community. Zimmermann defines it as "the state of a context at a certain point (or region) in space at a certain point (or interval) in time, identified by a name" [15]. Being a structured representation of a part of the context, it can be compared to a snapshot taken by a camera. Location and time can be used as spatio-temporal coordinates. Giunchiglia follows a more philosophical approach and sees context as a "subset of the complete state of an individual that is used for reasoning about a given goal" [16]. Situation is then "the complete state of the universe at an instant of time". With regard to situation-awareness, Billings [17] defines a situation as "an abstraction that exists within our minds, describing phenomena that we observe in humans performing work in a rich and usually dynamic environment". In summary, a situation may contain an infinite variety of contextual information.

Context can be classified into the following (incomplete) list of elements:

- *Spatial context*: information about the location, e.g. absolute geographic coordinates, relative physical proximity (distance), street, city, main business of places in proximity (e.g. shopping mall, university campus);
- *Temporal context*: information about the absolute time, relative time, day time (e.g. morning, afternoon, evening), weekend vs. business day, season;
- *Device context*: information about the users' interaction device(s), i.e. processing capabilities, input sensors, visualisation capabilities (e.g. supported codecs, screen size);
- *Network and communication context*: information about network characteristics, e.g. Wi-Fi access points in proximity, available bandwidth and throughput, supported Quality of Service (QoS) class, delay, transmission costs [18];
- *Environmental context*: information referring to the physical environment of an entity, e.g. noise level, air pressure, light intensity, pollution;
- *Individuality and user profile context*: information about the preferences, interests and habits associated to uniquely identified users;
- *Activity context*: information about what an entity does, which task it is currently involved in and what it intends to do next;
- *Mental context*: information about internal states of mind, e.g. a user's feelings and mood, level of stress;
- *Interaction context*: interaction may comprise both social interaction between several users and interaction between users and an application or service.

Interestingly, some of these context categories are directly or indirectly related to each other. For instance, considering the contextual illustrations from Fig. 4, the spatial context influences the temporal context since time of day directly depends on the current time zone. Various abstraction levels are also notable within most of the presented context types, e.g. some of the context elements associate information to any type of entity (user, device, room) whereas other elements are only applicable to selected sets of entities. Location, identity, time and activity are considered being the most important [19]

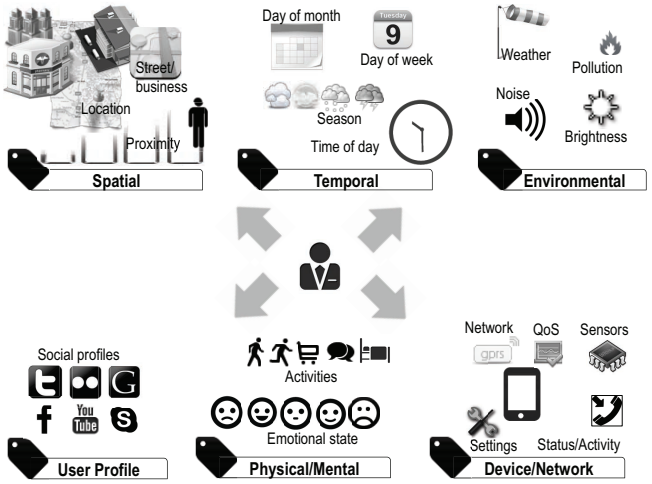

Fig. 4. Examples of various types of personal context

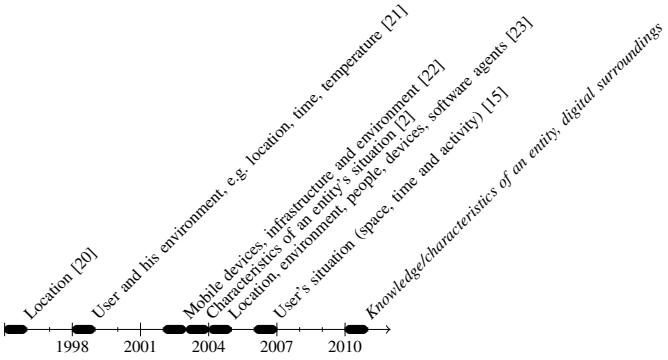

Fig. 5. A partial timeline of the evolution in the definition and understanding of 'context' (human-computer interaction) in literature.

in contemporary literature and context-aware systems. Fig. 5 illustrates a partial evolution map, according to published literature, of the definition and understanding of 'context' in human-computer interaction. In the scope of this article, we define context as follows:

Context is any information that provides knowledge and characteristics about an entity (a user, an application/service, a device, or a spatially bound smart place) which is relevant for the interaction between the entities themselves and with the digital world. Context can be categorised as being static, dynamic and rapidly changing.

C. Context-Awareness

The ability of a service, application or actuator to adapt to a specific context is referred to as context-awareness. For a piece of data to contribute to the adaptation ability of an application or service, it has to go through to complete lifecycle of acquisition, reasoning, representation, organisation, and then be communicated to the application or service to act upon it. These lifecycle stages are illustrated in Fig. 6. Given the classification of context types (cf. Section II-B), awareness can be categorised accordingly to define the specific focus, e.g. location-awareness, network-awareness. The term *context-awareness* was first introduced by Schilit and Theimer [20]. Typically, a context-aware system follows the autonomic com-

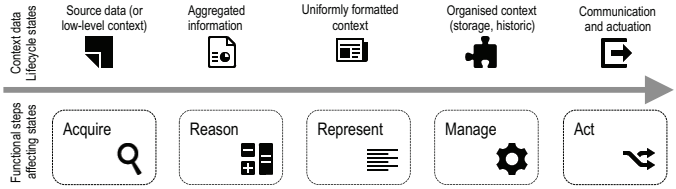

Fig. 6. Lifecycle stages that a context datum goes through, with respect to the functional processes in a context middleware.

munication feedback loop presented in Fig. 3 and comprises the following tasks [12], [15], [24]:

1) *Perception/Acquisition*, i.e. collection of relevant data allowing conclusion to the context;
2) *Reasoning/Inference*, i.e. deduce more meaningful information from the raw data;
3) *Learning* from historic context information and actions;
4) *Context Representation*, i.e. representation and modelling of context information in a machine interpretable way;
5) *Management and Diffusion* of context information;
6) *Actuation*, i.e. Triggering/adapting the service execution or application behaviour based on the available context information.

Moreover, three different categories of adaptation can be identified [19]:

1) presentation of information and services to a user, i.e. information is filtered and services are selected based on users' context;
2) automatic execution of a service and adaptation of an actuator, i.e. the service execution logic and actuation behaviour depend on users' context;
3) tagging of context to content (or information in general), i.e. context is associated to content in order to retrieve it more easily later on.

In order to take various levels or complexity into account the following definition of context-awareness is proposed.

The adaptive behaviour of services, applications and actuators according to the detected context depends on the degree of consciousness and context complexity. It can be subdivided into three different stages: ***Context-based adaptation*** *refers to applications querying context on-demand synchronously and changing the execution based on current context parameters.* ***Context-aware adaptation*** *requires an event-based asynchronous context diffusion in order to allow the execution logic to adapt on sensed and propagated events of interest.* ***Situation-aware adaptation*** *is more complex and requires (primitive) context to be further aggregated and high-level context to be inferred.*

The different stages of this adaptive behaviour can be explained with the following example: consider an application on a user's mobile device that can notify the user about nearby restaurants in a certain location. Such an application may poll the on-device GPS for coordinates and the time service for the time of the day. When certain conditions are met, e.g. time of day that can be considered a meal time and location is known, the application can query a location information service for nearby restaurants and notify the user. We term this behaviour

context-based because the application itself is not aware of the user's context, but is merely looking for information that can fire certain rules in its logic.

The next stage of *context-awareness* comes when such an application no longer actively engages in information polling but rather registers its interest in particular types of context information regarding an entity and is notified asynchronously when certain conditions are met. It becomes *aware* of the various context elements when they exist. While these types of applications are economical in their logic flow and execution, they do however depend on the availability of asynchronous event registration and notification in the middleware infrastructure. For such applications to be practically useful, there needs to be a higher level of contextual understanding, e.g. the user may not be in a situation to have lunch (driving, meeting, may already have had a meal).

A *situation-aware* adaptation scenario means that the application will base its recommendation (or lack thereof) on additional information which may include establishment of the user's current activities, historic context regarding meals taken at this time and place in the past, his social situation, list of commitments from his digital calendar, etc.

The review of existing applications that provide context related facilities, both research based prototypes and widely used smartphone-based applications, suggests that the current stage is predominantly context-aware adaptation.

III. Context Modelling & Representation

Context modelling is the process of designing a model of real world entities, their properties, state of their environment and situations that can be used as a reference for acquiring, interpreting and reasoning contextual information. The purpose of creating a context model for use in a context-aware middleware is to provide a uniform, machine processable context representation scheme, facilitate context sharing and inter-operability between different middleware layers and external applications. The uniformity of the model asserts a common understanding between all software components and applications in the middleware. This common understanding is vital because the acquisition, reasoning and utilisation of context information is usually done by separate components, layers or applications in the middleware. Moreover, additional demands of a context model are raised due to the temporal nature of context, variable certainty of sensed information in the environment and about users, the fluid nature of relationships between entities as well as the availability of new types of knowledge about the environment.

Context modelling techniques and requirements have been surveyed by Strang and Linnhoff-Popien [25] and Bettini et al. [26]. Context information needs to be represented and modelled for being machine interpretable and exchangeable using well-defined interfaces. The goals are to support easy manipulation (low overhead in keeping the model up-to-date), easy extension (cheap and simple mechanism for adding new types of information), efficient search and query access as well as scalability. A number of different approaches for representing contextual knowledge and semantic information can be found in literature. On the one hand the representation is tightly coupled to the inference mechanism, e.g. probabilistic logic requires the modelling of probabilities. On the other hand the representation is often tailored to the problem domain and to the specific goal of the system.

Strang and Linnhoff-Popien [25] identify generic requirements: The modelling approach should (1) be able to cope with high dynamics and distributed processing and composition, (2) allow for partial validation independently of complex interrelationships, (3) enable rich expressiveness and formalism for a shared understanding, (4) indicate richness and quality of information (QoI), (5) must not assume completeness and unambiguousness, and (6) be applicable to existing infrastructures and frameworks.

A. Classification of Context Models

Bolcini et al. [27] provide a framework for the analysis of context models that caters for modelled aspects, representation features, usage and context management features. They analyse existing context modelling efforts and, based on their level of fulfilment of analysis parameters, they are classified into five 'classes of use' that include *Context as a matter of* (1) channel-device presentation, (2) location and environment, (3) user activity, (4) agreement and sharing and (5) data/service selection.

Moreover, context models can be classified into six different model categories, namely key-value models, markup scheme models, graphical models, object oriented models, logic based models and ontology based models [28]. Recently, chemistry inspired models have been proposed [29] as well. Table I provides an overview of the individual advantages and disadvantages of different context modelling schemes, which are further discussed in the following paragrahs.

Key-value pairs form a simple tuple of information. The context information is assigned to a unique key in order to allow for easy lookup by applying a matching algorithm. These pairs are easy to manage but lack structuring and therefore do not allow for efficient context retrieval. *Markup scheme models* incorporate a hierarchical data structure of markup tags, attributes and content. Examples include the User Agent Profile and the Composite Capabilities/Preference Profile (CC/PP) [31], which are based on XML (Extendible Markup Language) and standardised by the World Wide Web Consortium (W3C). The Context Meta Language (ContextML) [30] is another markup based scheme that not only represents the context information but context metadata as well.

Graphical models (e.g. based on the Unified Modelling Language) allow for a picturesque description of a context model [36] and for deriving an Entity Relationship model as required in rational databases. An extension is proposed by Henricksen and Indulska [32], introducing Object-Role Modelling (ORM). Object oriented models offer powerful capabilities of inheritance, reusability and encapsulation. Access of contextual information is provided by well defined interfaces [37]. *Logic based models* offer a high degree of formalism and typically comprise facts, expressions and rules. They enable formal inference, e.g. by means of general probabilistic logic, description logic, functional logic or first-order predicate logic.

Ontological modelling refers to an abstract conceptual vision of the world. The relations within could also be de-

TABLE I
CONTEXT MODELLING AND REPRESENTATION APPROACHES

Context Modelling Approach	*Advantages*	*Disadvantages*	*High Dynamics* *	*Distributed Processing* *	*Formalism* *	*QoI Indication* *	*Incompleteness* *	*Examples*
Key-Value	Fast processing, storage and lookup	Lack of data structure	✓	/	/	/	/	
Markup Scheme	Structured representation; fast reasoning; schemes allow for a common definition and knowledge of available parameters	Data processing slows down with increasing size of the scheme	(✓)	✓	(✓)	(✓)	/	ContextML [30], CC/PP [31]
Graphical	Readable by machines and humans; clear relations between model components	Implementation of graphical models typically requires a technology model (e.g. object oriented design) for implementation		(✓)	(✓)	(✓)	/	ORM [32]
Object Oriented	Support of coalgebraic and abstract data types, recursive types, encapsulated states, inheritance	Data exchange in a distributed middleware requires an object oriented communication protocol	(✓)	/	(✓)	✓	/	JCAF [33]
Logic based	Fast processing, high degree of formalism	Typically restricted to the targeted reasoning mechanism (cf. Section V); possible need of model conversion		✓	✓		/	
Ontology based	Implicit reasoning mechanisms; coherent and unambiguous knowledge representation	A large ontology slows down the reasoning; domain experts and ontology engineering are required	/	(✓)	✓	/	/	Wang et al. [34], CoBrA [23], SOCAM [35]
Chemistry inspired	Ability to fuse environment and user context to trigger services autonomously; similarity to chemical bonding and chemical reactions	Complex model requiring event-based context propagation	(✓)	✓	(✓)	/	/	Ikram et al. [29]

* ✓= support; (✓) = limited support; / = no support. Empty fields indicate a significant dependency on the chosen implementation.

scribed by object oriented methods. However, an ontology is commonly described by using languages standardised by the W3C in the context of the semantic web. Most relevant are the Resource Description Framework Schema (RDF-S) [38] and the Web Ontology Language (OWL) [39].

Korpipää and Mäntyjärvi [40] enumerate the following goals for designing a context ontology: simplicity, flexibility, extensibility, generality and expressiveness. Many researchers have come to the conclusion that ontologies are theoretically the best way to represent and model context due to the extendibility and unambiguousness [28], [34]. However, there may be certain drawbacks as ontology engineering is a challenging and interminable matter. With the size of the ontology, querying and processing the information embedded within becomes slow, in particular if performed on resource constrained mobile devices. The context model can be arranged in layers to cushion this effect. Wang et al. [34] propose ontology modularisation, i.e. a generic upper ontology on top and domain specific ontologies below. Full featured ontological representations tend to decrease the inference performance and are not suited for highly dynamic systems. Especially if resource constrained mobile devices are envisaged as the main source and consumer of context an appropriate alternative must be chosen. Another argument for not applying ontological representation is its limited support for modelling uncertain and unavailable data.

B. Context Meta Data

Beside the context information itself, a model for representing related *meta data* is crucial. Such context meta information

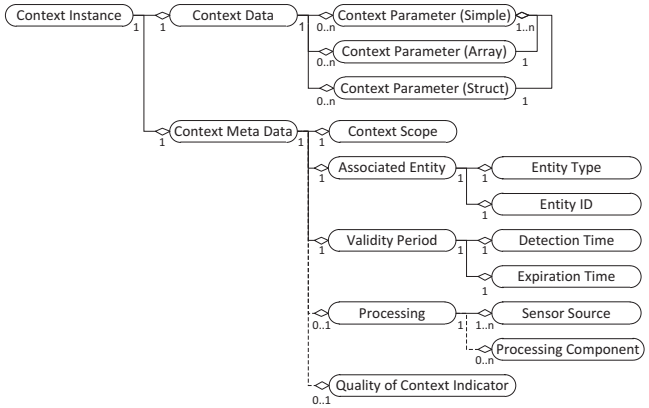

Fig. 7. Context meta data in the ContextML schema [30])

may include a quality of information quantifier, the degree of uncertainty, possibility, measurement accuracy, resolution or confidence interval. As an example, Fig. 7 depicts the context meta data elements of the ContextML schema [30].

These attributes are especially helpful for certain inference and reasoning mechanisms (cf. Section V). When taking historic context into account, it is important to embed data related to the temporal domain such as time of detection and expiry time. Rapidly changing information can be differentiated from rather static information (such as gender, year of birth). In addition, if tracing back context inference processes is required, both context sources and context processing entities may be captured and stored as meta data. Crucial – but out of scope of this article – is privacy and security information [41]. In their

work, Chang et al. [42] propose the modelling of a lifecycle for contextual information and an appropriate representation in meta data. The state of contextual information (e.g. ready, running, expired, suspended) enables flexible and fast transitions when the context changes temporarily. However, other models such as ContextML only utilises two states of context information; transitions only occur unidirectionally and ultimately from *valid* to *historic* context. The following definition clarifies this differentiation.

Context meta data *refers to information explicitly attached to the context data. Meta data may be subdivided into mandatory and optional parameters and may comprise the detection time, validity period, source of information, quality of context, probability or uncertainty, associated entity, applied processing, etc.*

A ***context instance*** *refers to a specific instantiated object of contextual information, including meta data. Due to limited validity of context, each context instance only remains valid for a specific period of time. Any expired context instance is considered to be* ***historic context****.*

IV. Context Management & Provisioning

The management and provisioning of context information are essential elements for realising context-aware services and applications. A notable number of context management approaches have been presented; surveys of which have been published for instance in [28], [43], [44].

A. Functionalities

The overall management is usually subdivided into several functional tasks as illustrated in the core layer of Fig. 8. The access to contextual information is typically aided by a context management system, toolbox, framework or middleware enabling services and applications to acquire and utilise context.

Sensor Data Acquisition deals with how raw information about any context is fetched and used as input to the middleware. It is important that the system can cope with a variety of heterogeneous sources and sensors simultaneously. Sensors may be physical, virtual, or logical in nature. Depending on the intelligence and computational power, preprocessing and filtering may be performed by the sensor nodes themselves or as part of the middleware functionality. Both synchronous and asynchronous sources are generally supported. The benefit of *Context Storage* is twofold. Caching strategies allow for faster provisioning of the required context since repeated processing stages may be omitted. Moreover, the storage of expired context in a history database enables the analysis of previous situations. Such information can be used to determine habits and long-term intentions taking successive sequences of activities towards the desired goal into account.

Context Lookup & Discovery provide means for an application, service or actuator to identify the available context and how to acquire and query for it. Commonly used approaches include lookup tables, semantic queries or web service mechanisms such as SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language). *Context Diffusion & Distribution* are related to the output of a middleware system, i.e. how context information is made available to the consumers. This encompasses not only the definition of query models (e.g. key-value based or SQL based) but also the mode of communication. Communication protocols may support event-driven asynchronous publish/subscribe mechanisms to notify the application layer about context changes of interest. Additionally, synchronous on-demand queries may be supported by the middleware. Since context reveals private information about users, *Privacy, Security and Access Control* are crucial tasks in every middleware. However, due to the selected scientific focus, they are not considered in-depth in this article.

Context Processing & Reasoning refer to the capability of inferring context from raw sensor data or from existing primitive low-level context. The middleware may apply feature extraction, description logic, rule-based reasoning or probabilistic inference on behalf of the application layer, hence saving battery consumption on mobile resource constrained devices. A powerful middleware has to support modularity so that numerous processing mechanisms and algorithms can be plugged in. Details of processing and reasoning techniques employed in contemporary systems are discussed later in Section V.

Functional, spatial, synchronisation and temporal decoupling of these elementary tasks is targeted in most of the architectural principles as further discussed below. Functional decoupling is achieved whenever functionalities can be invoked independently from each other. Spatial decoupling foresees a sender being unaware of the receivers' address or presence. Synchronisation decoupling prevents blocking in the sender and receiver components when exchanging context so that their main flow of execution can continue to be carried out. Temporal decoupling requires senders and receivers of context not to be involved at the same time.

B. Existing Classifications

Hong et al. [44] classify context middleware approaches into the following six categories: (1) agent-based, (2) reflective, (3) metadata based, (4) tuple space based, (5) adaptive and objective based and (6) Open Services Gateway Initiative (OSGI) based. However, this grouping mixes up architectural approaches, context representation and applied technologies. Baldauf et al. [28] differentiate between (1) direct sensor access, (2) middleware infrastructure and (3) context server as architectural models. Hence architectural design is focused without going into detail about technological details. Winograd [45] proposes a distinction between (1) widgets providing interfaces for hardware sensors, (2) networked services resembling the context server architecture and a (3) blackboard model representing a data-centric view.

Regarding the overall system design, Hong et al. [44] recommend four layers (bottom-up): (1) Network Infrastructure Layer, (2) Middleware Layer, (3) Application Layer and (4) User Infrastructure Layer. Baldauf et al. [28] identify five layers focussing on functional stages: (1) Sensors, (2) Raw data retrieval, (3) Preprocessing, (4) Storage/Management and (5) Application. Zimmermann [15] derives a layered framework comprising (1) Sensor Layer, (2) Semantic Layer, (3) Control Layer and (4) Actuation Layer where only the first

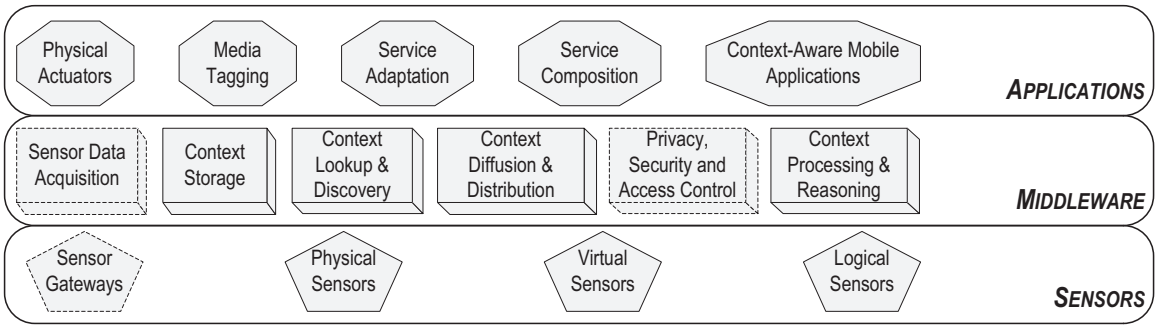

Fig. 8. Middleware Functionalities

two layers are directly related to context management and provisioning and the latter two deal with decision logic and service adaptation.

Makris et al. [46] identified six categories of abstract context aware functionalities. These functionalities are (1) context acquisition, (2) context modelling, (3) context exchange, (4) context evaluation, (5) exploitation of context from business logic perspective and (6) context aware horizontal functionality dealing with security, privacy and trust issues.

Based on these existing classifications, the following subsections discuss concrete design approaches, categories of architectural models and relevant implementations. We present an overview of selected context management frameworks and middleware solutions, classify them according to our proposed characteristics and furthermore review their successive evolution.

C. Design-based Classification of Context Provisioning Middleware

As discussed in the previous paragraphs, a number of design approaches have been attempted for the collective provisioning of context-awareness related functions. These design approaches range from layered middleware designs to object-oriented and event-based middleware designs. Different communication and coordination semantics between individual components of the logical design result in varied *architectures* of context-aware systems as well. The layered middleware design is prominent, where each functional layer hides the details of the underlying layers. The primary benefit of this approach is the encapsulation of varying complexities of different functions. Each layer builds on the information available from the layer below it, e.g. a context processing layer uses data collected at the data acquisition layer while the application platform layer interacts with the context processing layer to retrieve context and does not concern itself with the details of data acquisition or synthesis process. Within this design approach, the provision of functions is often separated into different architectural components, e.g. some functions are provided by a central server while applications that use these functions are deployed remotely. *SOCAM* [35] is a prominent example of a layered middleware design.

Some examples of context-provisioning middleware adopt an object oriented model, where functions of context-awareness are assigned to separate software objects. These objects encapsulate the internal processing of the functions and provide interfaces for interaction with other objects. The primary difference between the layered middleware and the object-oriented design is that in the layered design the context processing pipeline is vertical and operates in series amongst the various layers, whereas an object-oriented approach allows different functions to interact 'out of series' and horizontally. This feature is demonstrated in CORTEX [47], which showcases the concept of *sentient* objects enabling context-awareness in ad-hoc environments by exchanging context amongst themselves. Object oriented design also provides developers with the benefit of extensibility through polymorphism and inheritance, as demonstrated in *JCAF* [33], which exploits these benefits to provide an infrastructure and a programming API for developing and deploying context-aware applications and services.

Event based middleware design is conceptually closest to the nature of context processing because human context and that of related computing entities is naturally reactive and fluctuates with stimuli. Context-aware middleware designed with this approach operate on the production, detection, consumption of, and reaction to events. An event in such a system is any change in state of any user or device context. System components are usually divided into context *producers* that generate context related events and context *consumers* that react to contextual events. An event service is usually employed to process and route events between consumers and producers and may also perform filtering and transformation on context events in complex context-aware systems. Event consumers register their interest in particular events either directly with the event producers or with the event service. Publish/subscribe and message notification based communication semantics are used in the middleware developed using this design approach. Event based design affords better responsiveness in context-aware systems as they are by design more normalised to unpredictable and asynchronous environments.

Different middleware designs exist due to the different approaches used by designers to realise the functions of context-awareness (acquisition, reasoning, communication etc.). We have classified some of the contemporary middleware according to their design philosophy (cf. Table II). The *realisation* of a design approach is carried out by specifying functional responsibilities to architectural components of a software system. While most of the existing middleware can be categorised

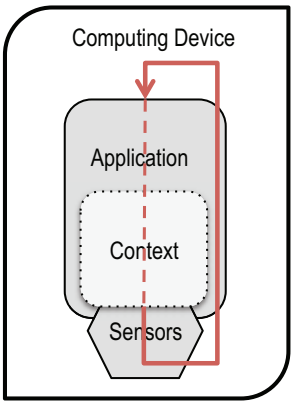

Fig. 9. Direct sensor coupled application running on a single computing device

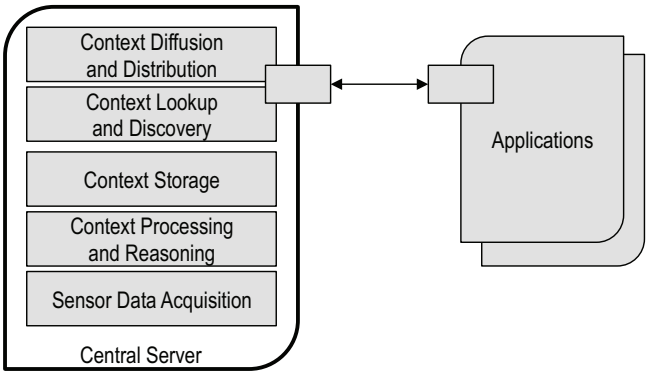

Fig. 10. Central server based middleware architecture

under one of the listed design categories, there is a further categorisation that can be elicited by examining these systems from a deployment and component architecture perspective. This *architectural* categorisation, described in the following paragraphs, is primarily based on how the components that carry out different context-awareness functions in the middleware interact amongst each other to achieve the overall goal of context provisioning.

D. Architectural Classification of Context Middleware

In some middleware systems, context related applications operate by directly accessing the sensors states on which the context information is based (*direct sensor coupling*). Generally, in such middleware a context-aware application directly consumes information retrieved from sensors and there are no dedicated context reasoning, communication or coordination functions. As illustrated in Fig. 9, the role of the middleware is limited to that of a hardware abstraction layer, with no other specialised functions available.

In *context server based architectures*, a central server performs the functions of collecting and synthesising context from sensor data and other information sources. As illustrated in Fig. 10, clients of the server access it remotely to retrieve context or raw data to process locally. Therefore, the context server performs most of the middleware functions. This is one of the most common architectures used in context-aware systems. One of the reasons for the prevalence of this architecture is the functional flexibility it affords to designers. For example, the central server may limit its functions to mere data acquisition or it may provide context synthesis and application platform on top of it.

In a *peer-to-peer architecture* the functional tasks of context-awareness are carried out by peer components, each peer acting as both a server and a client. It is a distributed design approach where peers either individually carry out mutually exclusive subsets of context-awareness related functions or replicate the functionality in different geographical or logical domains. These peer components usually utilise a centralised server for coordination of context with context consuming applications e.g. Hydrogen [37] and Context Toolkit [48]. Server-less peer-to-peer architectures also exist (*JCAF* [33], CORTEX [47]) but may suffer from limitations due to the lack of a coordinating component e.g. the requirement of hard-coding addresses of communication endpoints between context services and clients in *JCAF*.

Different approaches to designing context-aware systems exist due to the constraints imposed by a variety of factors. Leading factors effecting the design decision and resultant architecture include location of sensors, number and mobility patters of users, available resources, targeted scale of the system and methods of context acquisition and distribution. Direct sensor coupled architecture has been used when sensors and context consuming applications are accessible within a single system (e.g. a mobile device or a desktop computer) and there is no pressing requirement for complex context reasoning. However, due to this tight coupling, the scale, scope and usability of these middleware to be exploited as holistic context-aware systems is severely restricted. Most sensors have limited communication range and software components that access data from such sensors have to be located physically close to the sensors. These limitations triggered the evolution towards a context server approach where a central server acquires, processes and stores context while providing interfaces for local and remote applications to access context. Context server architecture allows reuse of sensor data and relieves resource constrained devices from context acquisition and processing. While leveraging these benefits, context server architecture requires consideration of new factors in the design that include selection of appropriate network protocols, quality of service parameters, network performance, mobility management etc.

The basic central server architecture allows remote components to access context but the data acquisition function is still restricted by the limitation in communication range of sensors and other information sources. To overcome this shortcoming, the central server architecture has evolved further in terms of distribution of its constituent components, e.g. distribution of the data acquisition function into multiple remote data acquisition modules that acquire data from their assigned sources and push it to the central server. This further distribution of functionality, illustrated in Fig. 11, has transformed context-aware middleware systems into truly distributed systems where each function may be hosted on different machines in a network and a central server coordinates the flow of information and control between these components. Other factors that have influenced the adoption of this architecture include availability of a large number of distributed information sources and sensors, ded-

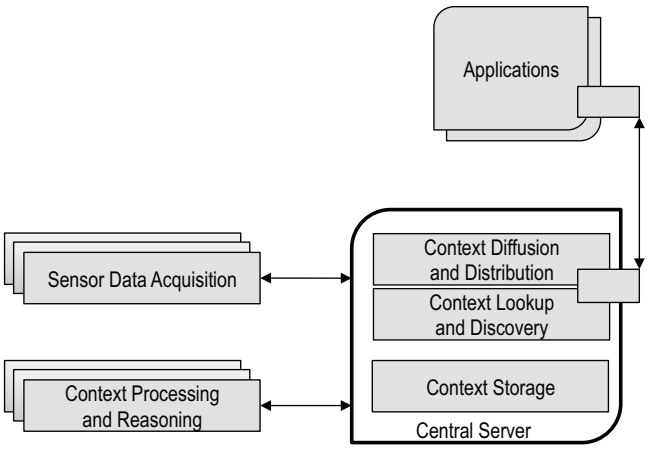

Fig. 11. Central server based middleware architecture with distributed components

icated reasoning components, mobility of modern day users and abundance and increased usage of mobile devices. Table II presents the architectural approach based categorisation of context-aware systems that have been classified earlier according to their design approaches (cf. Section IV-C) and provides a consolidated view of the overlap between design and architectural approaches of these systems.

E. Implementation

Network based frameworks have utilised *RMI* (Java Remote Method Invocation) or *CORBA* (Common Object Request Broker Architecture) standards as technological basis for interaction. Examples for RMI comprise JCAF (Java Context Awareness Framework) [33] and SOCAM [35]. CORBA has been used in Gaia [53]. Both CORBA and RMI can be considered as rudimentary technologies for realising distributed networked applications. Because of the relatively low abstraction level, it may outperform newer technologies as far as speed and a low footprint are concerned. However, development is more complex and the interoperability limited.

The *OSGI* framework is an open modular service platform based on Java that implements a dynamic component model as an extension to the standalone runtime environment. OSGI interface principles have been applied in [59] and [60]. *Multiagent systems* (MAS) evolved from Distributed Artificial Intelligence and are comprised of individual agents. Each is considered a locus of problem-solving activity. By operating asynchronously it has a certain level of autonomy and intelligence. Cooperative agents collaborate towards achieving common goals [61]. Hence, agent-based systems have also been successfully adopted in context provisioning middleware designs, for instance in CoBra [62] and EgoSpaces [63].

Moreover, *web service* technologies have recently received greater attention in development of context middleware. Web services utilise standard protocols, such as HTTP (Hypertext Transfer Protocol) and SOAP (Simple Object Access Protocol). In addition, WSDL (Web Services Description Language) may be applied for describing the interfaces of architectural components semantically. In the traditional client-server view, interacting services invoke each other, the requesting one becomes a client of the invoked service [43]. A full level of web service implementation has been used in CA-SOA (Context-aware Service Oriented Architecture) [64], CoSWAMI [55], ESCAPE [65], inContext [66] and Omnipresent [67]. The primary advantage of web service technologies is their wide interoperability across networks and devices. Representational State Transfer (REST) [68] compliance allows for stateless information exchange, e.g. based on XML. These concepts are not only applicable for context-aware web services but also for other types of actuators and ubiquitous computing applications.

F. Overview of Related Frameworks and Middleware

Table III provides an overview of related systems and their historic evolution. Furthermore, major achievements and limitations are discussed in the following paragraphs.

ActiveBadge [50], one of the earliest localisation and context-aware systems, is based on a centralised location server and communicates through Remote Procedure Calls (RPC). Mainly due to its very limited scope, the heterogeneity of networks, context consumption devices and sensors is not supported. Context representation is proprietary and details are not specified in related literature. The support for sensor, context and entity diversity as well as model extendibility and scalability are weak since only localisation of infra-red based sensors embedded in badges is targeted. The *ActiveMap* [20] system is comparably limited and again, location is the focus. Due to utilisation of the publish/subscribe paradigm, the context management scalability capability is supposed to outperform ActiveBadge, albeit no specific or comparative performance analysis has been published. Alongside users, the concept foresees support of generic objects. Hence, entity heterogeneity is addressed, however not discussed in detail. Localisation is arranged in hierarchies, for example room, building, regions. In addition to basic RPC communication, infra-red based multicast is supported. *Cyberguide* [49] was developed as a rapid prototyping system for assessing the utility of context-awareness in mobile devices and focussed on location and orientation context of users. Infra-red based beacons and GPS sensors are used for estimating the position of a device. Support of sensors, context and context processing capabilities are very narrow and static. *GUIDE* [57] is another framework supporting location-based travelling. The system, which uses HTTP-based communication, demonstrates an improvement over its predecessors by applying a distributed architecture in which separate servers manage individual cells.

The systems mentioned above have been primarily designed to illustrate the benefit of context-awareness based on a selected usage scenarios. Being very restricted in functionality, degree of awareness as well as acquisition and processing of contextual knowledge, they can be classified as early systems that increased interest and understanding of ubiquitous computing. However, they cannot be referred to as context provisioning middleware. The need for generic and systematic support of context-aware applications and services emerged slowly and has been addressed in later systems.

The *Context Toolkit* [48] is one of the first demonstrations of utilising a complex combination of different types of user context information. Its architecture is based on distributed

TABLE II
CONTEXT-AWARE MIDDLEWARE DESIGN APPROACHES AND SYSTEM ARCHITECTURES MATRIX

	Layered	Object Oriented	Event Based
Direct Sensor Coupling		Cyberguide [49]	
Central Server	Active Badge [50] CoBrA [23] OPEN [52]	CASS [51] Active Map [20]	
Central Server with Distributed Components	Gaia [53] PACE [32] SOCAM [35]	C-CAST [54] CoWSAMI [55] GUIDE [57] MobiLife [58] ACE [56]	C-CAST [54] ACE [56]
Peer-to-Peer	Hydrogen [37]	CORTEX [47] Context Toolkit [48] JCAF [33]	CORTEX [47]

TABLE III
CONTEXT MANAGEMENT & PROVISIONING FRAMEWORKS

Middleware/Project	*Communication Technology* ‡	*Ctx Model & Representation* §	*Support for Sensor Diversity* *	*Support for Ctx Diversity* *	*Ctx Model Extendibility* *	*Support for Entity Diversity* *	*Support for Historic Ctx* *	*Ctx Processing Scalability* *	*Ctx Management Scalability* *
Active Badge [50]	RPC	?	/	/	/	/	/	/	/
Active Map [20]	RPC & o	?	/	/	/	(✓)	/	/	(✓)
Cyberguide [49]	?	?	/	/	/	/	/	/	/
GUIDE [57]	web	?	/	/	/	/	/	/	(✓)
Context Toolkit [48]	web	M	✓	✓	(✓)	/	/	(✓)	(✓)
Gaia [53]	RPC, CORBA	o	✓	✓	?	✓	?	(✓)	(✓)
CoBrA [23]	ag, (web)	Ont	✓	✓	/	?	?	/	(✓)
SOCAM [35]	RMI, OSGi	Ont	✓	✓	?	?	?	?	?
CASS [51]	?	?	(✓)	(✓)	?	/	?	?	/
CORTEX [47]	o	M	✓	(✓)	?	✓	?	(✓)	/
JCAF [33]	RMI	OO	✓	✓	(✓)	✓	?	(✓)	/
PACE [32]	RMI, web	OO, M, G	✓	✓	(✓)	?	?	(✓)	/
CoWSAMI [55]	web	M	✓	✓	(✓)	(✓)	?	(✓)	/
MobiLife [58]	web	Ont	✓	✓	✓	?	?	(✓)	(✓)
OPEN [52]	?	Ont	✓	✓	/	?	/	(✓)	?
ACE [56]	web	Ont	✓	✓	(✓)	?	/	/	?

‡ web = web based; ag = agent based; RPC = Remote Procedure Calls; RMI = Remote Method Invocation; CORBA = Common Object Request Broker Architecture; OSGi = OSGi based; o = other; ? = unknown, i.e. not discussed in available literature.
§ Ont = Ontology; M = Markup Scheme; G = Graphical; OO = Object Oriented; o = other; ? = unknown, i.e. not discussed in available literature.
* ✓= support; (✓) = limited support; / = no support; ? = unknown, i.e. not discussed in available literature.

widgets that hide complexity and heterogeneity of sensors. The hybrid design further encompasses a central context interpreter and server. Context is represented by using a custom XML schema. Subscription based context updates are communicated based on the TCP/IP networking suite through SMTP (Simple Mail Transfer Protocol) or HTTP. While the system is by design not intended to be used over a large scale and is focussed on utilising context information bound to a geographical area, physical and functional scalability are partly supported.

Gaia [53] is the first context provisioning and management framework entitled as a middleware by the authors. It models a smart space as a programmable entity, providing support to context based applications via well established operating system concepts. Gaia applies I/O operations and file system manipulation for interconnecting active space objects. Event-based communication is established through both RPC and CORBA. Context diversity originates from the so called Context Providers and covers location, room conditions, weather, stock prices and executing applications. Gaia has not been assessed in terms of large-scale use. The authors mention federating multiple active spaces but such an attempt has not been documented.

CoBrA [23] is architecturally based on distributed agents being organised by a central broker. Context is represented in ontologies (RDF/OWL). The broker is responsible for providing a shared model of context, removing inconsistencies and masking heterogeneous context sensing sources. Communication is established through the Agent Communication Language (ACL) and HTTP. The concept of compartmentalising context into domains exists in CoBrA but physical scalability has not been evaluated. Functional scalability is problematic due to a tight coupling of applied technologies

and the resulting need for following these strict guidelines.

SOCAM [35] targets providing an architecture specified for rapid prototyping of context-aware services. The framework encompasses Context Providers, Context Interpreter, Service Location Service and Context-aware Mobile Services. Context Providers acquire primitive data which is further processed by a central Context Interpreter. Context is modelled based on ontologies (RDF/OWL). SOCAM is built on top of an OSGi service platform and also utilises RMI, hence facilitating service discovery and management through these technologies. The availability of both on-demand queries and subscriptions is only briefly mentioned without any details and, therefore, the scalability capabilities are difficult to estimate.

CASS [51] focusses on providing higher level context abstractions through a knowledge base, sensor listener, context storage database and rule engine (for reasoning) deployed on a server. Context-aware applications on mobile devices can listen for relevant changes in context data propagated through the central database. The rule engine monitors such modifications accordingly. Context representation is not discussed, however queries are realised through SQL. While prototype applications demonstrate feasibility of CASS in keeping resource intensive context processing tasks from mobile devices, it is not designed to be scalable. Firstly, both sensor nodes and mobile devices have to communicate with a central CASS server and no discovery mechanism is built in. Secondly, support for mobility is lacking.

CORTEX [47] proposes a sentient object model to support the construction of ubiquitous applications in wireless mobile ad-hoc networks. Sentient objects are defined as entities consuming, processing and producing events and effect actuators. Therefore, a publish-subscribe component for discovery of neighbouring components and sharing data is provided. Context is represented in XML and serves as input and output of sentient objects. While CORTEX is suitable for a number of mobile entities interacting with sentient objects, this communication is specifically designed to work in ad-hoc environments focussed around a particularly bounded area. Event propagation is limited by the range of multicast protocols, hence scalability is very limited.

The *JCAF* [33] system is based on a distributed, event-based infrastructure for development and deployment of context-aware applications. Each covered context domain is modelled by a dedicated context service being realised as Java Enterprise Application deployed on an Application Server. Inhabitants of such a domain are programmed as entities. Context is modelled "semantically-free" and utilises an object oriented representation. Context monitor and actuator components serve for acquisition and publication of context. Components in JCAF use RMI for communication which limits device heterogeneity severely. Due to the inadequate central RMI registry and unavailability of dynamic lookup facilities JCAF fails to support large-scale deployments.

The *PACE* middleware [32] provides a context management component along with a programming toolkit for development and deployment of context-aware applications. In addition, decision support and a flexible messaging framework are investigated. Architecturally a distributed set of context repositories is proposed, each repository managing a specific catalogue which is implemented as relational database. In addition to Java RMI, an HTTP based web interface is provided. The authors explicitly state that scalability is not a targeted design goal. Moreover, the middleware uses an event-based message routing scheme on top of RMI and HTTP communication. This configuration compounds the overall scalability due the fundamental differences in the communication protocols i.e. messaging notification of the routing scheme and object-oriented RPC of Java RMI.

CoWSAMI [55] addresses the issue of using context from dynamic sources by decoupling context sources from context aggregators and providing dynamic discovery. Context sources expose web service interfaces, including SOAP and WDSL technologies. Users can define relational context views and map them to the context source interfaces and aggregators produce context to satisfy these mappings accordingly. Both context and rules are encoded in XML. The application of CoWSAMI is limited to a particular application domain of mobile users and vehicles. Moreover, the loose coupling and dynamic discovery of context requires explicit involvement of users in defining rules and mappings, hence limiting the usability.

MobiLife [58] applies a role model comprising distributed Context Providers, Context Consumer and a central Context Broker in its Context Management Framework. RESTful HTTP interfaces are used for communication and ensure the support of device heterogeneity. Context is represented in ontologies. Entity diversity is not discussed, however the described application scenarios are clearly user centric. The design can be used for a large-scale *deployment*, however a single, central Context Broker can be identified as bottleneck in practice.

OPEN [52] tries to address the developers' functional requirements as well as users' simplicity requirements. Therefore, OPEN enables different layers of programming support from simple program parametrisation to incremental and composite programming paradigms. Several cooperation patterns are woven into the programming framework for encouraging the users' cooperation. Context queries are implemented by an adopted SPARQL context request and user defined rules are stored and executed based on a JESS engine. The usability of the system has been evaluated based on a treasure-game scenario. However, processing and administrative scalability aspects have not been investigated so far.

ACE [56] aims at closing the gap between the (1) management and retrieval of context and (2) the demands of higher-level context related tasks such as application triggers. An application context model is proposed to allow application developers to explicitly describe their context logic. The application context engine stores the described models and handles the whole lifecycle of the defined application. Experiments have demonstrated the use of the framework but also identified current limitations. The reasoning delay from the underlying inference machine prevents its practical usage in a large-scale system.

To summarise, a diverse set of design approaches has been adopted for developing context provisioning frameworks and middleware. The general trend towards large-scale systems as well as the support of application and device heterogeneity

becomes obvious. However, recent work still fails to adequately display the desired features including middleware extendability and the required support for both emerging and evolving context-aware applications and services. This is not only related to the introduction of new context categories but also to the aggregation of detected context in various abstraction levels. Another prominent reason that hinders contemporary context middleware from stepping out of the bounds of lab environments into the real world is the lack of scalability and its appropriate evaluation.

V. Context Processing & Reasoning

In general terms, *reasoning* derives "conclusions from a corpus of explicitly stored information" [69]. In the original philosophical sense deductive, inductive, abductive, analogical and fallacious reasoning can be differentiated, the first two variants being most relevant: Deduction attempts to show that a conclusion necessarily follows from a set of premises or hypotheses. Induction tries to derive propositions about unobserved objects based on previous observations, either specifically or generally [70]. Anagnostopoulos et al. [71] describe context reasoning as "a process for inferring *new* context, previously unidentified on the basis of a-priori known context". In the domain of context-aware and ubiquitous computing and for the scope of this article, Context Reasoning and Context Inference are synonymously defined as follows.

Context Reasoning *and* ***Context Inference*** *refer to the automated deduction of high-level context from lower-level or primitive context. In addition to an aggregation of available context information, reasoning/inference may derive new conclusions based on existing context (i.e. measurable and observable facts) and an available knowledge base.*

Consequently, a significant number of mechanisms originating from the fields of artificial intelligence and knowledge-based systems can be adopted for context reasoning. Moreover, there is a strong relation to *activity recognition* (e.g. [72]) and *plan recognition* (e.g. [73]). Before reviewing the different context reasoning mechanisms, we discuss the role of a context middleware in the the following subsection.

A. Context Processing and the Middleware

The entire context detection process can be subdivided into subsequent stages as depicted in Fig. 12. A specific utilisation of such a layered design is for example discussed in [74]. From bottom to the top of the processing chain, the complexity of mechanisms increases while reducing the amount of data through aggregation. Real world events are initially sensed and sampled. Particularly if physical sensors are utilised, the collected raw data is preprocessed afterwards, either on-the-fly or in bulk. Pre-processing may comprise removal of noise and faulty data by applying filtering techniques, averaging sensor readings or fusion of redundant data. In the next step, features are extracted, for example by fuzzy logic or crisp filters, to derive low-level context. This primitive context may also be directly fetched from virtual and logical sensors and serves as input for inferring common-sense abstractions, for example a user's situation, his mental and physical activities, his intentions.

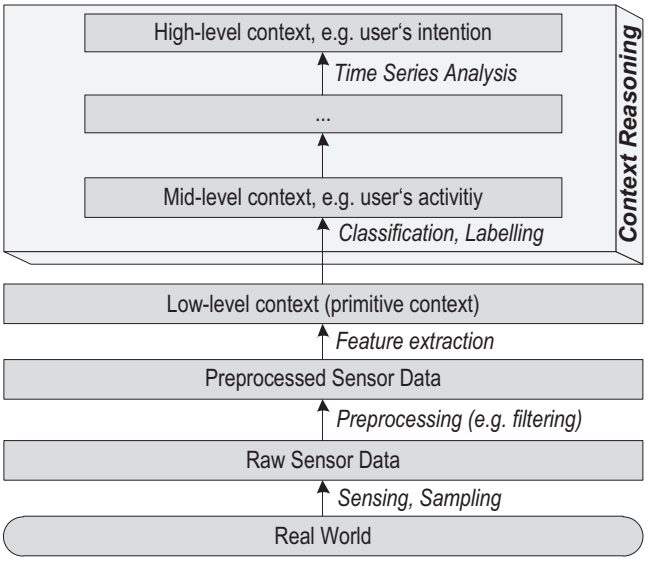

Fig. 12. Context Processing Stages

The process of context reasoning is evidently a complex one, and the context middleware is tasked with masking this complexity from other components and applications in the system. The encapsulation of complexity is important because it provides decoupling between the functionally separate components of the middleware and external applications. Consider a simple example of a reasoning component in a middleware that uses finite state machines (FSM) to determine if a person is free to take a phone call. This reasoning engine depends on input from various physical or virtual sensors that may include location, motion, phone status, noise, activity, etc. Assuming a common context model exists within the overall system, raw data from these sensors will serve as input the the FSM and may or may not result in a state that signifies a person's ability to take a phone call within that context. A different reasoning component, which may be present in the same middleware, may use a different reasoning mechanism, e.g. rule-based or probabilistic reasoning. The role of the middleware in such scenarios is to facilitate the data acquisition to feed the acquired data to the reasoning component without being concerned about what process the data will go through during the reasoning stage. Similarly, once the reasoning and context inference has been carried out, interested third party applications or other components of the middleware (e.g. context storage components) need only request the deduced context information. While this encapsulation of the context reasoning complexity is important for the vertical integration between the components/layers of a middleware, it is still critical to review the variety of context reasoning mechanisms that have been employed in existing context-aware systems.

Different reasoning mechanisms may be suited to different types of context domains, offer varying degrees of confidence in the contextual information that is based on fuzzy or incomplete data and have different computational costs associated with their application in practice. The factors are important to consider because mirroring the state of human activities into computational constructs is a challenging process. Kim

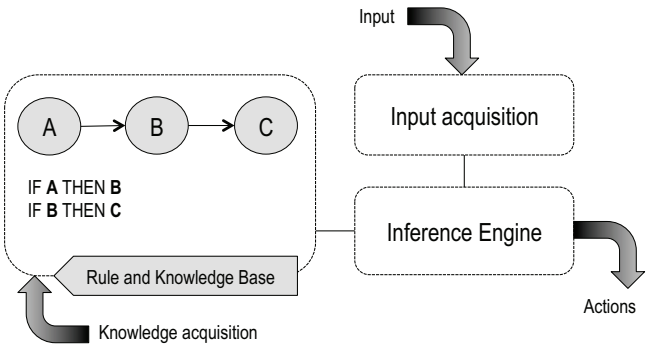

Fig. 13. Architecture of a Rule-based reasoning system

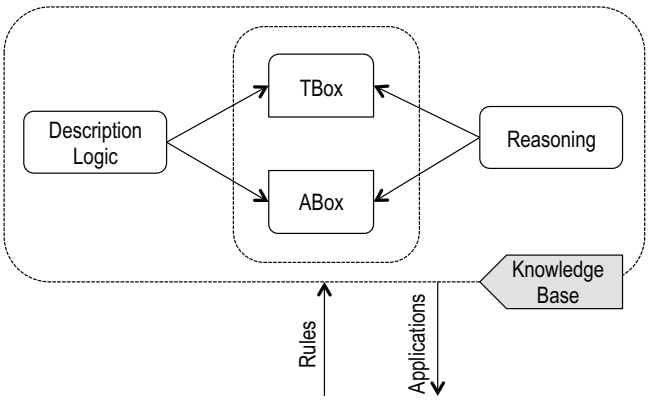

Fig. 14. Architecture of a Description Logic based system

et al. [72] identify typical challenges when dealing with the nature of human activities: (1) several activities may be performed concurrently; (2) activities may be interleaved; (3) the interpretation of activities may be ambiguous; (4) multiple residents may be present in a smart environment and follow collaborative activities. Identifying users' activities refers to either a classification problem or to a time-series analysis task [75]. Various reasoning mechanisms can be applied for recognising high-level context information. The selection of an adequate technique mostly depends on the targeted context, the available input information and the chosen context model. The following subsections present fundamental background information and discuss related work.

B. Rule-based Reasoning

As in any other rule-based system, rule-based context reasoning requires a fact base (knowledge base) and a rule base in which rules are stored. For inferring high-level context, primitive context is asserted into the fact base. Rule-based reasoning comprises a threefold cycle of (1) matching, (2) conflict-resolution and (3) acting. Whenever the fact base is updated, the internal inference engine checks if the rule base contains any matching premises. After solving potential conflicts, e.g. though consideration of assigned priorities, the rule fires, which may optionally change the fact base and/or trigger an asynchronous context update message to external components. This mechanism is also referred to as *forward chaining*. The general architecture of a rule-based reasoning system is shown in Fig. 13.

Rule-based reasoning comes with certain drawbacks. A large rule base easily becomes confusing and intractable. Particularly, if actions result in fact assertion, unforeseen and undesired chain reactions may occur which may not only halt the system but possibly even destroy the previously derived asserted facts and/or conclusions. Moreover, the storage of fact knowledge requires a large amount of memory consumption since the input context parameters of all entities need to be watched constantly. Most importantly, rule-based reasoning can only be usefully applied in systems supporting event-based context propagation. Boolean execution, i.e. rules fire or they do not, prevents the support of impreciseness.

C. Description Logic

Description Logic (DL), a compound of logic based formalisms for knowledge representation and reasoning, is applied in conjunction with ontological context representation. The semantic modelling of concepts (classes), roles (properties, relationships) and individuals allows terminological knowledge to be specified in a machine interpretable manner. An ontology is a formal specification of a shared conceptualisation of a domain of interest [76]. The so called `TBox` (terminological box) contains axioms relating concepts to each other, e.g. describing concept hierarchies. The `ABox` (assertional box) comprises ground sentences, i.e. it associates individual objects to concepts. Both `TBox` and `ABox` constitute the knowledge base whereas the `TBox` formalises intensional knowledge and the `ABox` extensional knowledge. An architectural representation of the description logic based system is shown in Fig. 14.

DL are variable-free fragments of First Order Logic. Several classes of formal expressiveness are differentiated. The smallest propositionally closed DL is $\mathcal{ALC}$ and comprises atomic concepts, atomic rules, conjunction, disjunction, negation, existential restriction and value restriction [77]. $\mathcal{S}$ is used if transitive roles are supported moreover. Additional letters indicate further extensions, e.g. $\mathcal{H}$ for role hierarchy, $\mathcal{O}$ for nominals/singelton classes, $\mathcal{I}$ for inverse roles, $\mathcal{N}$ for number restrictions, $\mathcal{Q}$ for qualified number restrictions and $\mathcal{F}$ for functional number restrictions. $\mathcal{SHIQ}$ is the basis for OWL, $\mathcal{SHOIQ}$ for OWL-DL and $\mathcal{SHIF}$ represents the formal range of OWL Lite.

Description logic based reasoning has been applied in [34] for inferring users' activities in the home domain (e.g. watching movies, having dinner, taking shower) based on an OWL representation. In [62], ontologies describing places, locations and activities have been defined and used for reasoning by an OWL inference engine.

Performance measurements of DL based reasoning have been conducted in [34] and [78]. Researchers share the opinion that reasoning based on ontologies is computationally intensive and that response times largely depend on the size of the data set and rule set. Scalability is a problem addressed in [78], in which a hybrid approach combining relational and semantic representation is proposed. In addition to the reasoning complexity, designing an ontology is a complex endeavour requiring domain knowledge and expert agreement. In addition to a base layer of (potentially too large) commonly defined ontologies, more specific application dependant on-

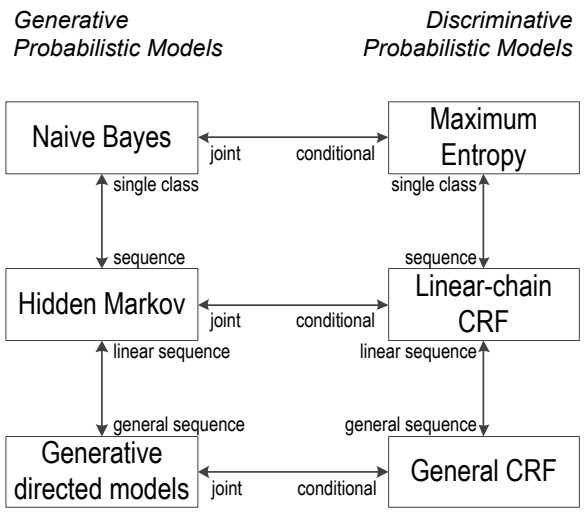

Fig. 15. Probabilistic Reasoning Models

tologies need to be designed on top. Extra effort is needed if uncertainty and unavailability of context are to be supported.

D. Probabilistic Logic

Probabilistic models can be distinguished as *generative* models or *discriminative* models as depicted in Fig. 15. Assuming input values x and the target class variable y, generative models require joint probability distributions $p(y, x)$ to be represented whereas discriminative models are based on conditional probabilities, i.e. $p(y|x)$ [79]. Both categories of probabilistic models can be represented graphically. Each node represents a random variable. The conditional independence of two random variables is graphically represented by the absence of an edge between the associated nodes. Hence, causal dependencies become human comprehensible. Let $G = (V, E)$ be a graph with vertexes V and edges E. The vertexes $V = X \cup Y$ are depicted as circles, X being the set of input or observation variables and Y the set of output variables. Generative models are represented as *Directed Acyclic Graph*, i.e. their edges are directed. Discriminative models are undirected and link random variables to so called factor nodes graphically represented as small filled rectangle. A factor Ψ_s comprises all random variables to which the factor node is connected.

The *Naïve Bayes (NB)* model [80] is the simplest generative approach for classifying single class variables in dependence of multiple feature values. The *Hidden Markov Model (HMM)* [81] extends NB for representing sequentially structured data, allowing modelling of temporal changes. Like the NB model, the Maximum Entropy model [82] is used for classifying a single output variable based on a set of observed input parameters. In contrast to NB, it contains conditional probabilities, hence it is discriminative. Linear-chain Conditional Random Fields (CRF) [79] can be interpreted as a linearly sequential extension to the Maximum Entropy model. Generic sequential analysis of data is also supported when using the unconstrained CRF models. According to Lafferty et al. [83] CRF tend to be more robust than generative models to the violations of their independence assumptions.

In particular, physical sensors are likely to give imprecise measurements. Moreover, due to potential temporary lack of communication facilities, sensor data may not be available at all. This is why Probabilistic Reasoning is especially applicable in context-aware environments. Furthermore, probabilistic models, i.e. causal dependencies, can be learnt from training data, e.g. by applying the Expectation Maximization or Maximum Likelihood methods. With regard to context-aware systems, such training data may be collected from user based feedback. Sequential probabilistic models can be represented as finite state machine (FSM) with specific transition probabilities between states. This is particularly useful, if the context-aware system tries to detect real-world activities that usually occur in a particular order.

In related work, both CRFs and HMMs have been applied for recognition of human activity in the kitchen domain [72]. A Naïve Bayesian Classifier is used in [74] to identify various activity context features taken from audio sensors and accelerometers; such as driving a car, running to the door, taking an elevator and listening to music. Oliver et al. [84] propose theoretical concepts of Layered HMM allowing for inference at multiple levels of temporal granularity. Dynamic Belief Networks (DBN) (a generalisation of Bayesian Networks incorporating temporal dependencies) and Linear Dynamical Systems (LDS) (a more general form of HMMs without constraints on number of state spaces) and their applicability to human activity recognition is discussed by Turaga et al. [85].

E. Other Reasoning Mechanisms

Wen-Yu et al. [86] use *Situation Calculus* to model ubiquitous information services. Situation Calculus [87] is a formal first-order language defining actions, objects, situations, precondition axioms and successor state axioms. Sharing its basic philosophy with commonly applied FSM, Situation Calculus eliminates the need for a-priori definition of possible states since situations are dynamically generated and an infinite number of instances is allowed. This makes Situation Calculus eligible to model an open dynamic world but makes it unmanageable and not applicable for context-aware applications that expect a defined output state. Situation Calculus has been applied for incremental plan recognition in [73].

A combined approach of Fuzzy Logic and clustering is presented in [88]. Imprecise reasoning about situational context and unsupervised model learning are supported. Yin et al. [89] present an activity recognition algorithm segmenting low-level sensor data with a probabilistic model. Each segment of signals is represented as an LDS model where transitions are modelled as Markov processes. The overall goal is to derive high-level activities from Wi-Fi radio signal strengths.

A notable number of approaches apply workflow-inspired patterns focussing on temporal flows. Bosse et al. [90] and Both et al. [91] introduce the Temporal Trace Language (TTL), a formally specified language based on predicate logic which shares similarities with Situation Calculus. Inference rules are translated into temporal rules for describing the reasoning behaviour in temporal partial logic. Both forward reasoning and backward reasoning are supported for deriving a human's progress in task execution while observing his behaviour.

F. Overview of Reasoning Mechanisms

Table IV provides an overview of the approaches presented above. It rates the support of machine based model learning, sequential data analysis and whether the mechanism can cope with imprecise and unavailable input data. In summary, a diverse set of mechanisms has been successfully applied to derive a rich set of high-level context.

VI. Evaluation of Ubiquitous Middleware

A. Challenges of Middleware Evaluation

Context-Awareness and Ubiquitous Computing are both multidisciplinary research areas, incorporating elements of HCI design, artificial intelligence, communication engineering, psychology and social/behavioural sciences. Correspondingly, the evaluation of such a complex and interwoven system is extraordinarily challenging and requires multidisciplinary techniques. Interesting studies and surveys have been conducted with focus on the evaluation of a UbiComp *application* from the user's perception, for example in [98], [99]. Proposed methodologies for context-aware systems are borrowed and extended from the background of HCI research.

Though the users' experience is essential, the work presented in this article concentrates on the evaluation of the context management middleware rather than on the applications themselves. Hence, a context middleware is assumed which is able to support the users in their everyday life with their device(s) moving through their usual environment seamlessly. The middleware has the difficult task of bridging the virtual and real worlds, which comes with numerous challenges. One problem in evaluating a middleware supporting rich context-awareness is that a generic one-size-fits-all approach is unrealistic [100]. In related work, evaluation often fails to be objective and comparable. A common consensus is required, not only related to UbiComp applications but also to the underlying middleware caring for holistic or partial awareness.

Neely et al. [100] explain the need for a common evaluation framework and for a combination of multidisciplinary approaches. We agree with their opinion that evaluation is always goal oriented and needs to be tailored accordingly. The following sections survey existing methods, propose recommendations for usage and discuss limitations.

B. Prototyping

Prototyping the conceptualised middleware is the most commonly applied evaluation technique. Usually, core functionalities are implemented and deployed in a controlled lab environment. In an early stage, lab staff and colleagues are usually taken as test subjects. This community is rarely extended to external participants. Selected applications or services are realised to demonstrate the benefit of the system under test and prove its correct functionality. Testbeds have been developed, for instance, in the Active Badge project [50], for the Active Map Service [20], the Cyberguide [49] and GUIDE [57] project. Other examples comprise an Office Assistant [101] and Gaia [53]. This kind of evaluation technique has its advantage in illustrating the final user experience while hiding middleware complexity. Obviously, it has been frequently used in early work where the innovative idea of UbiComp and context-awareness had to be delivered to a non-technical audience. But still, recent evaluations have focused on and are sometimes even restricted to building a small demonstrator (e.g. inContext [66], MobiLife [58]). A disadvantageous side effect of achieving middleware transparency in such evaluations is that the internal performance metrics do not become visible. Testbeds seldom succeed in proving scalability. Showing the context and application flexibility and variety requires enormous efforts (time and costs) since a lot of disjunct and complementary services need to be prototyped. However, prototyping is a suitable choice if either the end user view is to be investigated or if the ability of rapid prototyping is to be assessed (e.g. Context Toolkit [102]). Nonetheless, the experiences while coding the concepts often help in identifying shortcomings and improving or extending the architectural/functional model. The main target is to show the capabilities of the middleware based on experimental applications. As with other HCI evaluation, the user experience can be explicitly (e.g. interviews, questionnaires) or implicitly (observation, application based feedback) evaluated, albeit this view may not necessarily mirror the middleware capabilities but only the look & feel and usability of the applications.

C. Field Trials

1) Basic Approaches: Numerous methodological approaches have been used for evaluating UbiComp applications in the field, particularly their HCI design. Before discussing their relevance for *middleware* evaluation, the most important techniques for collecting and analysing user behaviour and experience are briefly introduced; see [99], [103]–[105] for more details.

- *Interviewing*: Qualitative data is gathered by evaluators asking individual users or groups open-ended questions about their work, background, ideas, etc. The answers are captured by a combination of note taking, audio and video. Though this approach is inexpensive, automatic everyday actions might not be reported. When being conducted outside the natural environment, the user might forget to mention important information and impacts.
- *Direct Observation* or Contextual Field Research: Trained observers directly follow the test subjects and note down their behaviour. This costly, time-consuming and disruptive technique may utilise photographic and video analysis and is inappropriate in private settings and for large-scale scenarios. One of the most relevant advantages is that users do not have to recall their actions and the quality of data is independent from variations in their individual reporting.
- *Self Reporting*: *Recall Surveys* that are based on users orally reporting their behaviour suffer from recall and selective reporting biases. Activities are often either not remembered or incorrectly reported. Alternatively, users note down their daily routines in *Time Diaries*. This method provides less biased data but due to distraction and annoyance, the records usually do not provide complete information.

TABLE IV
CONTEXT PROCESSING & REASONING APPROACHES

Reasoning Approach	*Requirements*	*Impreciseness & Unavailability* *	*Sequential Analysis* *	*Machine Learning* *	*Applicability (Examples)*
Rule-based Reasoning	Event-based context propagation; scalable fact storage	/	(✓)	/	Situation Recognition [92]
Description Logic	Domain knowledge represented in ontologies	/	✓	/	Home activities [34], locations [62], unspecific/generic [93], [78], [94], [95]
Situation Calculus	Formalised language model	/	✓	/	Modelling ubiquitous information services [86]
Naïve Bayes	Labelled training data	✓	/	✓	Activity Recognition [74]
Hidden Markov Models, Linear Dynamical Systems	Labelled training data *or* knowledge about conditional probabilities, availability of historic context	✓	✓	✓	Activity Recognition [85], [89]
Bayesian Networks	Labelled training data	✓	/	✓	Generic High-level Context Reasoning [96]
Dynamic Belief Networks	Labelled training data, availability of historic context	✓	✓	✓	Activity Recognition [85]
Conditional Random Fields	Labelled training data *or* knowledge about conditional probabilities, availability of historic context	✓	✓	✓	Activity Recognition [97]

* ✓= support; (✓) = limited support; / = no support.

- *Usability Testing*: Conventionally, usability data is collected using a combination of methods (observation, interviews, questionnaires) in a controlled setting, usually a lab, where equipment for recording is available. Tailored to software applications executing on desktop computers, the primary goal is to determine whether an interface is usable by the intended user population.
- *Lag Sequential Analysis (LSA)*: Originating from development psychology, the LSA technique allows for acquisition of quantitative data by observing users performing their normal activities. Evaluators can generate statistics such as frequency and conditional probabilities of events. A drawback is that users being recorded by video may alter their behaviour, knowing they are being observed.
- *Automatic Tracing*: Instead of a human evaluator, the behaviour is tracked by an autonomous systems with the help of various sensors and devices. User input or feedback is not supported.
- *Wizard of Oz*: In a Wizard of Oz experiment, subjects typically interact with a computer system. This system is believed to work autonomously but its functionalities are in fact being performed by unrecognised human being(s).
- *Obstacle Course Data Collection*: Evaluators prepare a to-do list for the test subjects. Instead of just observing their natural behaviour, specific activities or goals are defined which the user has to follow.
- *Experience Sampling Method (ESM)*: This technique, also referred to as *Ecological Momentary Assessment*, shares similarities with Self Reporting but avoids the retrospective character. An alarm is presented to the user during or shortly after the behavioural activity has been conducted. Upon reception of the alarm, the user has to answer questions or report what he is doing or feeling. This way, data does not suffer from recall bias. Commonly used ESM tools comprise PDAs, smart phones, paper booklets, mobile phones, traditional phones, audio player/recorder, pagers, watches, cameras or custom devices [106]. ESM allows the collection of both structured qualitative data (by defining fixed responses) and unstructured qualitative data (by asking open-ended questions).

The presented techniques are not only useful for UbiComp application assessment. The recognition of high-level context (i.e. *reasoning* or *inference*) can best be evaluated by the users whose context is to be derived. Three different main targets can be identified:

1) Evaluate the reasoning/inference algorithms qualitatively and quantitatively, i.e. assess their accuracy by comparing expected (calculated) context against actual (user reported) context;
2) Collect training and feedback data for reasoning/inference algorithms that support supervised machine learning;
3) Evaluate the overall user experience, particularly as far as adaptive service/application execution is concerned.

The evaluation of ubiquitous everyday assistance benefits from *in-situ* studies, also referred to as *in the wild* or *ethnomethodological*. The key idea is to collect data while users are situated in their own natural environment rather than creating an artificial one in the lab which might influence their behaviour [99]. Due to the characteristics of the UbiComp domain (user mobility, invisible sensing systems, distributed and fragmented interaction across different applications and devices [107]), the ESM technique appears most promising, in particular if combined with automatic tracing and logging.

Preferably, the data collection is performed by common everyday devices so that users do not have to wear additional hardware. It is important to minimise distraction from their ordinary workflow. Especially if long-term studies are envisaged, minimal obtrusiveness on the user experience is essential. When applying ESM, this can be achieved by supporting (1)

event-based context-aware triggering of questions (i.e. *when* is the alarm activated?), (2) context-aware selection of the alarming mode (i.e. *how* is the alarm presented?, e.g. audible vs. tactile) and (3) context-aware selection of questions and possible answers. Bearing in mind the variety of context, ESM tools may further serve as appropriate *human sensors* for understanding subjective areas such as emotional state, time use, social interactions, intentions, abstracted activities, habits and goals [108]. Another advantage of ESM is that evaluators can collect preliminary data in near real time and the context recognition accuracy can be measured quantitatively by applying stochastic methods. However, the individuality of users and their cognitive perception of events and activities (cf. [109]) results in potential errors and is problematic for learning generic models. The same applies to long-term activities consisting of short-term successive or even parallel actions. Multitasking, false starts and human error can be serious dilemmas when assigning labels to time sequences [110].

2) ESM Tools: Various tools for the evaluation of applications have been designed, primarily based on ESM. Four of them are briefly presented below.

The *Context-Aware Experience Sampling Tool (CAES)* [111] addresses HCI researchers and allows them to acquire feedback from users in particular situations detected by sensors. It runs on PocketPC devices and supports context-aware alarm triggering for the ESM. The main target is to assess the application interface design while concentrating on defined situations, hence decreasing the degree of interruption and annoyance. The flexibility of questions presented to the test subjects is increased by enabling the dynamic upload of new protocols. Sequences of questions, multiple choice and multiple response questions are supported, as well as recurrence patterns and randomisation. Audio and photographic feedback can be collected.

MyExperience [105] combines autonomic caption of objective sensor data (passive tracing) and subjective user feedback (active context-triggered ESM). The open-source implementation is based on .NET and available for Windows Mobile devices. MyExperience utilises a three-tiered, event-driven architecture of sensors, triggers and actions – hence supporting conditionally triggered report surveys and SMS/Email notifications. Moreover, it allows for preliminary data collection by providing direct remote access to collected data without any need for physical access to the device under study. An XML schema has been designed for enabling the scripting based remote configuration or control of the evaluation client. Therefore, study customisation (i.e. definition of surveys, triggers and actions) is simplified and accelerated.

Momento [112] utilises an architecture that comprises a (1) desktop platform being used by the evaluators in order to configure and monitor an experiment, (2) a server component and (3) mobile devices collecting sensor data and applying ESM. The desktop platform can be connected to the fixed applications via a Context Toolkit. The mobile client is realised based on Java and C#. It can be configured via simple text files in which rules for ESM alarm triggering can be defined.

The *ActivityDesigner* [110] focuses on the test-driven design process of UbiComp applications. It refers to the domain of Activity-Centered Design (ACD) and targets an activity-based UbiComp prototyping. Long-term everyday user activities are captured over an extended time period. A graphical user front-end is used to analyse automatically generated activity journals and to design prototypes for defined storyboards (i.e. activities, actions, scenes and the transitions in between). These prototypes can be deployed in a virtual machine (high-end device) or by a web application based on the Google Web Toolkit. In summary, the ActivityDesginer tries to support the interlinked process of (1) field observations, (2) activity analysis/modelling, (3) interaction prototyping and (4) in-situ testing.

3) Trends & Evolution: The increasing penetration of smartphones and the availability of equipped sensors could significantly accelerate the evolution of ESM based clients without the need for any extra hardware. The basic requirements still remain the same: avoidance of massive battery consumption, systems crash and negative impact on the user experience. One constraint of earlier work was the very limited number of participants. In contrast, mobile applications can be developed and disseminated more easily nowadays. Distribution frameworks such as Apple App Store or Google Play Store allow large scale studies to be conducted. This would turn the human sensor into a *crowd sensor* and enable participatory sensing. The more feedback is provided by unknown and unbiased test subjects the more fruitful and meaningful the statistical analyses. However, *adaptive filter* techniques are required to support a focus on situations and users of special interest. Context-aware alarming could be combined with context-aware selection of test users. Both hoped and unexpected effects have to be monitored [99]. Another important aspect is the motivation of users. Long-term studies may be burdensome if the number of ESM triggered feedback requests is too high. A socially or gaming inspired reward mechanism may help. Finally, it is questionable how a test subject feels about being observed by strangers. In this regard, privacy, security and anonymisation have to be supported. Moreover, interested users would benefit from improved context recognition accuracy. In summary, field trials offer essential insight into otherwise hidden results. They are not only useful for the evaluation of UbiComp applications but also for the assessment and gradual optimisation of reasoning/inference algorithms.

D. Emulation & Simulation

In the literature, especially in the scope of UbiComp and context-awareness, the terms *emulation* and *simulation* are often used synonymously without much differentiation. In general, the goal of emulation is to be able to substitute the object that is emulated. The focus of a simulation is the modelling of essential features of the system under test. In general simulation does not necessarily lead to emulation. In particular, a simulation may run slower (or faster) than real-time whereas emulation requires the system under test to be at least partly available in a real deployment and invoked in real time.

Transferred to the domain of context-aware middleware, at least four basic categories of approaches appear useful:

1) *Emulating Context*: The system under test (middleware core) is used as is, i.e. prototyped system components

are directly invoked. But context does not originate from real world events and real users, instead it is emulated.

2) *Emulating Middleware Components*: The real life implementation of the middleware core remains untouched. However, the middleware is extended by further components, e.g. processors, sources and sinks of context information that are considered outside of the real prototype.
3) *Emulating Actuation*: In order to illustrate the adaptive behaviour of context-aware applications or actuators, a virtualised environment can be added.
4) *Middleware Simulation*: The system under test is simulated entirely. The contextual input as well as the functionalities of the system components are modelled in an abstract manner. There is no real life deployment required outside the simulation environment.

These four approaches are analysed and discussed in more depth below before related work is summarised.

1) Context Emulation: The emulation of context aims at ensuring controlled test conditions. In a UbiComp environment it is impractical to send real users around in the real world and ask them to follow a defined behaviour. That is why the real world (or parts of it) are substituted by a virtual world designed by the evaluators. The reaction of the middleware based on events and context changes in this virtual world is then evaluated. The key challenge is to represent the physical world *reasonably and realistically* and not just randomly. One approach is to define labelled and parameterised sequences of context changes (e.g. "having breakfast at home"). These templates can then be executed by the emulation environment to create a vivid setting autonomously. Alternatively, a Graphical User Interface (GUI) allows for temporary manipulation of selected parameters. The following targets can be addressed:

- Context Emulation allows for generating huge amounts of context information. Not only knowledge about the individual user but also the number of users can be easily increased in the virtual test world. Real context is duplicated and associated to pseudo entities (also known as avatars) [113]. The same applies to device context. Therefore, a real middleware system can be probed in order to analyse its scalability behaviour under almost realistic circumstances.
- Context reasoning, aggregation and processing in general, can be evaluated if primitive context (e.g. acceleration, location) is emulated. The high-level context (e.g. activity) output of the middleware is then compared against expectations. This requires context to be recordable and replicable.
- The adaptation behaviour of applications and other actuators can be observed while manipulating the context.

It is essential to make recorded context profiles exchangeable so that the results of different approaches can be compared with each other. Efforts have been made by researchers at the Massachusetts Institute of Technology (MIT) who shared their sensor data collected in a smart live-in lab [114]. A representation schema is for instance proposed in [115]. The need for dataset exchange has also been emphasised in the BoxLab project at MIT (http://datasets.mit.edu). In their shared resources not only datasets are available for downloads but also annotation schemas. Current research focus is on smart homes but extensions towards a smart world covering everyday activities is desirable.

2) Emulating Middleware Components: The target of emulating middleware components is the substitution of hardware (e.g. physical sensors) by software. This approach is useful if new context types are to be added to the system. Before putting too much effort into the design and implementation, the behaviour of the middleware can be estimated beforehand. The effect of the number of components can be measured. This applies not only to context sources and processors but also to context consumers or sinks. Especially in an event-driven system the number of consumers is expected to influence metrics such as the notification time and network load.

3) Emulating Actuation: The evaluation of actuating components can be tricky in a UbiComp environment. Imagine a smart home adjusting the heating or light level according to its inhabitants. The smarter and larger the space is, the more expensive becomes classical prototyping. Therefore, evaluators experimented with virtual three-dimensional graphic engines, e.g. those used in popular games including Half Life, Quake III Arena, and Unreal Tournament, to produce a realistic demonstration of how the environment would react to context events.

Obviously this approach fails if we consider a large scale, e.g. an urban smart space. Even in a geographically small area, it is questionable if a realistic and fancy layout would add value and insight to the middleware evaluation. Only if the emulation of actuation is combined with context emulation, these techniques might improve understanding and the look & feel of selected UbiComp spaces. Still the focus of emulating actuation remains on the end user perspective and experience rather than on the middleware.

4) Middleware Simulation: The methodology of a system-level simulation is borrowed from classical communication engineering. The behaviour of the middleware, including all its distributed components, is modelled. Typically, only limited core functionalities and the exchanged messages are included in the simulation model where Discrete Event Simulation can be applied. When building a simulation model the following questions and challenges need to be addressed:

- The appropriate degree of abstractness must be identified for the simulation model. A model with too much details will increase development costs and simulation time without providing more realistic results. The model depends on the specific goals of the simulation [116].
- The parameters (input values) and metrics (output values) must be chosen. In a UbiComp setting, the definition of reasonable scenarios is extremely complex. Not only the user behaviour but also the application behaviour (or in general the requirements of context consumers) has to be estimated. This comprises the number of context requests/subscriptions per unit time, the amount and frequency of context changes.
- In addition, as far as a distributed middleware is to be evaluated, knowledge about the behaviour (e.g. response time, processor and memory consumption) of each component is required. The key is finding simple

parameters that are able to represent the real system without simplifying too much. Obviously it is difficult – if not impossible – to build a realistic simulation model without ever having implemented a prototype of the middleware and feeding the simulation parameters with metrics determined in the real testbed.

The essential advantage of system-level simulations is that parameters can be varied easily to investigate various scenarios. Parameters like the number of users, number of devices, the amount of context types, the context query frequency can be used to estimate the large-scale performance of a middleware. Network load, context traffic and response times are interesting metrics to observe. If modelled well, the simulation can help in identifying bottlenecks of the system and provide guidelines for improving the concepts and algorithms. To facilitate the comparison of results obtained by different researchers, common simulation scenarios (sets of parameters) need to be defined in the future.

5) Simulation/Emulation Tools: Categories of emulation and simulation are not necessarily differentiated in related literature. There is a body of research which concentrates on three-dimensional graphical user interfaces. Testers can navigate around in the first person perspective and see the (emulated) actuation capabilities of the smart space based on emulated context changes, as expanded below.

UbiWise [117] is built on the Quake III Arena graphics engine and allows multiple users to participate in interactive UbiComp scenarios. The physical environment view is simulated by a subcomponent called UbiSim whereas a device view is provided by a separated module, realised in Java, named Wise. As an example the authors emulate a UbiComp camera and an actuating picture frame. Real web service interaction is supported as well. The emulated camera can be controlled in the Wise view and it can be carried around in the UbiSim 3D-simulator environment. Spatial and environmental interactions of devices and users are focused.

A *Hybrid Test and Simulation Environment* [118] has been developed by researchers at Lancaster University. In contrast to UbiWise, their approach does not provide a three-dimensional GUI interface. According to the system under test, the interfaces of the evaluator components are based on Web Services (HTTP and SOAP). Important features are centralised logging and the support of configuration and automated test scripts. The simulation environment focuses on the evaluation of location-based applications under consideration of different radio access technologies. Quantitative measurements such as the response and transport delay have been taken.

CIVE [119] presents a context-based interactive System for distributed virtual environments. It tries to bridge the gap between real and virtual cyber-world with focus on how to deliver real UbiComp interaction (e.g. gestures) into cyber systems. The submodule *ubi-UCAM* generates contexts and user profiling out of real world interaction, whereas *NAVER* cares for a virtual heritage and sensing of users' activities in the cyber environment. Another component called *INTERFACE* connects those two.

TATUS [113] also aims at building a virtual ubiquitous computing environment. The complexity is reduced by emulating sensors and actuators. Users' devices are not emulated but the original ones are connected to the simulator. Further, rapid scenario and virtual world modelling are facilitated by providing a design toolkit allowing for a reasonable level of realism, software flexibility, experiment usability and simulator extensibility support. Moreover, a Wireless Network Simulator has been designed in order to evaluate the performance of communication systems used in UbiComp environments. Its main purpose is to model channel conditions based on distance and obstacles (human beings, walls, etc.).

UbiREAL [120] is another simulator addressing virtual 3D spaces. Its developers extend previous work by generating more realistic context and supporting systematic testing. Both virtual and real devices can cooperate via Ethernet. Besides a network simulator and the 3D GUI interface, UbiREAL comprises a simulator for physical quantities and a test suite. The former one models the change of physical conditions based on triggered actions (e.g. the room becomes warmer if the heating has been switched on). The latter one can be used to define action rules and expected outcomes in a formal representation schema, hence allowing for autonomous validation of results.

CAST (Context-Awareness Simulation Toolkit) [121] concentrates on a virtual home domain and allows creating virtual context. Security aspects are taken into consideration when sharing such context between simulator components. Conceptually a Virtual Device Manager is responsible for emulating sensors and context consumers. Pseudo user entities can be generated by a Virtual Person Manager and the Virtual Home-Map Editor allows for designing the virtual smart space. The two-dimensional graphical presentation is based on Macromedia Flash and fails to provide a realistic impression of the setting. The comic-like approach is not suited for middleware evaluation either.

UbiHolo [122] is rather a UbiComp software paradigm than a simulator. However, the authors' work deserves attention because it contains one of the few approaches to apply system-level simulation in the UbiComp domain. To evaluate the large-scale performance of their concept, a peer-to-peer simulator called p2psim has been utilised.

Siafu [123] is a representative example of a Java-based context emulator. Based on a two-dimensional GUI, it allows the comfortable manipulation of context temporarily. Since it relinquishes the three-dimensionality, it is suited to cover (spatially) larger areas such as urban smart spaces. Siafu can generate its own simulated context (e.g. by defined user movements) and in parallel incorporate real world context acquired from sensors. Datasets can be produced for machine learning and effects of context changes can be demonstrated and visualised by plugging an application.

C-ProMiSE (A Context Provisioning Middleware with Support for Evolving Awareness) [124] has been quantitatively evaluated by discrete event simulation. The simulation models have been derived from black-box assessments of prototyped systems components. The definition of various context query scenarios and context update assumptions facilitates a systematic analysis of the overall performance.

E. Comparison of Methodologies

This section has highlighted the dependence on multidisciplinary approaches and challenges in evaluating middleware for ubiquitous computing. Techniques incorporating prototyping, emulation, simulation and field trials have been discussed and their utilisation in various systems has also been presented. These techniques not only target different – and often mutually exclusive – objectives, but also require varying aspects of functional availability in the middleware being evaluated. Table V provides a comparative summary the these evaluation techniques in terms of targets and requirements along with examples of their utilisation in existing middleware systems.

VII. Summary and Outlook

A. Synopsis

This article has presented a survey of context provisioning middleware with emphasis on context modelling, context management, context processing and evaluation of context middleware approaches. Related work has been analysed and categorised correspondingly.

The main points of this article and key findings from the analyses of various domain specific topics can be summarised as follows:

1) Context-awareness is a key feature to enable the vision of ubiquitous computing. A Context Provisioning Middleware can aid in collecting sensor data, detect the user's situation and make it available to context-aware services and applications. Its major requirements are functional and physical scalability while ensuring coherent and simple access to contextual information.
2) A broad definition of context helps in identifying a large set of application domains. Early systems were limited to spatial context whereas recent, more innovative systems tend to increase the abstraction level, e.g. activity or plan recognition. Semantic *context modelling* utilises ontologies represented by semantic web technologies. Numerous researchers highlight their difficulties when designing ontologies and applying reasoning with large ontologies; therefore a layered approach has been proposed with common sense ontologies at the bottom and domain specific knowledge on top.
3) Recent work in the area of *context management* architectures concentrates on web service approaches and tries to support modular extendibility to cope with emerging services and applications; furthermore the interoperability and wide utilisation of web standards support heterogeneous devices and (radio access) networks.
4) A number of different approaches, from various domains of computing and artificial intelligence, have been used in the complex functional task of context reasoning/processing. Different reasoning mechanisms may be suited to different types of context domains and offer varying degrees of confidence in the processed contextual information. The selection of an appropriate technique depends on the type of context being reasoned about, available input data and the adopted context model.
5) The approaches to context middleware *evaluation* are often limited in focus to functional assessment, particularly through prototype implementation. The size and number of playgrounds and test beds is increasing. However, context-aware systems have not yet stepped out of the laboratory environments into the real world significantly.

B. Lessons Learnt

Based on the multi-spectrum review of the domain of context-awareness in general and context-provisioning middleware in particular, we summarise the lessons learnt in this section:

- Because it deals with the practical application of wide-ranging computing concepts, ubiquitous computing in general is becoming indistinguishable from other fields of computing. This aspect highlights both the cross disciplinary applicability and conceptual complexity in designing context-aware systems.
- We do not expect a single reasoning/inference technology or a single context model to be able to cope with all application domains simultaneously. Instead, context conversation and translation mechanisms as well as a diverse set of context processing technologies need to be incorporated.
- The evaluation of context provisioning middleware is a complex endeavour. A fair and objective comparison of concurrent context representation and management approaches is challenging mainly due to the fact that the amount of published measurements is insufficient. Here we encourage the definition of useful metrics and a common evaluation methodology that enables the quantitative juxtaposition.
- The number of context sources and sinks is expected to increase tremendously with Internet of Things (IoT) deployments and sensor equipped smartphones. Therefore, the scalability of a middleware, both in terms of computation and administration, is a sincere concern.
- A systematic evaluation methodology definitely benefits from prototyping, (global) field trials, context emulation and system-level simulations.

C. Trends and Evolution

In addition to academic research, industrial developments have recently gained a foothold into context-aware computing. Dominant market forces, including Apple, Google and Microsoft, not only build and equip smartphones but also entire software ecosystems that include software development kits and software distribution platforms. An obvious trend is the seamless integration of social networking suites and tools for sharing user-generated content. Corresponding functionalities are integrated in the operating system. Siri (www.apple.com/uk/ios/siri/) and Vlingo (www.vlingo.com/) are prominent examples of mobile applications that provide a speech-based HCI and constitute the first practical realisation of a context-aware personal assistant.

User-generated content and voluntary disclosure of confidential information in social networks appears to be a more

TABLE V
OVERVIEW OF EVALUATION STRATEGIES

Evaluation Technique	*Targets*	*Requirements*	*Examples*
Prototyping	Most realistic proof-of-concept for end-to-end scenarios; provides input parameters for simulations	Implemented applications and test persons	Active Badge [50], Active Map Service [20], Cyberguide [49], GUIDE [57], Conference Assistant [125], Office Assistant [101], Gaia [53]
Field Trials	Assess Reasoning/Inference accuracy, collect feedback & training data	Context-aware experience sampling, dynamically reconfigurable by evaluators, easily distributable and deployable on everyday devices	MyExperience [105], Momento [112], CAES [111] [103], ActivityDesigner [110]
Context Emulation	Real-world context is extended or replaced by creating a virtual world; prototypes can be fed with input in order to evaluate scalability	For evaluating context processing, reasonable models must be defined (e.g. taken from recorded real-life context and associated to avatars)	Siafu, UbiWise [117], TATUS [113], UbiREAL [120]
Emulating Middleware Components	Substitution of hardware (e.g. physical sensors) by software; reduction of costs in large scale UbiComp environments; Estimation of consequences when extending the middleware	Emulation models must reflect the behaviour of the real hardware	UbiWise [117], TATUS [113]
Emulating Actuation	Providing a realistic outlook of context-based actuation in a smart environment	The influence of the actuation must be fed back into the system by producing emulated context (e.g. heating $\Rightarrow$ warmth)	CIVE [119], UbiWise [117], TATUS [113], UbiREAL [120]
Middleware Simulation	System-level analysis of large-scale behaviour	Abstract Simulation model, reasonable parameter assumptions (taken from prototypes)	UbiHolo [122], C-ProMiSE [124]

expressive source of context than physical sensor readings, at least for most of the application domains. Social and emotional sensors might alternatively utilise the Experience Sampling Method that has been discussed in this article. Sensor equipped smartphones are the users' primary source of interaction with the digital world; furthermore they allow for opportunistic or participatory sensing, i.e. multiple phones may form a crowd sensor. Data exchange may occur globally via the Internet or by low range communication and the utilisation of opportunistic networking [126].

There is a growing realisation of the need for real-world experimental facilities in order to advance the domain of ubiquitous computing. The *SmartSantander* initiative [127] uniquely conducts a smart city experiment that provides a smart space at an urban scale across four European cities, in essence becoming one of the largest living labs in Europe. It remains to be seen if the IoT concepts and related technologies will increase the size of such smart spaces to larger geographical areas or even globally.

Recent advances point towards a computing paradigm shift in progress, which can be perceived as an intermediary step towards realisation of ubiquitous computing. The first generation of computing provided one computer to many individuals (mainframes), the second generation allowed for one computer per individual (PC) and the third generation (distributed/mobile) computing has made possible many computers per individual. The current transformations are paving the way for a many computers–many individuals correspondence through the Cloud computing paradigm. It is interesting to note that while earlier generations maintained a strong identity association between the computer and the individual, future generations may no longer maintain that association due to virtualised (Cloud) and community administered (sensors) computing resources. In such computing environments, we believe that it will be the role of middleware to mask the numerous computing resources from users and manage the identity associations, which have accessibility, privacy and security concerns, at the same time.

D. Open Issues and Future Work

The expected evolution discussed above motivates further research contributing to the vision of a generic and evolving context provisioning middleware for utilisation in ubiquitous computing. Based on the current state of the art, the following objectives appear essential.

The paradigm of Cloud computing may aid in context processing and context storage by offering virtual resources that can be efficiently utilised on demand [128]. How to support and embed virtual resources in context modelling, context management and reasoning mechanisms is still an open issue. Moreover, privacy and security concerns increase with the available variety and amount of context – and its potential storage in the Cloud. Personal information allows virtually tracing users and their activities. Therefore, appropriate protection mechanisms are required. Particularly user management, authentication and authorisation need to be systematically incorporated. Third-party context access has to be bound to explicit grants or autonomous anonymisation. Further investigation is required to establish how to efficiently and safely support these mechanisms in a Cloud-based context provisioning middleware.

Recent middleware designs tend to unidirectionally detect and publish irrelevant context that is not required by any application/service at all or at least not in the provided granularity or accuracy. Hence, processing resources and energy are wasted. The efficiency could be increased by autonomous remote (re-) configuration of sensing devices according to the context demands of consumers. These demands may either be explicitly defined through context subscriptions or implicitly derived and predicted from earlier queries. The resulting feedback loop may be adequately addressed and analysed from control systems engineering point of view.

Most researchers have had difficulties in identifying a killer application for ubiquitous computing. Ubiquitous computing and context-awareness intend to pro-actively support users during their everyday life without any need for complicated interaction to the computing resources. This leads to the ap-

plication of ubiquitous computing and context-aware concepts in a huge variety of useful application domains. However, we underscore the quite obvious choice of an all encompassing *personal digital assistant* as the context-aware application of choice, which can provide an interface for different application domains and further route the demands to other domain specific services. In existing systems, the focus has been on the adaptation of the service/application logic and its execution. However, future research may investigate the possibility of context-aware service *composition* and context-aware service *deployment*.

Another avenue of exploration is the level of control delegation to context-aware systems. Even considering simple context-aware services, there is a significant amount of computing and processing in the background that the user is intentionally not made aware of. The system does not want to be intrusive and distract the user from his current activity i.e. there is not necessarily the need for a computing-aware user. However, the user needs to understand the system; and more importantly, the user needs to be able to easily overrule automatisms and thus enforce individualistic control. This need for individualistic control is important because though human beings collectively share certain common-sense logic and knowledge, a particular human being behaves with a certain individuality. This individuality has other ramifications as well, e.g. most reasoning techniques in existing context-aware systems apply probabilistic logic, which is based on probabilistic models derived through machine learning. These techniques depend on available training data that usually originates from an individual user and/or from a group of users. Hence, the training data is often tainted with individualistic *noise*, which may over represent random individualistic expressions. A useful trade-off for training data has to be identified to compensate for this individualistic *noise* and thus derive general context reasoning heuristics.

Finally, one of the early proponents of ubiquitous and context-aware computing, Abowd [129] states that ubiquitous computing has performed an intellectual *disappearing act* through its multi-disciplinary nature i.e. its ideas and concepts already pervade much of computing research and practice. This argument is not meant to close the chapter on focussed ubiquitous computing research but rather to emphasise the importance of this next generation of computing. Abowd argues that the intellectual agenda of ubiquitous computing has become so profound that it is increasingly indistinguishable from the overall agenda of computing research today [129, p. 2], echoing Weiser's argument that the most profound research topics are those that disappear by weaving themselves into the fabric of every research until they are indistinguishable from it [130]. This analysis best explains the diversity in the scope of the work discussed in this article, as it is increasingly difficult to analyse the state of the art in context-aware systems without addressing the multi-disciplinary nature of the domain of ubiquitous computing.

VIII. Conclusion

This article has presented a comprehensive review and analysis of how context-aware middleware systems undertake context modelling, management, reasoning and provisioning related functions. By examining the state-of-the-art in the multifarious aspects of context-awareness, the article has aimed at developing an understanding of the functional diversity of the middleware bridging the virtual and real worlds. We have also examined how such complex and interwoven systems can be evaluated through multidisciplinary approaches. Based on the discussions and analyses of various domain specific topics, this article has also presented the main trends and ongoing evolution of context provisioning. Open research issues have been identified and recommended for future work.

Acknowledgment

The work presented in this article has been partly funded by the European research project "Context Casting (C-CAST)" in Framework Programme 7. Its main objective is to evolve mobile multimedia multicasting to exploit the increasing integration of mobile devices with our everyday physical world and environment.

References

[1] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 272, no. 3, pp. 78–89, 1995.

[2] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, pp. 4–7, 2001.

[3] N. Kara and O. A. Dragoi, "Reasoning with contextual data in telehealth applications," in *Proc. Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, ser. WIMOB '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 69–. [Online]. Available: http://dl.acm.org/citation.cfm?id=1318474.1318622

[4] Y.-M. Huang, Y.-H. Kuo, Y.-T. Lin, and S.-C. Cheng, "Toward interactive mobile synchronous learning environment with context-awareness service," *Computers & Education*, vol. 51, no. 3, pp. 1205 – 1226, 2008.

[5] W. Choi, H. Kim, J. Kim, and J. Chae, "A context-aware framework for mobile navigation service," in *Computer and Information Technology, 2007. CIT 2007. 7th IEEE International Conference on*. IEEE Computer Society Press, Oct 2007, pp. 423–428.

[6] H. Chen, F. Perich, D. Chakraborty, T. Finin, and A. Joshi, "Intelligent agents meet semantic web in a smart meeting room," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, ser. AAMAS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 854–861. [Online]. Available: http://dx.doi.org/10.1109/AAMAS.2004.152

[7] J. Simoes and T. Magedanz, "Smart advertising in the home of the future," *International Journal of Computer Aided Engineering and Technology*, vol. 2, no. 2, pp. 164–180, 2010.

[8] B. Adams, D. Phung, and S. Venkatesh, "Extraction of social context and application to personal multimedia exploration," in *Proc. the 14th annual ACM international conference on Multimedia*, ser. MULTIMEDIA '06. New York, NY, USA: ACM, 2006, pp. 987–996. [Online]. Available: http://doi.acm.org/10.1145/1180639.1180857

[9] S. Björk, J. Holopainen, P. Ljungstrand, and R. Mandryk, "Special issue on ubiquitous games," *Personal and Ubiquitous Computing*, vol. 6, no. 5-6, pp. 358–361, 2002.

[10] A. Gupta, S. Paul, Q. Jones, and C. Borcea, "Automatic identification of informal social groups and places for geo-social recommendations," *International Journal of Mobile Network Design and Innovation*, vol. 2, no. 3, pp. 159–171, 2007.

[11] D. Cook and S. Das, *Smart environments: technologies, protocols, and applications*. Wiley-Interscience, 2005.

[12] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Trans. Autonomous and Adaptive Systems (TAAS)*, vol. 1, no. 2, pp. 223–259, 2006.

[13] E. Aarts, R. Harwig, and M. Schuurmans, *Ambient intelligence, The invisible future: the seamless integration of technology into everyday life*, P. J. Denning, Ed. New York, NY, USA: McGraw-Hill, Inc., 2001.

[14] H. Gellersen, "Smart-its: computers for artifacts in the physical world," *Commun. ACM*, vol. 48, no. 3, pp. 66–, Mar. 2005. [Online]. Available: http://doi.acm.org/10.1145/1047671.1047707

[15] A. Zimmermann, "Context Management and Personalisation," Ph.D. dissertation, University of Aachen, 2007.

[16] F. Giunchiglia, "Contextual reasoning," *Epistemologia, special issue on I Linguaggi e le Macchine*, vol. 16, pp. 345–364, 1993.

[17] C. Billings, "Situation awareness measurement and analysis: A commentary," in *Proc. International Conference on Experimental Analysis and Measurement of Situation Awareness*, D. Garland and M. Endsley, Eds. Daytona Beach, FL: Embry-Riddle Aeronautical University Press, 1995, pp. 1–6.

[18] P. Bellavista, A. Corradi, and C. Giannelli, "A unifying perspective on context-aware evaluation and management of heterogeneous wireless connectivity," *IEEE Commun. Surveys Tutorials*, vol. 13, no. 3, pp. 337 –357, quarter 2011.

[19] A. Dey, "Providing architectural support for building context-aware applications," Ph.D. dissertation, Georgia Institute of Technology, 2000.

[20] B. Schilit and M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, no. 5, pp. 22–32, 1994.

[21] N. S. Ryan, J. Pascoe, and D. R. Morse, "Enhanced reality fieldwork: the context-aware archaeological assistant," in *Computer Applications in Archaeology 1997*, ser. British Archaeological Reports, V. Gaffney, M. van Leusen, and S. Exxon, Eds. Oxford: Tempus Reparatum, October 1998. [Online]. Available: http://www.cs.kent.ac.uk/pubs/1998/616

[22] G. D. Abowd, E. D. Mynatt, and T. Rodden, "The human experience," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 48–57, 2002.

[23] H. Chen, "An intelligent broker architecture for pervasive context-aware systems," Ph.D. dissertation, University of Maryland, Baltimore County, December 2004.

[24] C. Burghardt and T. Kirste, "Inferring intentions in generic context-aware systems," in *Proc. 6th international conference on Mobile and ubiquitous multimedia*, ser. MUM '07. New York, NY, USA: ACM, 2007, pp. 50–54. [Online]. Available: http://doi.acm.org/10.1145/1329469.1329475

[25] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *First International Workshop on Advanced Context Modelling, Reasoning And Management at UbiComp 2004*, Nottingham, UK, September 2004.

[26] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 161–180, 2010.

[27] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca, "A data-oriented survey of context models," *SIGMOD Rec.*, vol. 36, pp. 19–26, December 2007. [Online]. Available: http://doi.acm.org/10.1145/1361348.1361353

[28] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, 2007.

[29] A. Ikram, N. Baker, M. Knappmeyer, E. Reetz, and R. Tonjesy, "An artificial chemistry based framework for personal and social context aware smart spaces," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. IEEE, 2011, pp. 2009–2014.

[30] M. Knappmeyer, S. Kiani, C. Frá, B. Moltchanov, and N. Baker, "Contextml: A light-weight context representation and context management schema," in *In Proc. IEEE International Symposium on Wireless Pervasive Computing*, May 2010, pp. 367 – 372.

[31] Ubiquitous Web Applications Working Group, "Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies 2.0," World Wide Web Consortium (W3C), W3C Working Draft, April 2007. [Online]. Available: http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/

[32] K. Henricksen and J. Indulska, "A software engineering framework for context-aware pervasive computing," in *Pervasive Computing and Communications, 2004. PerCom 2004. Proc. Second IEEE Annual Conference on*. IEEE Computer Society, Mar 2004, pp. 77 – 86.

[33] J. E. Bardram, "The java context awareness framework (JCAF): a service infrastructure and programming framework for context-aware applications," in *Proc. Third international conference on Pervasive Computing*, ser. PERVASIVE'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 98–115. [Online]. Available: http://dx.doi.org/10.1007/11428572_7

[34] X. Wang, D. Zhang, T. Gu, and H. Pung, "Ontology based context modeling and reasoning using owl," in *Pervasive Computing and Communications Workshops, 2004. Proc. Second IEEE Annual Conference on*, March 2004, pp. 18 – 22.

[35] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *J. Network and Computer Applications*, vol. 28, pp. 1–18, January 2005.

[36] Q. Sheng and B. Benatallah, "Contextuml: a uml-based modeling language for model-driven development of context-aware web services," in *Mobile Business, 2005. ICMB 2005. International Conference on*, W. Brookes, E. Lawrence, R. Steele, and E. Chang, Eds. IEEE Computer Society, July 2005, pp. 206 – 212.

[37] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger, "Context-awareness on mobile devices - the hydrogen approach," in *Proc. 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9*, ser. HICSS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 292.1–. [Online]. Available: http://dl.acm.org/citation.cfm?id=820756.821849

[38] D. Brickley and R. Guha, "Rdf vocabulary description language 1.0: Rdf schema," W3C, W3C Recommendation, Feb 2004. [Online]. Available: http://www.w3.org/TR/rdf-schema

[39] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL web ontology language reference," World Wide Web Consortium (W3C), W3C Recommendation, February 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/

[40] P. Korpipää and J. Mäntyjärvi, "An ontology for mobile device sensor-based context awareness," in *Proc. 4th international and interdisciplinary conference on Modeling and using context*, ser. CONTEXT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 451–458. [Online]. Available: http://dl.acm.org/citation.cfm?id=1763142.1763181

[41] A. Ranganathan, J. Al-Muhtadi, and R. Campbell, "Reasoning about uncertain contexts in pervasive computing environments," *IEEE Pervasive Computing*, vol. 3, no. 2, pp. 62–70, 2004.

[42] H. Chang, S. Shin, and C. Chung, "Context Life Cycle Management Scheme in Ubiquitous Computing Environments," in *Mobile Data Management, 2007 International Conference on*. IEEE Computer Society, May 2007, pp. 315–319.

[43] H. Truong and S. Dustdar, "A survey on context-aware web service systems," *International Journal of Web Information Systems*, vol. 5, no. 1, pp. 5–31, 2009.

[44] J. Hong, E. Suh, and S. Kim, "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8509–8522, 2009.

[45] T. Winograd, "Architectures for context," *Human-Computer Interaction*, vol. 16, no. 2, pp. 401–419, 2001.

[46] P. Makris, D. Skoutas, and C. Skianis, "A survey on context-aware mobile and wireless networking: On networking and computing environments' integration," *IEEE Commun. Surveys Tutorials*, 2012.

[47] C.-F. Sørensen, M. Wu, T. Sivaharan, G. S. Blair, P. Okanda, A. Friday, and H. Duran-Limon, "A context-aware middleware for applications in mobile ad hoc environments," in *Proc. 2nd workshop on Middleware for pervasive and ad-hoc computing*, ser. MPAC '04. New York, NY, USA: ACM, 2004, pp. 107–110. [Online]. Available: http://doi.acm.org/10.1145/1028509.1028510

[48] A. K. Dey, D. Salber, M. Futakawa, and G. D. Abowd, "An architecture to support context-aware applications," Georgia Institute of Technology, Technical Report GIT-GVU-99-23, 1999. [Online]. Available: http://smartech.gatech.edu/handle/1853/3390

[49] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: a mobile context-aware tour guide," *Wireless Networks*, vol. 3, no. 5, pp. 421–433, Oct 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1019194325861

[50] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Trans. Information Systems*, vol. 10, no. 1, pp. 91–102, Jan 1992.

[51] P. Fahy and S. Clarke, "Cass-middleware for mobile context-aware applications," in *Workshop on Context Awareness, The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Boston, Massachusetts: ACM SIGMOBILE, June 2004, pp. 304–308.

[52] B. Guo, D. Zhang, and M. Imai, "Toward a cooperative programming framework for context-aware applications," *Personal and Ubiquitous Computing*, vol. 15, no. 3, pp. 221–233, 2011.

[53] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. Campbell, and K. Nahrstedt, "A middleware infrastructure for active spaces," *IEEE Pervasive Computing*, vol. 1, no. 4, pp. 74 – 83, OIct-Dec 2002.

[54] M. Knappmeyer, N. Baker, S. Kiani, and R. Tönjes, "A context provisioning framework to support pervasive and ubiquitous applications," in *Proc. 4th European conference on Smart sensing and context*, ser. EuroSSC'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 93–106.

[55] D. Athanasopoulos, A. Zarras, V. Issarny, E. Pitoura, and P. Vassiliadis, "CoWSAMI: Interface-aware context gathering in ambient intelligence environments," *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 360–389, 2008.

[56] J. Zhu, H. Pung, M. Oliya, and W. Wong, "A context realization framework for ubiquitous applications with runtime support," *Communications Magazine, IEEE*, vol. 49, no. 9, pp. 132–141, 2011.

[57] N. Davies, K. Cheverst, K. Mitchell, and A. Friday, "'Caches in the Air': Disseminating tourist information in the guide system," in *Proc. Second IEEE Workshop on Mobile Computer Systems and Applications*, ser. WMCSA '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 11–. [Online]. Available: http://dl.acm.org/citation.cfm?id=520551.837524

[58] R. Kernchen, D. Bonnefoy, A. Battestini, B. Mrohs, M. Wagner, and M. Klemettinen, "Context-awareness in mobilife," in *Proc. 15th IST Mobile Summit*. Mykonos, Greece: IST Mobile Summit, June 2006.

[59] T. Gu, H. Pung, and D. Zhang, "Toward an OSGi-based infrastructure for context-aware applications," *IEEE Pervasive Computing*, vol. 3, no. 4, pp. 66–74, 2005.

[60] Z. Yu, X. Zhou, Z. Yu, D. Zhang, and C.-Y. Chin, "An osgi-based infrastructure for context-aware multimedia services," *IEEE Commun. Mag.*, vol. 44, no. 10, pp. 136 –142, Oct 2006.

[61] V. Lesser, "Cooperative multiagent systems: A personal view of the state of the art," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 1, pp. 133–142, 2002.

[62] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive computing environments," *The Knowledge Engineering Review*, vol. 18, no. 03, pp. 197–207, 2003.

[63] C. Julien and G. Roman, "Egospaces: Facilitating rapid development of context-aware mobile applications," *IEEE Trans. Softw. Eng.*, vol. 32, no. 5, pp. 281–298, May 2006.

[64] I. Chen, S. Yang, and J. Zhang, "Ubiquitous provision of context aware web services," in *Services Computing (SCC '06), IEEE International Conference on*. Chicago, IL: IEEE Computer Society, Sept 2006, pp. 60–68.

[65] H.-L. Truong, L. Juszczyk, A. Manzoor, and S. Dustdar, "Escape – an adaptive framework for managing and providing context information in emergency situations," in *Smart Sensing and Context*, ser. Lecture Notes in Computer Science, G. Kortuem, J. Finney, R. Lea, and V. Sundramoorthy, Eds. Springer Berlin / Heidelberg, 2007, vol. 4793, pp. 207–222. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75696-5_13

[66] H.-L. Truong, S. Dustdar, D. Baggio, S. Corlosquet, C. Dorn, G. Giuliani, R. Gombotz, Y. Hong, P. Kendal, C. Melchiorre, S. Moretzky, S. Peray, A. Polleres, S. Reiff-Marganiec, D. Schall, S. Stringa, M. Tilly, and H. Yu, "incontext: A pervasive and collaborative working environment for emerging team forms," in *Applications and the Internet (SAINT 2008), International Symposium on*. IEEE Computer Society, Aug 2008, pp. 118 –125.

[67] D. de Almeida, C. de Souza Baptista, E. da Silva, C. Campelo, H. de Figueiredo, and Y. Lacerda, "A context-aware system based on service-oriented architecture," in *Advanced Information Networking and Applications (AINA '06), 20th International Conference on*, vol. 1. IEEE Computer Society, Apr 2006, p. 6.

[68] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[69] R. Greiner, C. Darken, and N. I. Santoso, "Efficient reasoning," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 1–30, Mar. 2001. [Online]. Available: http://doi.acm.org/10.1145/375360.375363

[70] S. Russell, P. Norvig, J. Canny, J. Malik, and D. Edwards, *Artificial intelligence: a modern approach*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, Dec 2009.

[71] C. Anagnostopoulos, A. Tsounis, and S. Hadjiefthymiades, "Context awareness in mobile computing environments," *Wireless Personal Communications*, vol. 42, pp. 445–464, 2007, 10.1007/s11277-006-9187-6. [Online]. Available: http://dx.doi.org/10.1007/s11277-006-9187-6

[72] E. Kim, S. Helal, and D. Cook, "Human activity recognition and pattern discovery," *IEEE Pervasive Computing*, vol. 9, no. 1, pp. 48–53, 2010. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2010.7

[73] A. Goultiaeva and Y. Lespérance, "Incremental plan recognition in an agent programming framework," in *Working Notes of the AAAI Workshop on Plan, Activity, and Intention Recognition (PAIR)*, Vancouver, BC, July 2007.

[74] P. Korpipää, M. Koskinen, J. Peltola, S.-M. Mäkelä, and T. Seppänen, "Bayesian approach to sensor-based context awareness," *Personal Ubiquitous Comput.*, vol. 7, no. 2, pp. 113–124, Jul. 2003. [Online]. Available: http://dx.doi.org/10.1007/s00779-003-0237-8

[75] P. Nurmi, P. Floréen, M. Przybilski, and G. Lindén, "A framework for distributed activity recognition in ubiquitous systems," in *Proceedings International Conference on Artificial Intelligence (ICAI)*, vol. 1. Las Vegas, Nevada, USA: CSREA Press, June 2005, pp. 650–655.

[76] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993. [Online]. Available: http://dx.doi.org/10.1006/knac.1993.1008

[77] M. Schmidt-Schauß and G. Smolka, "Attributive concept descriptions with complements," *Artificial intelligence*, vol. 48, no. 1, pp. 1–26, 1991.

[78] D. Ejigu, M. Scuturici, and L. Brunie, "Semantic approach to context management and reasoning in ubiquitous context-aware systems," in *Digital Information Management (ICDIM'07), 2nd International Conference on*, vol. 1. Lyon: IEEE, Oct 2007, pp. 500–505. [Online]. Available: http://dx.doi.org/10.1109/ICDIM.2007.4444272

[79] R. Klinger and K. Tomanek, "Classical probabilistic models and conditional random fields," Technische Universität Dortmund, Dortmund, Germany, Algorithm Engineering Report TR07-2-013, Dec 2007.

[80] D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science, C. Nédellec and C. Rouveirol, Eds. Springer Berlin / Heidelberg, 1998, vol. 1398, pp. 4–15.

[81] L. Rabiner and B. Juang, "An introduction to hidden markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, 1986.

[82] I. Csiszár, "MaxEnt, mathematics, and information theory," *Maximum entropy and Bayesian methods*, pp. 35–50, 1996.

[83] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[84] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *Proc. Fourth IEEE International Conference on Multimodal Interfaces*. IEEE Computer Society, 2002, pp. 3–8. [Online]. Available: http://dx.doi.org/10.1109/ICMI.2002.1166960

[85] P. Turaga, R. Chellappa, V. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1473–1488, Nov 2008. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2008.2005594

[86] D. Wen-Yu, X. Ke, and L. Meng-Xiang, "A Situation Calculus-based Approach To Model Ubiquitous Information Services," *Arxiv preprint cs/0311052*, 2003.

[87] F. Pirri and R. Reiter, "Some contributions to the metatheory of the situation calculus," *J. ACM (JACM)*, vol. 46, no. 3, pp. 325–361, May 1999. [Online]. Available: http://doi.acm.org/10.1145/316542.316545

[88] C. B. Anagnostopoulos, P. Pasias, and S. Hadjiefthymiades, "A framework for imprecise context reasoning," in *IEEE International Conference on Pervasive Services*. IEEE Computer Society, 15-20 July 2007, pp. 181–184. [Online]. Available: http://dx.doi.org/10.1109/PERSER.2007.4283913

[89] J. Yin, Q. Yang, D. Shen, and Z. Li, "Activity recognition via user-trace segmentation," *ACM Trans. Sensor Networks (TOSN)*, vol. 4, no. 4, pp. 1–34, 2008.

[90] T. Bosse, F. Both, C. Gerritsen, M. Hoogendoorn, and J. Treur, "Model-based reasoning methods within an ambient intelligent agent model," in *Constructing Ambient Intelligence*, ser. Communications in Computer and Information Science, M. Mühlhäuser, A. Ferscha, and E. Aitenbichler, Eds. Springer Berlin Heidelberg, 2008, vol. 11, pp. 352–370. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85379-4_40

[91] F. Both, M. Hoogendoorn, and J. Treur, "Model-based ambient analysis of human task execution," in *Proc. 1st international conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '08. New York, NY, USA: ACM, 2008, pp. 92:1–92:8. [Online]. Available: http://doi.acm.org/10.1145/1389586.1389690

[92] L. Goix, M. Valla, L. Cerami, and P. Falcarin, "Situation inference for mobile users: a rule based approach," in *Mobile Data Management (MDM '07), International Conference on*. Mannheim, Germany: IEEE, May 2007, pp. 299–303. [Online]. Available: http://dx.doi.org/10.1109/MDM.2007.63

[93] J. Serrano, J. Serrat, and A. Galis, "Ontology-based context information modelling for managing pervasive applications," in *Autonomic and*

Autonomous Systems (ICAS '06), International Conference on, P. Dini, D. Ayed, C. Dini, and Y. Berbers, Eds. California, USA: IEEE Computer Society, 19-21 July 2006.

[94] T. Gu, X. Wang, H. Pung, and D. Zhang, "An ontology-based context model in intelligent environments," in *Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference*. San Diego, California, USA: The Society for Modeling and Simulation International (SCS), 18–21 Jan 2004, pp. 270–275.

[95] E. Christopoulou, C. Goumopoulos, and A. Kameas, "An ontology-based context management and reasoning process for UbiComp applications," in *Proc. 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services, Usages and Technologies*, ser. sOc-EUSAI '05, ACM. New York, NY, USA: ACM, 2005, pp. 265–270. [Online]. Available: http://doi.acm.org/10.1145/1107548.1107613

[96] M. Knappmeyer, E. Wittkorn, S. Kiani, R. Tönjes, and N. Baker, "Context provisioning middleware with probabilistic reasoning support," in *Proc. 20th Future Network and Mobile Summit*, Warsaw, Poland, 2011.

[97] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *Proc. 6th international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '07. New York, NY, USA: ACM, 2007, pp. 235:1–235:8. [Online]. Available: http://doi.acm.org/10.1145/1329125.1329409

[98] J. Scholtz and S. Consolvo, "Towards a discipline for evaluating ubiquitous computing applications," Intel Research, Tech. Rep. IRS-TR-04-004, Jan 2004. [Online]. Available: http://www.seattle.intel-research.net/pubs/022520041200_232.pdf

[99] Y. Rogers, K. Connelly, L. Tedesco, W. Hazlewood, A. Kurtz, R. E. Hall, J. Hursey, and T. Toscos, "Why it's worth the hassle: the value of in-situ studies when designing ubicomp," in *Proc. 9th international conference on Ubiquitous computing*, ser. UbiComp '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 336–353. [Online]. Available: http://dl.acm.org/citation.cfm?id=1771592.1771612

[100] K. Connelly, K. Siek, I. Mulder, S. Neely, G. Stevenson, and C. Kray, "Evaluating pervasive and ubiquitous systems," *Pervasive Computing, IEEE*, vol. 7, no. 3, pp. 85 –88, july-sept. 2008.

[101] H. Yan and T. Selker, "Context-aware office assistant," in *Proceedings of the 5th international conference on Intelligent user interfaces*, ser. IUI '00. New York, NY, USA: ACM, 2000, pp. 276–279. [Online]. Available: http://doi.acm.org/10.1145/325737.325872

[102] A. K. Dey and G. D. Abowd, "Cybreminder: A context-aware system for supporting reminders," in *Proc. 2nd international symposium on Handheld and Ubiquitous Computing*, ser. HUC '00. London, UK: Springer-Verlag, 2000, pp. 172–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=647986.757284

[103] S. Intille, E. Tapia, J. Rondoni, J. Beaudin, C. Kukla, S. Agarwal, L. Bao, and K. Larson, "Tools for studying behavior and technology in natural settings," in *UbiComp 2003: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, A. Dey, A. Schmidt, and J. McCarthy, Eds. Springer Berlin / Heidelberg, 2003, vol. 2864, pp. 157–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-39653-6_13

[104] S. Consolvo, L. Arnstein, and B. R. Franza, "User study techniques in the design and evaluation of a ubicomp environment," in *Proc. the 4th international conference on Ubiquitous Computing*, ser. UbiComp '02. London, UK: Springer-Verlag, 2002, pp. 73–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=647988.741490

[105] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, "Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones," in *Proc. 5th international conference on Mobile systems, applications and services*, ser. MobiSys '07. New York, NY, USA: ACM, 2007, pp. 57–70. [Online]. Available: http://doi.acm.org/10.1145/1247660.1247670

[106] S. Consolvo and M. Walker, "Using the experience sampling method to evaluate ubicomp applications," *Pervasive Computing, IEEE*, vol. 2, no. 2, pp. 24 – 31, Apr-Jun 2003.

[107] A. Crabtree, S. Benford, C. Greenhalgh, P. Tennent, M. Chalmers, and B. Brown, "Supporting ethnographic studies of ubiquitous computing in the wild," in *Proc. 6th conference on Designing Interactive systems*, ser. DIS '06. New York, NY, USA: ACM, 2006, pp. 60–69. [Online]. Available: http://doi.acm.org/10.1145/1142405.1142417

[108] S. S. Intille, L. Bao, E. M. Tapia, and J. Rondoni, "Acquiring in situ training data for context-aware ubiquitous computing applications," in *Proc. SIGCHI conference on Human factors in computing systems*, ser. CHI '04. New York, NY, USA: ACM, 2004, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/985692.985693

[109] J. Zacks and B. Tversky, "Event structure in perception and conception," *Psychological Bulletin*, vol. 127, no. 1, pp. 3–21, 2001.

[110] Y. Li and J. A. Landay, "Activity-based prototyping of ubicomp applications for long-lived, everyday human activities," in *Proc. the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1303–1312. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357259

[111] S. S. Intille, J. Rondoni, C. Kukla, I. Ancona, and L. Bao, "A context-aware experience sampling tool," in *CHI '03 extended abstracts on Human factors in computing systems*, ser. CHI EA '03. New York, NY, USA: ACM, 2003, pp. 972–973. [Online]. Available: http://doi.acm.org/10.1145/765891.766101

[112] S. Carter, J. Mankoff, and J. Heer, "Momento: support for situated ubicomp experimentation," in *Proc. SIGCHI conference on Human factors in computing systems*, ser. CHI '07. New York, NY, USA: ACM, 2007, pp. 125–134. [Online]. Available: http://doi.acm.org/10.1145/1240624.1240644

[113] E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, and D. Pesch, "A testbed for evaluating human interaction with ubiquitous computing environments," in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom) 2005.*, ser. TRIDENTCOM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 60–69. [Online]. Available: http://dx.doi.org/10.1109/TRIDNT.2005.7

[114] S. S. Intille, K. Larson, E. M. Tapia, J. S. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, "Using a live-in laboratory for ubiquitous computing research," in *Proc. 4th international conference on Pervasive Computing*, ser. PERVASIVE'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 349–365. [Online]. Available: http://dx.doi.org/10.1007/11748625_22

[115] S. Hossein, S. Helal, and A. Mendez-Vasquez, *Sensory Dataset Description Language (SDDL) Specification*, Mobile and Pervasive Computing Laboratory Department of Computer and Information Science and Engineering Std., Rev. 1.0, April 2009. [Online]. Available: http://www.icta.ufl.edu/persim/sddl

[116] A. M. Law and W. D. Kelton, *Simulation Modelling and Analysis*, 3rd ed. Mcgraw-Hill Higher Education, Dec. 1999.

[117] J. Barton and V. Vijayaraghavan, "UBIWISE, a simulator for ubiquitous computing systems design," Hewlett-Packard Labs, Palo Alto, Tech. Rep. HPL-2003-93, 2003. [Online]. Available: http://www.hpl.hp.com/techreports/2003/HPL-2003-93.pdf

[118] R. Morla and N. Davies, "Evaluating a location-based application: A hybrid test and simulation environment," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 48–56, Jul-Sept 2004.

[119] S. Jang, Y. Lee, W. Woo, G. U. vr Lab, and S. Korea, "CIVE: Context-based interactive system for distributed virtual environment," in *The 14th International Conference on Artificial Reality and Telexistence (ICAT 2004)*, Seoul, S. Korea, 30 Nov – 02 Dec 2004, pp. 495–498.

[120] H. Nishikawa, S. Yamamoto, M. Tamai, K. Nishigaki, T. Kitani, N. Shibata, K. Yasumoto, and M. Ito, "Ubireal: Realistic smartspace simulator for systematic testing," in *UbiComp 2006: Ubiquitous Computing*, ser. Lecture Notes in Computer Science, P. Dourish and A. Friday, Eds. Springer Berlin / Heidelberg, 2006, vol. 4206, pp. 459–476. [Online]. Available: http://dx.doi.org/10.1007/11853565_27

[121] I. Kim, H. Park, Y. Lee, H. Lee, and B. Noh, "Design of context-awareness simulation toolkit for ubiquitous computing," in *Industrial Electronics, 2006 IEEE International Symposium on*, vol. 4, July 2006, pp. 3220 –3225.

[122] J. Barbosa, R. Hahn, D. Bonatto, F. Cecin, and C. Geyer, "Evaluation of a large-scale ubiquitous system model through peer-to-peer protocol simulation," in *Distributed Simulation and Real-Time Applications (DS-RT 2007), 11th IEEE International Symposium*, Oct 2007, pp. 175–181.

[123] M. Martin and P. Nurmi, "A generic large scale simulator for ubiquitous computing," in *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*. IEEE, 2006, pp. 1–3.

[124] E. Reetz, M. Knappmeyer, S. Kiani, A. Anjum, N. Bessis, and R. Tönjes, "Performance simulation of a context provisioning middleware based on empirical measurements," *Simulation Modelling Practice and Theory*, 2012.

[125] A. K. Dey, D. Salber, G. D. Abowd, and M. Futakawa, "The conference assistant: Combining context-awareness with wearable computing," in *Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, ser. ISWC '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 21–. [Online]. Available: http://dl.acm.org/citation.cfm?id=519309.856496

[126] X. Gong, T. Chandrashekhar, J. Zhang, and H. Poor, "Opportunistic cooperative networking: To relay or not to relay?" *Selected Areas*

in Communications, IEEE Journal on, vol. 30, no. 2, pp. 307 –314, february 2012.

[127] L. Sanchez, J. Galache, V. Gutierrez, J. Hernandez, J. Bernat, A. Gluhak, and T. Garcia, "Smartsantander: The meeting point between future internet research and experimentation and the smart cities," in *Future Network Mobile Summit (FutureNetw), 2011*, june 2011, pp. 1 –8.

[128] S. Kiani, A. Anjum, N. Antonopoulos, and M. Knappmeyer, "Context-aware service utilisation in the clouds and energy conservation," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–21, 2012.

[129] G. D. Abowd, "What next, ubicomp?: celebrating an intellectual disappearing act," in *Proc. 2012 ACM Conference on Ubiquitous Computing*, ser. UbiComp '12. New York, NY, USA: ACM, 2012, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/2370216.2370222

[130] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, 1993.

Dr. Michael Knappmeyer received his Ph.D. (Computer Science) from the University of the West of England, Bristol, UK, in 2012. His thesis presented a Context Provisioning Middleware with Support for Evolving Awareness. In 2006 he received the Diplom-Informatiker degree from the University of Applied Sciences Osnabrück, Germany. As research associate Michael participated in the European research projects "C-MOBILE" and "Context Casting (C-CAST)". In the latter he led the context reasoning activity and contributed to the overall system architecture. His interests include smart spaces, context modelling, reasoning and mobile device management.

Dr. Saad Liaquat Kiani received his B.E. from the National University of Sciences and Technology, Islamabad, Pakistan, in 2003. He received his M.S. in Computer Engineering from Kyung Hee University, South Korea, in 2007 and completed his Ph.D. in Computer Science at the University of the West of England, Bristol, UK, in 2011. He is currently a Senior Lecturer in Networks and Mobile Computing at the University of the West of England. He is also a visiting lecturer at Cardiff University's School of Computer Science and Informatics. His research interests are in the areas of mobile and distributed computing, context-aware systems and participatory sensing.

Eike Steffen Reetz has studied Electrical Engineering with focus on communication technology at the University of Applied Sciences Osnabrück (UASO) and received his "Diplom Ingenieur (FH)" degree in August 2007. He is currently working as a Research Assistant at the Mobile Communication Research Group at UASO. His current research interests include the evaluation of context provisioning systems as well as testing of context aware services. He participated in the European research projects "C-MOBILE" and "Context Casting (C-CAST)" and is currently involved in the European research project "IoT.est". Eike is working towards his Ph.D. in Electrical Engineering with the research scope of evaluation and testing of IoT-aware services.

Nigel Baker led the Mobile & Ubiquitous Systems Group, was the Co-Director of CCCS research centre and Associate Professor (Reader) in Computer Science at the University of the West of England until 2011. His first degrees were in Physics and Nuclear & Particle Physics. His specialisms in the last twenty years have been Real Time Systems, Computer Networks, Distributed Systems and in the last decade Mobile Communications. He was a visiting researcher at CERN, Geneva for six years. He was also a Motorola Fellow until 2006 through which he developed and led the Mobile Applications of Software Technologies (MAST) Programme whilst a recipient of a Royal Academy of Engineering Industrial Fellowship award with Motorola European Cellular Infrastructure Division (ECID), Swindon. During this collaboration with Motorola he worked on several EU 5th, 6th and 7th Framework projects.

Prof. Dr.-Ing. Ralf Tönjes is heading the mobile communications group at the University of Applied Sciences Osnabrück. He studied communication engineering at the University of Hannover and biomedical engineering at the University of Strathclyde in Glasgow. In 1998 he received his Dr.-Ing. degree (summa cum laude) in electrical engineering from the University of Hannover. From 1998 to 2005 he was with Ericsson Research, working on future mobile networks and representing Ericsson in standardisation. In 2005 Ralf Tönjes joined the University of Applied Sciences of Osnabrück as full professor for Mobile Communications. He is a TPC member of several international conferences and (co-) authored more than seventy scientific publications. His current research interests include wireless communication networks, Internet of things, context-aware service platforms and test automation.