CCF Transactions on Pervasive Computing and Interaction
https://doi.org/10.1007/s42486-019-00005-2

REGULAR PAPER

Pervasive computing middleware: current trends and emerging challenges

Christian Becker[1] · Christine Julien[2] · Philippe Lalanda[3] · Franco Zambonelli[4]

Received: 3 October 2018 / Accepted: 27 January 2019

Abstract
Driven by the increasing diffusion of embedded sensors and actuators, and more in general by "Internet of Things" (IoT) devices, pervasive computing is becoming a reality. Yet, most actual implementations of pervasive computing environments rely on rather centralized architectures and on middleware solutions that integrate only the minimal set of services to enable interoperabilty and data integration. In this article, after having overviewed the state of the art in the area of pervasive computing middleware, we discuss the many challenges that still have to be faced for pervasive computing middleware to be able to support elastic, easy to configure, easy to develop, safe, and ethically acceptable, pervasive computing services and applications.

Keywords Pervasive computing · System software · Middleware

1 Introduction

Pervasive computing, as common in "Internet of Things" scenarios (Atzori et al. 2010), promotes the integration of connected electronic devices in our living spaces in order to assist us in our daily activities, be they professional or private. These pervasive devices can be blended in the environment, integrated in smartphones or into everyday objects and appliances, or even woven into clothing. They are mobile or static, can take multiple forms, and pick up a wide variety of signals from the environment. Pervasive devices collect contextual information, run local computation, and, in some cases, directly act upon the environment. This allows the implementation of simple, reactive services like opening an emergency door when a triggering condition is detected. Pervasive devices are also enhanced with networking capabilities so that they can communicate with each other or with more powerful computing elements, located in close proximity (i.e., in the "fog" Bonomi et al. 2012), or more distant (i.e., in the "cloud"). An elastic use of a mixture of device-to-device, fog, and cloud coordination is necessary to implement more complex services that integrate multiple data sources, must be responsive at human time scales, but may demand significant computing and memory capacities.

In the realm of pervasive computing and the Internet of Things, much progress has been made since initial research challenges were posited (Atzori et al. 2010), and challenges more specific to the convergence of the cyber and the physical of pervasive computing have been enumerated (Conti et al. 2012). However, given the many bold visions of pervasive computing applications, the services available today to end users are still quite limited in terms of performance, cost, security, adaptivity, and even basic functionality.

In this paper, we motivate a simple yet bold claim: while much research has made it possible to *connect* pervasive computing elements and let them interoperate to provide simple services, we still lack facilities to dynamically combine elements and allow them to *adaptively coordinate* to provide more flexible (and more useful) services, while also

✉ Christian Becker
christian.becker@uni-mannheim.de

Christine Julien
c.julien@utexas.edu

Philippe Lalanda
philippe.lalanda@univ-grenoble-alpes.fr

Franco Zambonelli
franco.zambonelli@unimore.it

[1] Universität Mannheim, Mannheim, Germany

[2] University of Texas at Austin, Austin, USA

[3] Univ. Grenoble, Grenoble, France

[4] Universita di Modena e Reggio Emilia, Modena, Italy

Published online: 19 February 2019

dynamically accounting for the needs of the *human inhabitants* of pervasive computing environments.

Indeed, many protocols exist to address the communication element of transmitting data from device to device (Asadi et al. 2014; Bello and Zeadally 2014; Cho and Julien 2016; Choi and Han 2015), from a device to the fog (Bonomi et al. 2012; Golrezaei et al. 2012; Satyanarayanan et al. 2009; Verbelen et al. 2012), or from a device to the cloud. Similarly, protocols exist for discovering the available resources surrounding a user in a pervasive computing environment (Amadeo et al. 2014; Gu et al. 2005; Guo et al. 2013; Jenson et al. 2014; Lin et al. 2014; Liu et al. 2014; Mayer et al. 2014; Quevedo et al. 2016; Wehner et al. 2014), and for combining them according to some static service composition patterns (Escoffier et al. 2007; Wehner et al. 2014). However, creating larger more general-purpose solutions out of these connected elements still founders in the face of complexity, dynamism, and the need to adapt to physical and social contexts.

Yet it is exactly such combinations that will fulfill the broader vision of pervasive computing. What are now small devices capable of simply sending sensor streams to a cloud database or receiving simple actuation commands will soon become highly intelligent and integrated embedded *systems* capable of autonomous decisions and able to "speak" to one another at a high level. They will "argue" and "arbitrate" myriad situations while negotiating how to cooperatively (or competitively) achieve their goals (on behalf of their human users) (Lippi et al. 2018). In this context, users will not simply "invoke" services by pushing some button or by launching some app that will trigger a composition of devices' functionalities according to some static design pattern. Rather, users will be able to dialogue with devices and smart environments and will be able to dynamically direct the coordination of such devices in order to satisfy needs. Pervasive computing devices and applications, by their side, will be able to make a better use of the available computing facilities, by dynamically re-arranging their configuration (such as the way code is distributed over devices, edge computers, and cloud servers, and the way in which cooperation among these architectural layers happens) in order to meet new challenging requirements related to changed patterns of usage or changed users' requirements, privacy and security, etc.

In realizing such a vision, middleware is an integral part. Middleware will provide the abstractions necessary to achieve high-level goals while also providing the concrete realizations of the nuts and bolts required to make it all happen. This paper focuses on precisely this integral role of middleware in realizing the long vision of pervasive computing. We frame the discussion by starting with a set of "hot" applications, which, while seemingly achievable today, are as yet unrealized. We then explicitly connect this lack of available implementations to gaps in middleware solutions for pervasive computing. Section 3 takes a deep dive into the state of the art; much research exists that relates to our vision of advanced middleware for pervasive computing. Yet our analysis of the state of the art clearly shows that there remain several open challenges in filling the identified gaps. Accordingly, Sect. 4 enumerates and details the key remaining open challenges that must be addressed to realize our vision.

2 Identifying the gap

In this section, we examine some particularly "hot" application domains and use them to frame the "gaps" that remains to be filled by pervasive computing middleware.

2.1 Hot application domains and paradigm gaps

The proliferation of pervasive computing devices, coupled with the widespread availability of the Internet, makes pervasive computing more concrete every day. Whether at home, in commute, or at work, we already enjoy a variety of simple, unobtrusive services that enhance our quality of life or allow us to optimize resource management. These new technologies have, and will continue to have, profound effects on entire industries and society.

In the *manufacturing domain* the notion of Industry 4.0 is gaining increasing attention (Lalanda et al. 2017). The purpose of the Industry 4.0 initiative is to bring together new technologies and production processes to enable the emergence of smart, connected manufacturing. The term, coined in 2011 by a government-funded German project, refers to what could be the fourth industrial revolution. Industry 4.0 envisions new production techniques, new materials, and the generalized adoption of digital technologies. Our focus in this paper is on the latter point, and in particular on the adoption of dense and pervasive networks of sensors and actuators in industrial environments (including robotic systems). This constitutes a tremendous challenge, in that it can promote the seamless integration of field devices controlling operations on a plant floor and supervision systems, usually located in IT facilities. Among many benefits, such integration should allow the systematic oversight and improvement of production activities and resource management and an overall increase in safety and sustainability of production systems.

However, given the dynamics of the modern economy, it is important that these new technologies also open the way for more flexibility in production processes and that the overall pervasive infrastructure can be easily (if not automatically) and safely reconfigured to meet changing demands and enable novel products.

Similar ideas and challenges also apply in the popular domains of smart homes, smart buildings, and more in general smart cities. The *smart home domain* is an interesting and highly symbolic example that touches our daily lives (Helal et al. 2005). A smart home is filled with electronic devices providing multiple, unobtrusive services to its inhabitants. Although most existing homes were not designed to be smart, the availability of small wireless devices enables the seamless instrumentation of home environments with pervasive sensing and actuating capabilities and thus makes it possible to realize services to monitor and control ambient conditions and nearly every appliance in our homes.

Clearly, such home environments are not expected to be inhabited by skilled administrators, and therefore novel pervasive devices and services must be extremely simple to use and install. They should not require the intervention of expert programmers to configure, update, and retire. Therefore, on the one hand users should be given a way to be in control of the configurations and activities of their home environments, while on the other hand such environments should be made somewhat "autonomic" in nature, i.e., capable of self-configuration and self-adaptation (VanSyckel et al. 2013).

Further, quality expectations are very high in our homes. Services must be secure, robust, tailored to inhabitants, and highly relevant. Keeping humans in the loop is crucial for social adoption of new technologies in our most private environments. There is nothing more annoying than shutters moving down for no apparent reason—and a great reason to give up on technology! Life at home must remain easy, calm, and predictable (Keith Edwards et al. 2001).

At a scale, very similar consideration can be applied to *smart buildings* and *smart cities*. In these environments, we may reasonably assume the existence of organizations that provide skilled administrators. Instead, it is the inherent spatial distribution of the system that calls for simple to install and configure devices and services, so as to make their management economically bearable. In any case, since acceptability and predictability of the technology is not only a usability issue but a general political and democratic one (Zambonelli et al. 2018), it is still important that inhabitants and citizens are somehow given a way to easily understand the overall functioning of such environment, and some way of tuning the environment's behaviour according to the user's specific needs.

Industry 4.0, smart homes, and smart cities are just three out of many "hot" application domains for pervasive and IoT systems. Additional interesting application areas include smart education systems, agriculture, logistics, smart transport systems, food industries, and the food chain. Yet, the technical and social challenges discussed in this paper transcend the particular application domains. However, the above discussion enables identifying important gaps in how today's pervasive systems and services are conceived, and consequently in the functionalities provided by pervasive middleware. The dominant paradigm today is to conceive pervasive devices as loci of simple services, e.g., a service for uploading sensed data or one for executing a simple actuation command. As a consequence, building and configuring complex pervasive systems out of these devices requires defining low-level composition rules for basic services and middleware to support the execution of such composite services. End users are mostly ruled out by this process and are left little control over defining or modifying the rules. This also naturally impacts the paradigm of usage: end users today tend to exploit pervasive devices by exploiting some "app" as a means to invoke specific services; users have little or no means of configuring devices or of programming new composite services.

This dominant perspective will have to change. First, as we have outlined in the discussion of the application areas, users have to be empowered with means to control and configure pervasive computing environments and services, other than by simply being given access to services. What are now "apps to invoke services" will have to become "handles to configure services". Second, in all the above application areas, what are now simple devices will soon become highly intelligent embedded devices with integrated autonomous decision logic. Simply consider the already emerging robotic assistants in smart homes or autonomous self-driving cars in smart cities. For these autonomous devices, providing composite services will not be simply a matter of composition, but a matter of distributed decision making and distributed agreement.

Such new characteristics must be explicitly addressed and enabled within pervasive computing middleware. Emerging middleware will need to facilitate on-device reasoning about high-level situations and focus on goal achievement and autonomy rather than assuming devices will blindly carry out simple commands. To support this autonomy, middleware will need to allow devices to interact directly, to share their views about the ambient situation, and to cooperatively reason to control it. The middleware will have to arbitrate fairness, conflict-freedom, and legal and ethical rules, as individual devices are empowered to make individual decisions. In short, the middleware will become a moderator of lively discussions among devices. Last but not least, *humans' voices* must be integrated in these conversations and decision making processes, in that the final decisions of the devices must respond to the humans' requirements. Further, the decisions made and actions taken by the devices must be understandable (even subconsciously) by the humans in the space. In a sentence, we expect pervasive computing middleware to support a paradigm shift from "pushing a button in an app" to "participating in discussions and decision making".

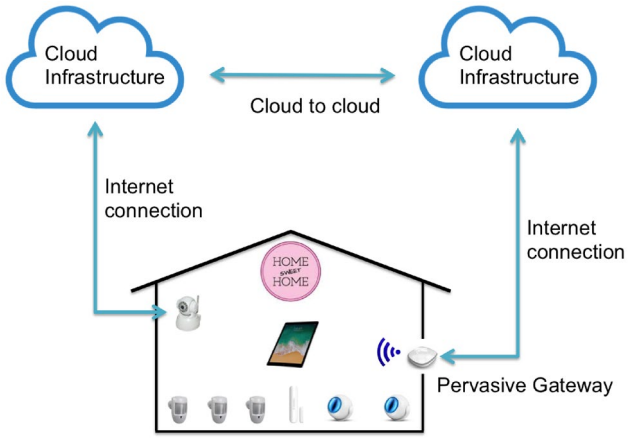

Fig. 1 Smart home services infrastructure

2.2 Architectural gaps

From the architectural viewpoint, most of today's industrial, home, and smart city services rely on cloud solutions. Data collected in the field is stored in centralized data centers and processed by powerful cloud servers deployed as necessary. As illustrated by Fig. 1 with reference to a smart home scenario, dedicated gateways act as intermediaries between physical environments (e.g., the home) and the cloud. These existing gateways are very simple: their role is to collect data from field devices, possibly perform simple pruning and mediation operations, and send relevant data to the cloud. In some cases, they also run simple, prescribed scenarios to coordinate devices' actions. Applications also rely on the gateways to reify actions to be taken in the physical space, as decided at the cloud level.

This architectural approach has many attractive aspects. The high-value data and analytics services identified and installed in the domains of smart buildings and homes are generally very greedy in terms of computing power and time. They are also based on large volumes of sensitive data that must be stored and accessed rapidly and easily. Cloud infrastructures provide the necessary facilities to run such complex services. They are known to offer great benefits in terms of computing power, elasticity, flexibility, pay-per-use facilities, and security.

A centralized cloud architecture also provides administration simplicity. Cloud providers are in charge of the management and control of the cloud infrastructure. Business service providers can focus on the administration of their own code, generally through virtualized gateways provided in the cloud. Most of the time, this is easy and fast. In contrast, managing business code running on field gateways is more complicated and time-demanding. In the telecommunications domain, the administration of gateways (e.g., Internet boxes, set-top-boxes, etc.) is generally delegated to teams of experts; these teams are often overwhelmed by maintenance and evolution requests. Such organization causes delays that are not in line with customers' expectations. A third major architectural benefit relates to integration. It is clearly much easier to adopt cloud-to-cloud integration (as illustrated by Fig. 1) to connect heterogeneous devices rather than implementing local integration. This is especially true when integrating devices that use new field buses or lack open APIs for integration. Cloud-to-cloud integration requires sharing data format and semantics but requires no additional tricky code since smart devices are already connected to and managed in a cloud.

As explained, cloud infrastructures and underlying organizations meet the requirements of complex analytics services. Today, however, emerging new services impose requirements that cannot be met by cloud-based architectures (Chiang and Zhang 2016; Shi et al. 2016). For instance, some services implement time-critical control loops that sense and act upon the environment. These services cannot be executed in the cloud due to unpredictable delays or insufficient bandwidth. Security also seriously challenges current architectures for several reasons. First, users are not comfortable with the idea of personal data being stored in clouds or data centers they do not trust. For instance, smart speakers connected to the cloud are not accepted by a growing crescendo of people concerned with eavesdropping. Further, the way cloud-based services are run raises issues. Cloud solutions for security rely on perimeter-based protection. If the perimeter is endangered, the common countermeasure is to take the system offline (Chiang and Zhang 2016). This causes service disruption in all the physical entities (e.g., homes, businesses) managed by the corrupted cloud. Finally, in economical and ecological terms, it does not appear opportune to transport and store huge amounts of data that could instead be processed and used in gateways located closer to data sources.

Executing services at the gateway level or even *in situ* is, however, very complex due to the dynamic, heterogeneous, and stochastic nature of the pervasive computing environments themselves. This is further complicated by the fact that gateways and *in situ* devices have limited resources that must be managed explicitly. Streamlining the production of fog-level services will require developers and system administrators to be equipped with new software engineering tools.

A common approach is to introduce an execution platform that provides a development model and a set of technical services. This can be done at the operating system level, like Φ-stack (Xu et al. 2017) for instance, or at a higher level. In the latter case, the term middleware is generally introduced.

Making a distinction between the execution platform and the hosted services lowers complexity in terms of code, debug, configuration, and administration operations. Decades of research in pervasive computing have led to many

solutions for individual components of such middleware. In the next section, we examine the state of the art in middleware for pervasive computing. A general takeaway is that, while we know how to build and connect individual solutions, it remains hard to *flexibly and adaptively combine* them. That is, we lack a fluid pervasive computing ecosystem that integrates these individual advances in support of envisioned applications. This motivates a need for middleware solutions that support *elastic pervasive computing*, embodying techniques that optimize across implementation options that include *in-situ* or on-device processing, fog integration, or off-loading entirely to the cloud. In integrating these options, middleware must still provide seamless interactive experiences, which may demand revisiting antique concepts like *graceful degradation*, *lazy transaction processing*, *prefetching* and other challenges addressed in classical distributed systems. Registries listing available services must be made more expressive to expose the qualitative ramifications of the myriad options without exposing developers (or users!) to the complexity of making an explicit decision.

3 State of the art

In this section we overview the state of the art and the current trends in pervasive computing middleware. Although there are several approaches and proposals that go in the directions of filling the previously identified gaps, they still exhibit several limitations and leave open several challenges that we elaborated upon in Sect. 4.

3.1 Pervasive computing platforms

As introduced before, software engineering principles and tools are needed to support the production and administration of pervasive computing applications. Today, most applications are built on top of specific platforms, or middleware, that provide a number of technical services to facilitate interaction (communication), context-awareness, adaptation, and self-awareness. Modern platforms also provide domain-specific languages, often embedded in existing popular languages like Java. Such an approach relieves programmers from tedious, hard-to-debug code and moves part of the complexity to the supporting platform.

Pervasive computing platforms can be complex and based on advanced architectures, as illustrated in Fig. 2.

Many platforms are now based on service-oriented computing, a compositional approach where applications are built through late composition of independent software elements, called services (Chollet et al. 2016; Papazoglou 2003). A service is characterized by the functions it provides. It is a software resource that is described and published by a provider in a service registry, sometimes

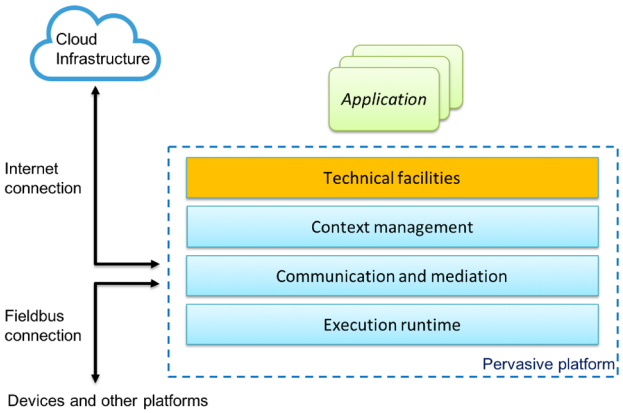

Fig. 2 Pervasive computing platform architecture

called a service broker. The registry acts as an intermediary between service providers and consumers. More precisely, service providers publish service descriptions in the registry. Service consumers can send queries to the registry to retrieve the available services meeting their requirements. Once a service has been selected, the consumer and provider can negotiate a contract specifying how the service is to be used. The next step, of course, is service invocation.

In the pervasive computing domain, service orientation promotes the development of modular, dynamic applications that can self-adapt to contextual evolution. Here, applications are built from loosely coupled services that can be distributed on different devices or computing nodes. Let us note also that the service-oriented approach offers excellent opportunities to achieve software application dynamism and is used more and more to build autonomic software systems (Lalanda et al. 2013).

Not surprisingly, an important number of service-oriented platforms related to pervasive computing applications have been developed over the years. Depending on their main field of application and the desired properties, various implementations of SOA principles have been proposed. In just the smart home domain, well known platforms like PCOM (Becker et al. 2004), iCasa (Escoffier et al. 2014), ubiSOAP (Caporuscio et al. 2012), SAI (Paganelli et al. 2010) nSOM (Familiar et al. 2012), AutoHome (Bourcier et al. 2011), DigiHome (Romero et al. 2013 or Microsoft's HomeOS (Dixon et al. 2012 propose different SOA implementations. While this level of activity is important for development, such diversity prevents services and applications developed on different middleware platforms from cooperating.

The service paradigm in itself is however not sufficient to easily build and manage pervasive computing applications. It has to be complemented, in a middleware or platform, by a number of technical services to deal with various forms

of communication, context modeling, data storage, deployment, or interoperability.

In the following, we structure our discussion along existing interaction models and service discovery mechanisms, followed by crucial technical services like mediation and context management. We then discuss deployment and configuration as well as adaptation support. We do not aim for a complete survey of the state of the art, rather we present exemplary approaches to aid our discussion of open challenges.

3.2 Interaction models

While many early projects did not build on top of middleware platforms but instead directly employed socket communication or direct communication with the hardware, with increasingly complex applications, interaction models have shifted. To support the growing demands of applications, a number of projects have emerged that explore different interaction models and supporting protocols.

Many proposals support traditional remote procedure call (RPC) or object-based interaction protocols. Such a choice enables adopting these widely assessed and understood models in the context of pervasive computing. Gaia (Román et al. 2002) is based on CORBA's Interoperability Protocol (IIOP) and thus provides an RPC based abstraction. IIOP is well documented and seamlessly enables the integration of existing backends and interoperability bridges for Gaia. Base (Becker et al. 2003) also provides application developers an RPC style programming abstraction. Its extensible microkernel provides means to map these abstractions to event-based or RPC based interoperability protocols. Finally, there are also approaches that use OSGi's (2007) object model and extend this by remoting, e.g., iPOJO (Escoffier et al. 2007) and P2PComp (Ferscha et al. 2004). The underlying object model results in RPC-like interaction protocols.

Several proposals also exist that extend the tuple space interaction protocol (as from the original proposal of the Linda language Ahuja et al. 1986), for their adoption in the support of interactions in pervasive computing systems. The adoption of the tuple space interaction model promotes mediated interactions between application components and services (in the form of putting or getting information from a tuple space) that can interact and synchronize even without knowing each other in advance. For instance, extending from one.world (Grimm 2004) is a tuple space based middleware built using Java objects. Similar to one.world, iROS (Johanson et al. 2002) is based on a tuple space based architecture, the *Event Heap*. Events are stored in the tuple space and age over time, allowing requests to gracefully expire if there is no recipient.

Extension of the tuple space model have also been conceived to more flexibly support interactions in the presence of mobility. LIME (Murphy et al. 2001), for instance, adopts a solution based on a multiplicity of mobile tuple spaces that can merge with each other depending on mobility patterns, thus enabling flexible dynamic event-based coordination across tuple spaces, in contrast to one.world and iROS where the tuple space is mostly for supporting persistent information. The TOTA middleware (Mamei and Zambonelli 2004) proposes a distributed middleware architecture based on a multiplicity of tuple spaces that can interact with each other in order to build distributed field-like structures supporting spatially-aware interactions between mobile devices and mobile services.

Again, this is not an exhaustive survey, but the diversity demonstrates that many interaction models have emerged for pervasive computing-like environments.

3.3 Discovery

Discovery is also a crucial service for any pervasive computing platform. Service discovery aims to find services for potential interaction. Different interaction models employ different service discovery patterns, though the patters also show some similarities. In particular, each interaction model uses advertisement messages and most systems use a lookup mechanism. However, the content of advertisement messages differs for different interaction models. For instance, event categories are advertised in a publish-subscribe model while service descriptions are advertised in the client-server and tuple space models. To counter this problem, event categories can be mapped to a service.

Industrial standards have also been established, e.g., Jini (Arnold et al. 1999) and UPNP (2016). While Jini provides a service oriented approach that is realized by a federation of lookup services, UPnP is a suite of protocols that can be combined in a flexible way to create discovery services. Obviously, pervasive computing environments must ensure that new services are found in a timely manner but should not spend too much energy on the discovery task; therefore related research has investigated aspects like energy management. Sandman (Schiele et al. 2004) is an example of a flexible discovery service that is mediator-based and allows to scheduled wake-up times of service providing nodes in order to save energy.

The state of the art and the state of the practice indicate that discovery is a well understood basic service of pervasive computing environments. An exception can be apparently represented by those middleware systems that provide tuple spaces as the only mean of interaction between components. In these cases, services and components can interact indirectly, without *a priori* knowledge each other. However, if the middleware provides interactions though a multiplicity of distributed tuple spaces, the discovery problem does not fully disappear, but simply translates in discovering the

existence—not of other components/services—but of the tuple spaces themselves.

3.4 Mediation

Originally, the mediation activity corresponded to the timely integration of disparate information sources (Wiederhold and Genesereth 1997) and was first used to integrate data stored in databases, knowledge bases, or even file systems. Those initial principles are now used to enable interoperability across diverse pervasive computing systems and platforms (Roth et al. 2018). In particular, a mediation solution can implement operations such as:

- Communication alignment to enable applications using different communication protocols to inter-operate.
- Syntactic alignment to homogenize data formats; this operation often relies on an intermediary format, commonly called a pivot.
- Semantic alignment to align data semantics, in the absence of recognized and used standards in a domain, applications develop different ontologies to represent (static and dynamic) knowledge.
- Non-functional property alignment to ensure certain quality properties for the integration, for instance security or availability.
- Persistency to keep track of all exchanges between applications; the mediation layer can accomplish this through logging support for all requests, responses and data.
- Monitoring to collect data for to verify that the expected quality of service is achieved.

Improved integration of mediation is still an active area of research in order, for instance, to deal with systems of systems or to make pervasive computing platforms interoperate.

3.5 Context support

Research on context-aware computing dates back to the 1990s. Schilit et al. (1994) introduced a still current definition of context and details of a comprehensive context management platform. Since then, a number of context management platforms with specific properties have been introduced.

Nexus (Lehmann et al. 2004) aimed at a global federation of world models that represent local context. Register structures and a query and modeling language are required for scalability. Aura's context service (Judd et al. 2003) also federates context but on a smaller scale and integrates context management using a SQL-like approach. VanSyckel (Becker et al. 2013) describes an extension of context management where prediction algorithms can be integrated to allow applications to adapt proactively.

Context can also be used implicitly, as e.g., in the already mentioned TOTA (Mamei and Zambonelli 2004), where the spatial distribution of data is implicitly used as context. More recently, contextual information has been represented as services (Aygalinc et al. 2016). Here, context appears as a dynamic set of services. Depending on the availability of context sources and the applications needs, different services can be published and withdrawn.

3.6 Configuration and deployment

A major challenge of pervasive computing environments is the mapping of application requirements to available services. These environments differ widely and typically there is no expert present when applications are deployed or configured. Some early approaches used scripting languages, which are obviously not suitable for end-user configuration. Gaia (Kon et al. 2000) started with a script-based configuration and extended this by an automated approach that used an operator-based configuration specification (Ranganathan et al. 2005). O2S (Paluska et al. 2008) has evolved over the years. Starting with goal oriented computing and specifying applications by goals that are automatically mapped to techniques, O2S then developed an abstraction layer of assemblies for composition. PCOM (Becker et al. 2004) relies on explicit contracts of components in order to resolve bindings and configure or adapt an application.

Some visual approaches to configuration programming have been investigated as well. JigSaw (Humble et al. 2003) and the approach in Weis et al. (2016) show promising results in usability. However, configuration and deployment of pervasive computing systems remains challenging and error prone, demanding additional future research.

3.7 Application adaptation

Adaptation—in contrast to configuration—describes the dynamic reconfiguration of an application during runtime. In most cases this is reactive and is based on changes in an application's execution context, e.g., services becoming unavailable or a change in personal context. There are few approaches that use prediction to support proactive adaptation. While reactive adaptation has to be performed when an application can no longer execute correctly (Becker and Schiele 2003), proactive adaptation also consider the quality of prediction, costs for adaptation, utility of configuration and penalty if a prediction fails (VanSyckel et al. 2013). Basically, applications can choose whether they adapt their structure to the change in the execution environment, or—if possible—they change the context (VanSyckel et al. 2014).

Since adaptation is key to pervasive computing, most middleware, applications, and systems offer support for adaptation. The discussion in Harter et al. (1999) provided

one of the first fundamental descriptions of such adaptation. Often, the configuration process is used for adaptation as well (Becker et al. 2004; Kon et al. 2000; Ranganathan et al. 2005). A final interesting approach to adaptation is realized by iROS (Johanson et al. 2002), in which application components are separated by the so-called event heap. Requests can time out if no matching service answers the request. Applications have to detect this by time-outs and react accordingly.

3.8 Edge-, cloud-support and scheduling

A recent trend in pervasive computing is the incorporation of cloud and edge resources. Edge devices can be other mobile devices in the vicinity but also services in the nearby access network. The latter allows mobile devices to rely on data and services that are provided with a minimum delay compared to cloud and grid services (Satyanarayanan 2017).

There are several research questions in this area that have been investigated in this area. A core mechanism in order to utilize functionality of other devices for mobile computing is code-offloading. Maui (Cuervo et al. 2010) is a prominent approach here. In the domain of cloud and edge computing the notion of serverless computing (Baldini et al. 2017) describes approaches in which computation is modeled as closures that contain data and the code that operates on the data. Openwhisk[1] and Amazon Lambda[2] are examples that enable lightweight computation in cloud environments. There are also the beginnings of efforts that combine serverless computing and pervasive computing (Heck et al. 2018). Tasklets started as an abstraction for cloud computing (Schafer et al. 2016) and addressed scheduling tasks in pervasive computing in later work (Edinger et al. 2017). A similar approach is presented in Cicconetti et al. (2019). Overall, integrating edge capabilities into pervasive computing appears promising but requires substantial research with respect to suitable programming abstractions, design methodology and runtime support by middleware platforms. Further, the tradeoffs between the cloud, edge, and mobile devices should be *elastic* this notion of elastic deployments as pervasive computing middleware services is also yet to be explored.

4 Open challenges

The gaps identified previously highlight a neglected issue in pervasive computing middleware, namely, an integrated view of the effective design, development, and deployment of pervasive computing environments (Zambonelli 2017). Given this broader view, it becomes reasonable to ask what are the most suitable software engineering abstractions and system capabilities that are particular to pervasive computing middleware.

4.1 Understanding the players

From the methodological viewpoint, traditional approaches to engineering information systems attack the analysis of system requirements by assuming the existence of well-defined "end-users", who will interact with the resulting system, and "system administrators", who are responsible for configuring the system. Together, these two sets of actors are the parties responsible for eliciting the system's requirements. Traditional approaches also typically adapt a purely functional (typically service-oriented) perspective. However, as we have seen, the situation in pervasive computing environments is much more complex.

Indeed, in such environments, the actors involved may belong to many different categories and may take on much more complex and overlapping roles. For instance, pervasive computing deployments often have *global administrators*, typically the owners of an overall pervasive system and infrastructure, or at least the people empowered to exert control over the configuration, structure, and overall functioning of its applications and services. This is also sometimes referred to as the *enterprise*. There are often also *local administrators*, who typically own (whether permanently or on a temporary basis) a limited portion of the pervasive computing system and are empowered to enforce local control for some portion of the infrastructure for some period of time. Then there are *users*, who typically have some limited access to the overall configuration of the applications and services, i.e., users may not be able to impose new policies on the broad system, but they may nevertheless be entitled to exploit the provided services and in some way configure how such services are provided. That is, *user-level programming* becomes much more mainstream in pervasive computing environments. What is an IFTTT rule[3] if not a user-written program?

The three classes of actors identified above are of a very general nature. For example, considering a scenario of a smart hotel, the above categories can correspond, respectively, to: the hotel managers imposing global policies on, e.g., heating level and surveillance strategies; the organizers of a conference who may be entitled to impose the required behaviours and policies on the meeting rooms they have rented; and regular clients, who need to access pervasive services in their room, and to some extent configure them.

[1] http://openwhisk.org.

[2] http://aws.amazon.com/lambda.

[3] https://ifttt.com/.

Similarly, in the area of urban mobility, the actor categories could correspond to, respectively: mobility managers, parking facility owners or car sharing companies, and private drivers. Accordingly, if a pervasive computing environment and its middleware are not properly developed and configured to account for the different needs of the above classes of actors, and if the above a classes are not properly accounted for the analysis phase, the final system may be unacceptable or unusable.

4.2 Pervasive computing ecosystems

Current pervasive computing architectures are centralized and standalone. That is, they comprise a number of devices, sometimes with direct interactions, linked to a centralized gateway. The purpose of such a gateway is to provide value-added services based on information collected by devices (as said earlier, this is not even so common since most gateways are only used to send information up to the cloud). So, there is no pervasive ecosystem *per se* available today.

This inhibits the development of advanced pervasive computing applications, where a number of devices and gateways have to communicate and then interoperate in order to meet their requirements. Several approaches are currently investigated to enable this broader vision of pervasive infrastructure.

The IoT European Platforms Initiative (IoT-EPI),[4] for instance, is an interesting initiative for IoT platform development. Its aim is to build a sustainable IoT ecosystem in Europe, and it comprises seven projects of which four revolve around interoperability at the communication, protocol, or service level. These projects aim to provide interoperability between IoT platforms through a uniform access to services (often provided by some sort of dedicated hub). Those approaches state that interoperability with legacy devices is ensured and mainly focus on the semantic data heterogeneity. Here, mediation plays a major role that is explored, for instance, through the use of Enterprise Service Bus (ESB) style approaches.

These different and complementary initiatives are noticeable in the sense that they show that a service-based view of the infrastructure allows the construction of pervasive computing ecosystems. However, they still have considerable limitations regarding data management. They see pervasive computing elements essentially as service providers. The fact is that most of them are also data providers. In some domains, like smart manufacturing, they are even intense data providers. Work is needed to allow interactions of almost continuous data flows between pervasive elements in service-based environments. We believe that, in the near future, several interaction paradigms will have to coexist in pervasive computing ecosystems.

4.3 The architecture of a system

From a system abstraction perspective, the functional (service-oriented) view that is typically adopted in "Web of Things" approaches, does not fit well in pervasive computing environments for multiple reasons. In addition to "things" that have basic sensing and actuating functionalities, one should consider the presence of smarter things that can be activated to autonomously perform some long-term activities associated with their capabilities and with their role in the socio-physical environment in which they are situated. These smarter devices can range from cleaning robots to more sophisticated autonomous personal assistants. Second, pervasive computing applications and systems are not simply concerned with providing a suite of coordinated functionalities, but they must often also globally regulate the activities of the system on a continuous basis, according to policies established by its stakeholders and their objectives.

As a consequence, developing pervasive computing services and applications, other than defining and implementing service functionalities, most often implies defining *policies* and *goals* that are then associated to services and applications. In general terms, policies and goals represent desirable "states of affairs" to strive for. In the context of a pervasive computing system, policies and goals represent specific configurations of the system (or of a portion of the system) that applications and services are in charge of eventually producing and/or maintaining. Policies and goals may be defined to apply to the whole system (as realized by global managers), or to apply to specific sub-portions of the systems (realized by local managers).

In this context, the traditional service-oriented perspective of software engineering methodologies and the strong emphasis on services and service composition fall short. On the one hand, software engineering methodologies must properly analyze and design not only services, but also goals and policies, and must provide guidelines for enabling designers to enact these goals and polices dynamically in the system, aided by middleware. One the other hand, middleware for supporting future pervasive computing systems must support the existence of autonomous goal-oriented entities, coordinating with each other towards the achievement of goals and policies, either at the local or at the global level, supported by increased autonomy and intelligence in the devices.

Apart from mapping high level objectives to basic functionality via policies and and goals, the application structure itself may change due to integrating fog and edge resources. Identifying, specifying, and scheduling offloadable parts of an application needs end-to end support from design to

[4] https://iot-epi.eu.

runtime. Especially at runtime user expectations and requirements have to be met and mapped to the dynamic execution environment, e.g., if access to a private cloud or edge resource is not possible, offloading of sensitive data/computation should not be done. Units of computation need to contain data and code. This resembles a closure for computing a task. Integrating this into software engineering methodology and supporting by middleware architectures is a challenge for future pervasive computing systems.

4.4 Supporting autonomy and intelligence

As discussed above, most current middleware systems for pervasive computing and the IoT assume a service-oriented perspective (Razzaque et al. 2016). That is, their primary goal is to coordinate and combine the execution of services and contextual events. Thus, the question of what additional (and possibly different) features a middleware should integrate to properly support autonomous components arises. We believe that much can be taken from the lessons and experiences of research in multiagent systems (Wooldridge 2009).

As is the case in pervasive computing, deploying and executing a distributed multiagent system (i.e., a system of interacting autonomous software agents), calls for a suitable middleware infrastructure. However, unlike traditional pervasive computing middleware, most research in the area of agent-based middleware has explicitly focused on the necessary support for autonomy and distributed decision making. Supporting autonomy implies giving agents the "freedom of action" to eventually pursue their goals, but at the same time implies defining means to monitor autonomous actions, guarantee such actions are "safe" from an overall system viewpoint, and possibly reclaim some degree of autonomy from components whenever necessary in order to preserve some global goal (Mostafa et al. 2017). Supporting distributed decision making implies more than simply composing a set of services according to specific orchestration rules and constraints. It implies supporting a variety of negotiation protocols (Beer et al. 1999) that enable autonomous components to dynamically reach consensus on their courses of action, preserving their autonomy in strategy, yet ensuring that such protocols adhere to "social norms" (Aldewereld et al. 2016).

Looking further into the future, another area in which multiagent systems research could suggest important guidelines for future pervasive computing middleware concerns knowledge-based reasoning. As of today, in the pervasive computing and IoT arenas, sensors are treated almost exclusively as producers of raw data streams and events. Advancements in machine learning techniques, and in the increase of computational power that can be embedded in everyday sensors and objects, will soon make it possible for such devices to *locally* analyze and classify streams of sensed data to extract relevant semantic knowledge (Lippi et al. 2018). We can also expect that such capabilities will evolve to recognize more complex situations, making them capable of *causally* connecting individual patterns into composite situations, that is, making assertions about what is happening around them. For instance, a set of wearables may construct the assertion that "Heart rate increased due to a training session" by integrating the results of sensing two distinct patterns. Or a camera may perform scene understanding, by relating the individual objects it recognizes, e.g., "patient Marco has left the stretcher in corridor X". Similarly, we can soon expect actuators to become not only capable of executing simple tasks, but they will also be able to understand and interpret goals at the semantic knowledge-based level, and possibly argue about their capabilities to achieve such goals. In an environment populated by such smart, autonomous and semantic sensors and actuators, coordination will have to naturally evolve from negotiation towards distributed multi-party *conversations*, or *dialogues* (Amgoud et al. 2000), where the devices discuss and argue with each other to reach a common understanding of situations around, talk to each other to agree on common courses of actions, and possibly dynamically re-negotiate their goals and beliefs. Clearly, as for negotiation protocols, the capability of supporting complex dialogues between such smart components will call for specifically conceived functionalities to be integrated in future pervasive computing middleware systems.

4.5 Humans in the loop

The vision of a future pervasive environment populated by smart goal-oriented components acting autonomously in our everyday environments cannot overlook *humans* as a vital component of the scenario. Humans, in their role of ultimate "users" (in a broad sense) of a pervasive computing system, are the ones that must ultimately be entitled to impose on components to act (and possibly how to act) towards the achievement of specific goals or states in the environment in which they live. To this end, humans must be given the ability to inspect, at any time, the current behaviour of the pervasive computing system. When the environment includes autonomous goal-oriented components, this also implies enabling the human to understand how the system perceives the current state of the affairs, what goals it is currently pursuing and with what planned actions, and why those goals are the "right" ones, given the perceived state.

The above issue can be seen as a specific instance of the more general issue—now a very hot one, due to the increased difficulty of understanding the behaviour of modern deep learning systems—of promoting "explainable" systems (Gunning 2017). In this regard, the perspective of future pervasive computing systems that—yes—devices

can integrate deep learning components to understand situations and plan actions, they can converse to justify their choices, and they can carry out these choices in real time, takes pervasive computing in the correct direction. Indeed, argumentation-based conversations are crucial to help users understand what is happening and are also a mean to enable users to effectively participate in the pervasive computing space.

The ability for humans to participate in the conversational process, other than for understanding, envisions humans as actual critical component of the system: they can participate by providing sensing capabilities (thus acting as smart semantic sensors), and they are inherently intelligent actuators. This convergence between human and software entities is witnessed by many modern *socio-technical systems* (Zambonelli 2012), and it demands researchers and practitioners to conceive, design, and develop systems seamlessly interacting with other software systems and with human agents as well.

Finally, conversation may be a useful and effective user-level way to program a system and configure its behaviour. The need to enable easy and flexible ways to support user-level programming is increasingly recognized as essential. Yet current approaches to user-level programming are very simple, enabling the simple configuration of some device parameters and the definition of cause-effect relations (Kubitza and Schmidt 2017). The approach of current chatbot-based home devices such as Google Home and Amazon Alexa is promising but must evolve to become an enabler for real conversations among humans and devices in ways that allow collaboratively understanding and achieving goals.

5 Conclusions

After three decades of research in pervasive and ubiquitous computing, there is a lot of common understanding and many fundamental research questions have been addressed. Core services, such as context-management and service discovery, are well explored. Interaction models from service oriented models to loosely coupled event based communication have been successfully deployed. However, there remain inherent open challenges that inhibit the realization of the long vision of pervasive computing.

Interoperability is one of these challenges. It will be a major requirement in the near future since many greenfield developments will be made of distributed and often heterogeneous platforms and devices that will need to communicate and cooperate. Interoperability is additionally challenged by the need to incorporate legacy systems (both hardware and software). In addition to all the services and devices present at a time, potentially all devices from earlier installations must also be dealt with and seamlessly integrated. This exceeds syntactic and semantic mappings. New services, sensors, and actuators may affect systems differently, leading to new compositions that provide a better utility than existing ones—or ones that lead to conflicts.

End-users with little or no knowledge and interest of configuration, computer technology, etc., will often be the only humans who are present at deployment, runtime, and maintenance. This is especially true in private places like homes, offices, and ... smart-phones! Means will be needed to keep human in the loop without being overwhelmed by technology.

Complexity and dynamism are inherent in these environments. When more and more everyday items communicate, provide sensor information, and allow outside entities to set their state, systems will have to be tailored to a specific environment. This, in most cases, will have to be done by the user. *Application composition and scheduling* remain cornerstone open challenges. Many early and existing approaches to pervasive computing are based on closed scenarios, e.g., smart homes, and on service-oriented interaction. Incorporating edge and fog devices leads to new challenges in the application architecture as well as in scheduling a distributed, Pervasive Computing application.

As of today, there is no silver bullet. But a number of different research areas converge and will help to conquer the challenges. Autonomic techniques, artificial Intelligence mechanisms, and efficient algorithms will provide means for configuration in dynamic environments. Human computer interaction has explored a number of applications and interaction models. This in combination with efficient algorithms, interaction models of multi agent systems and systems research can provide the necessary balance between the users expectations and skills in order to configure and manage the environment.

Even after such a long period of research middleware for pervasive computing remains an exciting research field.

References

Ahuja, S., Carriero, N., Gelernter, D.: Linda and friends. IEEE Comput. **19**(8), 26–34 (1986)

Aldewereld, H., Dignum, V., Vasconcelos, W.W.: Group norms for multi-agent organisations. TAAS **11**(2), 15:1–15:31 (2016)

Amadeo, M., Campolo, C., Iera, A., Molinaro, A.: Named data networking for IoT: an architectural perspective. In: Proceedings of the 2014 European Conference on Networks and Communication, pp. 1–5, (2014)

Amgoud, L., Maudet, N., Parsons, S.: Modelling dialogues using argumentation. In: Proceedings Fourth International Conference on MultiAgent Systems, pp. 31–38 (2000)

Arnold, K., Scheifler, R., Waldo, J., O'Sullivan, B., Wollrath, A.: Jini Specification. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)

Asadi, A., Qant, Q., Mancuso, V.: A survey on device-to-device communication in cellular networks. IEEE Commun. Surv. Tutor. **16**(4), 1801–1819 (2014)

Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., Suter, P.: Serverless Computing: Current Trends and Open Problems, pp. 1–20. Springer, Singapore (2017)

Becker, C., Schiele, G.: Middleware and application adaptation requirements and their support in pervasive computing. In: 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.(ICDCSW), vol. 00, p. 98, 05 (2003)

Becker, C., Schiele, G., Gubbels, H., Rothermel, K.: BASE—a micro-broker-based middleware for pervasive computing. In: Proceedings of PerCom, pp. 443–451 (2003)

Becker, C., Handte, M., Schiele, G., Rothermel, K.: PCOM–a component system for pervasive computing. In: Proceedings of PerCom, pp. 67–76 (2004)

Becker, C., VanSyckel, S., Schiele, G.: Ubiquitous information technologies and applications. Lecture Notes in Electrical Engineering **214**(1) (2013)

Beer, M., d'Inverno, M., Luck, M., Jennings, N., Preist, C., Schroeder, M.: Negotiation in multi-agent systems. Knowl. Eng. Rev. **14**(3), 285–289 (1999)

Bello, O., Zeadally, S.: Intelligent device-to-device communication in the internet of things. IEEE Syst. J. **10**(3), 1172–1182 (2014)

Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16 (2012)

Bourcier, J., Diaconescu, A., Lalanda, P., McCann, J.A.: Autohome: an autonomic management framework for pervasive home applications. TAAS **6**(1), 8:1–8:10 (2011)

Caporuscio, M., Raverdy, P.-G., Issarny, V.: ubiSOAP: a service-oriented middleware for ubiquitous networking. IEEE Trans. Serv. Comput. **5**(1), 86–98 (2012)

Chiang, M., Zhang, T.: Fog and iot: an overview of research opportunities. IEEE Internet Things J. **3**(6), 854–864 (2016)

Cho, S., Julien, C.: ChitChat: Navigating tradeoffs in device-to-device context sharing. In: Proceedings of the International Conference on Pervasive Computing and Communications (2016)

Choi, K.W., Han, Z.: Device-to-device discovery for proximity-based service in LTE-advanced systems. IEEE J. Sel. Areas Commun. **33**(1), 55–66 (2015)

Chollet, S., Lalanda, P., Escoffier, C.: Extension of service-oriented component models for dynamic environment. In: 2015 IEEE International Conference on Services Computing, SCC 2015, New York, NY, USA, June 27–July 2, 2015, pp. 648–655. IEEE Computer Society (2015)

Cicconetti, C., Conti, M., Passarella, A.: Low-latency distributed computation offloading for pervasive environments. In: Pervasive Computing and Communications (PerCom), 2019 IEEE International Conference on. IEEE (2019)

Colin, A., Gerbert-Gaillard, E., Vega, G., Lalanda, P.: Service-oriented autonomic pervasive context. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S., (eds.) Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016, Proceedings, Volume 9936 of Lecture Notes in Computer Science, pp. 795–809. Springer (2016)

Conti, M., Das, S., Bisdikian, C., Kumar, M., Ni, L., Passarella, A., Roussos, G., Troster, G., Tsudik, G., Zambonelli, F.: Looking ahead in pervasive computing: challenges and opportunities in the era of cyber-physical convergence. Pervasive Mobile Comput. **8**(1), 2–21 (2012)

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: making smartphones last longer with code offload. In: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62. ACM (2010)

Dixon, C., Mahajan, R., Agarwal, S., Bernheim B.A.J., Lee, B., Saroiu, S., Bahl, P.: An operating system for the home. In: Gribble, S.D., Katabi, D. (eds) Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012, pp. 337–352. USENIX Association (2012)

Edinger, Janick, Schäfer, Dominik, Krupitzer, Christian, Raychoudhury, Vaskar, Becker, Christian: Fault-avoidance strategies for context-aware schedulers in pervasive computing systems. In: 2017 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 79–88. IEEE (2017)

Edwards, W.K., Grinter R.E.: At home with ubiquitous computing: seven challenges. In: Abowd, G.D., Brumitt, B., Shafer, S.A. (eds) Ubicomp 2001: Ubiquitous Computing, Third International Conference Atlanta, Georgia, USA, September 30 - October 2, 2001, Proceedings, volume 2201 of Lecture Notes in Computer Science, pp. 256–272. Springer (2001)

Escoffier, C, Chollet, S., Lalanda, P.: Lessons learned in building pervasive platforms. In: 11th IEEE Consumer Communications and Networking Conference, CCNC 2014, Las Vegas, NV, USA, January 10–13, 2014, pp. 7–12. IEEE (2014)

Escoffier, C., Hall, R.S., Lalanda, P.: iPOJO: an extensible service-oriented component framework. In: Proceedings of International Conference on Services Computing (SCC), pp. 474–481. IEEE (2007)

Familiar, M.S., Martínez, J.-F., López-Santidrián, L.: Pervasive smart spaces and environments: a service-oriented middleware architecture for wireless ad hoc and sensor networks. IJDSN **8**, 725190 (2012)

Ferscha, A., Hechinger, M., Mayrhofer, R., Oberhauser, R.: A lightweight component model for peer-to-peer applications. In: Proceedings of the International Conference on Distributed Computing Workshops, pp. 520– 527, 04 (2004)

Golrezaei, N., Molisch, A.F., Dimakis, A.G.: Base-station assisted device-to-device communications for high-throughput wireless video netowrks. In: Proceedings of ICC, June (2012)

Grimm, R.: One.world: experiences with a pervasive computing architecture. IEEE Pervasive Comput. **3**, 22–30 (2004). 07

Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. **28**(1), 1–18 (2005)

Gunning, D.: Explainable artificial intelligence (xai). Defense Advanced Research Projects Agency (DARPA) (2017)

Guo, B., Zhang, D., Wang, Z., Yu, Z., Zhou, X.: Opportunistic IoT: exploring the harmonious interaction between human and the internet of things. J. Netw. Comput. Appl. **36**(6), 1531–1539 (2013)

Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: MOBICOM '99, The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, USA, August 15–19, 1999., pp. 59–68, (1999)

Heck, M., Edinger, J., Schäfer, D., Becker, C.: Iot applications in fog and edge computing: where are we and where are we going? In: 2018 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1–6. IEEE (2018)

Helal, S., Mann, W.C., El-Zabadani, H., King, J., Kaddoura, Y., Jansen, E.: The gator tech smart house: a programmable pervasive space. IEEE Comput. **38**(3), 50–60 (2005)

Humble, J., Crabtree, A., Hemmings, T., Åkesson, K-P., Koleva, B., Rodden, T., Hansson, P.: "playing with the bits" user-configuration

of ubiquitous domestic environments. In: UbiComp 2003: Ubiquitous Computing, 5th International Conference, Seattle, WA, USA, October 12-15, 2003, Proceedings, pp. 256–263 (2003)

Jenson, S.: The physical web. In: Proceedings of CHI'14: Extended Abstracts on Human Factors in Computing Systems, pp. 15–16 (2014)

Johanson, B., Fox, A., Winograd, T.: The interactive workspaces project: experiences with ubiquitous computing rooms. IEEE Pervasive Comput. **1**(2), 67–74 (2002)

Judd, G., Steenkiste, P.: Providing contextual information to pervasive computing applications. In: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), March 23–26, 2003, Fort Worth, Texas, USA, pp. 133–142 (2003)

Kon, F., Román, M., Liu, P., Mao, J., Yamane, T., Magalhã, C., Campbell, R.H.: Monitoring, security, and dynamic configuration with the dynamictao reflective ORB. In: IFIP/ACM International Conference on Distributed Systems Platforms, Middleware '00, pp. 121–143. Springer, Berlin, Heidelberg (2000)

Kubitza, T., Schmidt, A.: meSchup: a platform for programming interconnected smart things. IEEE Comput. **50**(11), 38–49 (2017)

Lalanda, P., McCann, J.A., Diaconescu, A.: Autonomic computing—principles, design and implementation. Undergraduate Topics in Computer Science. Springer (2013)

Lalanda, P., Morand, D., Chollet, S.: Autonomic mediation middleware for smart manufacturing. IEEE Intern. Comput. **21**(1), 32–39 (2017)

Lehmann, O., Bauer, M., Becker, C., Nicklas, D.: From home to world-supporting context-aware applications through world models. In: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), 14–17 March 2004, Orlando, FL, USA, pp. 297–308 (2004)

Lin, X., Andrews, J., Ghosh, A., Ratasuk, R.: An overview of 3GPP device-to-device proximity services. IEEE Commun. Mag. **52**(4), 40–48 (2014)

Lippi, M., Mamei, M., Mariani, S., Zambonelli, F.: An argumentation-based perspective over the social iot. IEEE Internet Things J. **5**(4), 2537–2547 (2018)

Liu, C.H., Yang, B., Liu, T.: Efficient naming, addressing and profile services in internet-of-things sensory environments. Ad Hoc Netw. **18**, 85–101 (2014)

Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the tota middleware. In: Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the, pp. 263–273 (2004)

Mayer, S., Inhelder, N., Verborgh, R., Van de Walle, R., Mattern, F.: Configuration of smart environments made simple: combining visual modeling with semantic metadata and reasoning. In: Proceedings of the 2014 International Conference on the Internet of Things, pp. 61–66 (2014)

Mostafa, S.A., Ahmad, M.S., Mustapha, A.: Adjustable autonomy: a systematic literature review. Artif. Intell. Rev. (2017)

Murphy, A.L., Picco, G.P., Roman, G.: Lime: a middleware for physical and logical mobility. In: Proceedings 21st International Conference on Distributed Computing Systems, pp. 524–533 (2001)

OpenConnectivityFoundation. UPnP Specifications, September (2016)

OSGi Alliance: OSGi Service Platform Core Specification Release 4, (2007)

Paganelli, F., Parlanti, D., Giuli, D.: Message-based service brokering and dynamic composition in the SAI middleware. In: 2010 IEEE International Conference on Services Computing, SCC 2010, Miami, Florida, USA, July 5–10, 2010, pp. 474–481. IEEE Computer Society (2010)

Paluska, J.M., Pham, H., Saif, U., Chau, G., Terman, C., Ward, S.: Structured decomposition of adaptive applications. Pervasive Mobile Comput. **4**(6), 791–806 (2008). PerCom 2008

Papazoglou, M.P.: Service-oriented computing: concepts, characteristics and directions. In: 4th International Conference on Web Information Systems Engineering, WISE 2003, Rome, Italy, December 10-12, 2003, pp. 3–12. IEEE Computer Society, (2003)

Quevedo, J., Antunes, M., Corujo, D., Gomes, D., Aguiar, R.L.: On the application of contextual iot service discovery in information centric networks. Comput. Commun. **89**, 117–127 (2016)

Ranganathan, A., Chetan, S., Al-Muhtadi, J., Campbell, R.H., Mickunas, M.D.: Olympus: a high-level programming model for pervasive computing environments. In: Third IEEE International Conference on Pervasive Computing and Communications, pp. 7–16 (2005)

Razzaque, M.A., Milojevic-Jevric, M., Palade, A., Clarke, S.: Middleware for internet of things: a survey. IEEE Inter. Things J. **3**(1), 70–95 (2016)

Román, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: a middleware platform for active spaces. SIGMOBILE Mob. Comput. Commun. Rev. **6**(4), 65–67 (2002)

Romero, D., Hermosillo, G., Taherkordi, A., Nzekwa, R., Rouvoy, R., Eliassen, F.: The digihome service-oriented platform. Softw. Pract. Exp. **43**(10), 1205–1218 (2013)

Roth, F.M., Becker, C., Vega, G., Lalanda, P.: XWARE—a customizable interoperability framework for pervasive computing systems. Pervasive and Mobile Comput. **47**, 13–30 (2018)

Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)

Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput. **8**(4), 14–23 (2009)

Schafer, D., Edinger, J., Paluska, J.M., VanSyckel, S., Becker, C.: Tasklets: "better than best-effort" computing. In: Computer Communication and Networks (ICCCN), 2016 25th International Conference on, pp. 1–11. IEEE (2016)

Schiele, G., Becker, C., Rothermel, K.: Energy-efficient cluster-based service discovery for ubiquitous computing. In: Proceedings of the 11th Workshop on ACM SIGOPS European Workshop, EW 11, ACM, New York, NY (2004)

Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on, pp. 85–90. IEEE (1994)

Shi, W., Cao, J., Zhang, Q., Li, Y., Lanyu, X.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)

VanSyckel, S., Schäfer, D., Majuntke, V., Krupitzer, C., Schiele, G., Becker, C.: COMITY: a framework for adaptation coordination in multi-platform pervasive systems. Pervasive Mobile Comput. **10**, 51–65 (2014)

VanSyckel, S., Schäfer, D., Schiele, G., Becker, C.: Configuration management for proactive adaptation in pervasive environments. In: Proceedings of IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 131–140 (2013)

Verbelen, T., Simoens, P., De Turck, F., Dhoedt, B.: Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the 3rd ACM Workshop on Mobile Cloud Computing, pp. 29–36 (2012)

Wehner, P., Piberger, C., Göhringer, D.: Using JSON to manage communication between service in the Internet of Things. In: Proceedings of the 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip, pp. 1–4 (2014)

Weis, T., Handte, M., Knoll, M., Becker, C.: Customizable pervasive applications. In: 4th IEEE International Conference on Pervasive Computing and Communications (PerCom 2006), 13–17 March 2006, Pisa, pp. 239–244 (2006)

Wiederhold, G., Genesereth, M.R.: The conceptual basis for mediation services. IEEE Expert **12**(5), 38–47 (1997)

Wooldridge, M.J.: An Introduction to MultiAgent Systems, 2nd edn. Wiley, Hoboken (2009)

Xu, Z., Peng, X., Zhang, L., Li, D., Sun, N.: The Φ-stack for smart web of things. In: Proceedings of the Workshop on Smart Internet of Things, SmartIoT@SEC 2017, San Jose/Silicon Valley, CA, USA, October 14, 2017, pp. 10:1–10:6. ACM (2017)
Zambonelli, F.: Toward sociotechnical urban superorganisms. IEEE Comput. **45**(8), 76–78 (2012)
Zambonelli, F.: Key abstractions for iot-oriented software engineering. IEEE Softw. **34**(1), 38–45 (2017)
Zambonelli, F., Salim, F., Loke, S.W., De Meuter, W., Kanhere, S.: Algorithmic governance in smart cities: The conundrum and the potential of pervasive computing solutions. IEEE Technol. Soc. Mag. **37**(2), 80–87 (2018)

Christian Becker studied computer science in Karlsruhe and Kaiserslautern, Germany. He received his Ph.D. from the University of Frankfurt. He worked as a postdoc at the University of Stuttgart, Germany, and was involved in the collaborative research center (SFB 627) NEXUS which investigated large scale context management and application support. Since 2006 he is full professor at the University of Mannheim, Germany, where he also acts as chairperson of subject area Information Systems. His research interests are distributed and adaptive systems. Dr. Becker is active in a variety of international events, such as the IEEE PerCom series (Stearing Committee member, TPC Chair 2016, General Chair 2010), IEEE Mobile Data Management (TPC Chair 2007, General Chair 2015). He has (co-)authored more than 170 publications in the areas of Distributed Systems, Pervasive Computing, and Computer Networks.

Christine Julien is a professor in the Center for Advanced Research in Software Engineering (ARiSE) in the Department of Electrical and Computer Engineering at the University of Texas at Austin, which she joined in 2004. She is the director of the Mobile and Pervasive Computing Group, where her research focuses on the intersection of software engineering and dynamic, unpredictable networked environments. Her specific focus is on the development of models, abstractions, tools, and middleware whose goals are to ease the software engineering burden associated with building applications for pervasive and mobile computing environments. Dr. Julien's research has been supported by the National Science Foundation (NSF), the National Institutes of Health (NIH) the Air Force Office of Scientific Research (AFOSR), the Department of Defense, Google, and Freescale Semiconductors. The work has been recognized by an NSF CAREER award and an AFOSR Young Investigator Award, and the results have appeared in many peer reviewed journal and conference papers. Dr. Julien received a D.Sc. in Computer Science in 2004 from Washington University in Saint Louis.

Philippe Lalanda is a Professor at Grenoble-Alpes University (UGA) where he teaches Software Engineering and leads the Adele research team. He completed his PhD on real-time blackboard systems in Nancy University and applied this work to the control of smart robots at Stanford University in the Knowledge System Laboratory. He then worked for ten years in the industry (Dassault Aviation, Thales, Schneider Electric) where he held the positions of software architect and R&D project leader. Philippe Lalanda now conducts research in the fields of autonomic computing and software engineering, mostly applied to pervasive computing. He has authored some 100 papers in international journals and conferences and supervised 20 PhD. He has also co-authored a reference book on Autonomic Computing (Springer Verlag). He has served in a number of conferences as reviewer, program chair and general chair. He finally serves as an expert at the European Commission and in diverse French research institutions. Philippe Lalanda received an IBM Faculty Award in 2015.

Franco Zambonelli is full professor of Computer Science at the University of Modena and Reggio Emilia. He got his PhD in Computer Science and Engineering from the University of Bologna in 1997. His research interests include: pervasive computing, multi-agent systems, self-adaptive and self-organizing systems. He has published over 100 papers in peer-reviews journals, and has been invited speaker at many conferences and workshops. He is in the editorial board of the ACM Transactions on Autonomous and Adaptive Systems, Elsevier Journal of Pervasive and Mobile Computing, IEEE Society & Technology Magazine, the BCS Computer Journal, and he is in the Steering Committee of the IEEE SASO Conference. He has been scientific manager of the EU FP6 Project CASCADAS and coordinator of the EU FP7 Project SAPERE. He is ACM Distinguished Scientist, member of the Academia Europaea, and IEEE Fellow.