Chapter 11
Middleware Support for Self-aware Computing Systems

Jennifer Simonjan, Bernhard Dieber, and Bernhard Rinner

Abstract The implementation of a distributed self-aware computing system (SACS) typically requires a substantial software infrastructure. A middleware system with dedicated services for self-awareness and self-expression can therefore support the development of SACS applications. In this chapter we show the advantages of using a middleware system as the basis for a self-aware computing system. We identify requirements for middleware systems to support the development of self-aware applications. By providing facilities for communication, decoupling and transparency, middleware systems can provide essential features needed in SACS. We compare different middleware paradigms and their suitability to support self-awareness in distributed applications. We argue that the publish/subscribe paradigm is very well suited for this application area since it supports modularisation and decoupling. Units can be added to and removed from existing applications and may well be reused in new applications. Thus, SACS can be constructed by recombining existing publish/subscribe modules. In addition, we present details of publish/subscribe and introduce our middleware implementation called Ella. We describe how different aspects of a SACS and patterns for self-aware applications can be represented using Ella. We present different communication paradigms in Ella (broadcasting, peer2peer) as well as decoupling mechanisms provided by the middleware. We argue that SACS applications can be developed (i) faster, (ii) more efficiently and (iii) more reliable with Ella. Finally, Chapter 13 presents a self-aware and self-expressive multi-camera application which has been implemented with Ella.

Jennifer Simonjan
Alpen-Adria-Universität Klagenfurt, Austria, e-mail: jennifer.simonjan@aau.at

Bernhard Dieber
Alpen-Adria-Universität Klagenfurt, Austria, e-mail: bernhard.dieber@counity.at

Bernhard Rinner
Alpen-Adria-Universität Klagenfurt, Austria, e-mail: bernhard.rinner@aau.at

This chapter is part of the book: P.R. Lewis, M. Platzner, B. Rinner, J. Tørresen, X. Yao (Eds.) Self-aware Computing Systems - An Engineering Approach. Springer 2016. pp 215-238. http://link.springer.com/chapter/10.1007/978-3-319-39675-0_11

11.1 Introduction to Middleware Systems

Self-aware computing systems (SACS) [310, 212, 241] are often distributed systems i.e., networked computing systems running on multiple network nodes. Distributed applications are inherently more difficult to design, develop and maintain than applications running on single nodes. This is also true for distributed SACS. Middleware systems are therefore employed in networked application architectures to ease the development by abstracting parts of the networking from the application. These concepts can also be reused in the context of SACS. However, for self-aware computing we see additional requirements for a middleware system.

In this chapter we look at different middleware paradigms and discuss their suitability to support SACS. We show which middleware functions can support architectural primitives of self-aware systems and focus on the publish/subscribe paradigm. Further, we present a specific implementation—the Ella middleware [100]—and describe how its properties can support the development of distributed SACS.

11.1.1 Middleware Basics

A middleware system is a software layer which is located between the operating system and the application. Therefore, it serves as a bridging layer connecting distributed applications. Middleware thus provides similar services as an operating system but for distributed applications rather than for a single computer. The distinction between operating system and middleware functionality is, to some extent, arbitrary. While core kernel functionality can only be provided by the operating system itself, some functionality previously provided by dedicated middleware systems has been integrated into operating systems nowadays. A typical example is the TCP/IP stack for networking, nowadays included virtually in every operating system. As shown in Figure 11.1, every device in a network runs middleware which is located between the application and the transport layer. The logical communication is established between the corresponding layers of different devices and is depicted by the horizontal dashed lines. The physical communication takes place in a vertical manner through the communication stack of the devices and is depicted by the solid lines.

In a distributed system, the applications typically face heterogeneity in multiple dimensions. Such applications run on different physical locations, using different hardware platforms, networking technologies, operating systems or programming languages. A middleware system provides services for a distributed execution of applications and therefore eases the application development. Key aspects are hiding the complexity and heterogeneity of a distributed system and providing a messaging service that enables communication across all platforms within the system. A further important aspect of middleware systems is reusability.

An example of modern heterogeneous systems is a smart environment application. Smart environments are typically composed of different sensors, such as visual, acoustic or infrared sensors [278, 332]. In these networks, each node performs local

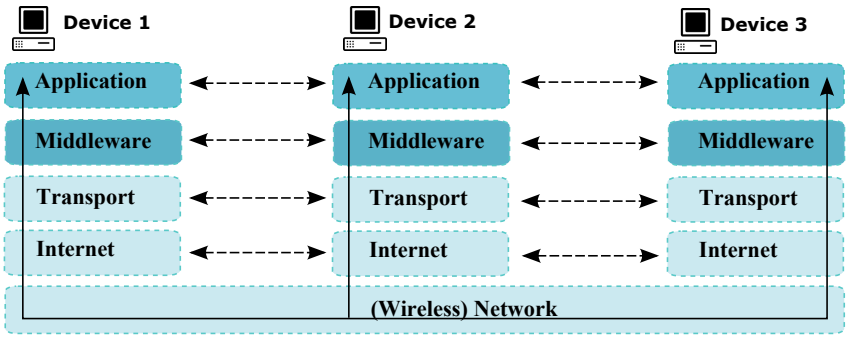

Fig. 11.1 Layered network architecture of multiple devices in a network

actions such as sensing and global actions including communication and cooperation with neighbouring nodes.

Summarised, a middleware system should provide the following functionalities: (i) hiding distribution, (ii) hiding heterogeneity and low-level details, (iii) providing uniform language and platform-independent interfaces to application developers, (iv) providing common services such as messaging. Applying the design patterns presented in Chapter 5, middleware can support an application in accessing external sensors and activating external actuators. Depending on the paradigm, middleware can also provide support in implementing self-awareness principles and self-expressive behaviour. The reference architecture introduced in Chapter 4 can be implemented by means of a middleware system if it is realised as a distributed application.

11.1.2 Application Example of a Distributed Self-aware Computing System

To explain SACS and middleware usage in such systems, we introduce a sensor network example. A distributed sensor network consists of various sensors and actuators which have some communication capabilities. There is no central entity controlling the network nodes. The actions taken by nodes rely on environmental information and on interaction with local neighbours. Sensors capture certain events (i.e., light intensity, temperature) and push them as stimuli into the application. This can mean either that the sensor uses this data itself to change its own behaviour or that it informs other nodes in the network (i.e., by a broadcast).

We use a sensor network within a smart environment as an example. It is responsible for controlling the access of persons to rooms within a building. In addition, it informs the infrastructure within a room of who has entered it, in order to al-

low the smart environment to adapt to specific users. It consists of passive infra-red (PIR) sensors, RGB cameras, an authentication/authorisation node as well as actuators which control door locks. To save resources and to minimise privacy invasion, the RGB cameras are in standby by default, where they do not record any images. Whenever a PIR sensor in front of a door recognises movement, it sends out a message which activates all corresponding cameras. They will start to stream images which are processed by the authentication/authorisation node. There, face recognition algorithms are used to determine if the person is allowed to access a certain room. If a face has successfully recognised, the sensor network will instruct the door locks to open. In addition, the smart environment within the room may adapt to the newly entered user by adapting the lighting or switching on devices. Which adaptation is performed depends on the devices in the room and the profile of the user. Figure 11.2 shows a visualisation of the example application using the reference architecture introduced in Chapter 4.

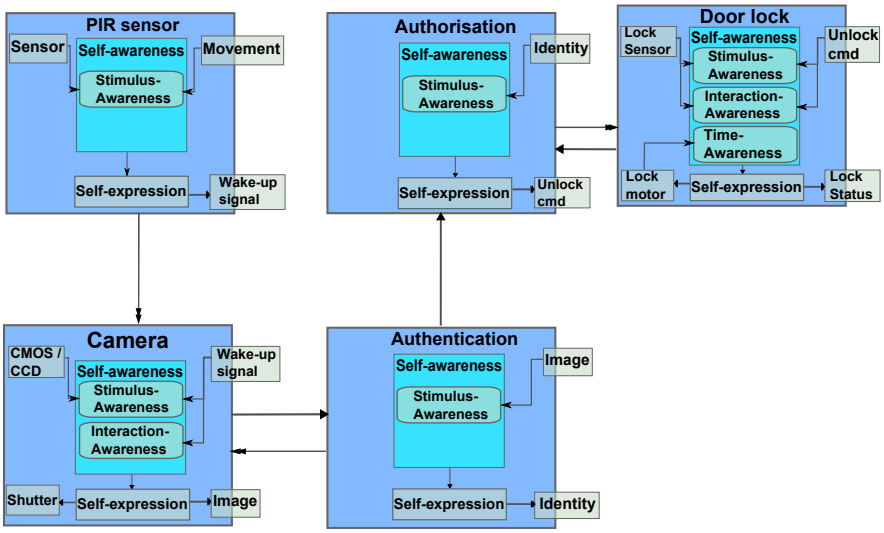

Fig. 11.2 Visualisation of the example application using the reference architecture introduced in Chapter 4

The PIR sensor detects movement in the environment with its external sensors. Upon a movement detection, the self-awareness module of the PIR sensor is informed by the sensors. This module is the prerequisite for self-expression capabilities, which allow a node to react properly to certain events. Thus, the self-expression module is responsible for deciding on how to act based on learnt models. In the case of the PIR sensor, this module issues a wake-up signal in order to inform the appropriate cameras of a movement in their area of interest. The wake-up signal received by a camera triggers its self-awareness module. Since cameras are aware

not only of stimuli from the environment but also of stimuli from other agents, they are stimulus- as well as interaction-aware. The self-expression module of a camera reacts properly by starting to stream images to the authentication node. This node performs a face detection and forwards the identity information to the authorisation node, which decides if a door should be unlocked. If so, it issues an unlock command which is received by the door lock node. The door lock node is also time-aware, since it is able to lock an unlocked door after a certain time period. The self-expression module of the door lock node decides on the lock status of the door based on the learnt knowledge.

This application poses several challenges and can profit from SACS design principles. First, there are components with different purposes which change their states based on input from internal and external sources. Second, time dependencies exist in this system (e.g., biometric information must be current in order to open the door for the right person). Third, the various components should not constantly poll their data producers for recent data. Instead, new data should be provided in a push-based manner.

This networked application can be supported by a middleware system. First, multiple networked devices have to be connected to an integrated system. Second, the same application may run in different infrastructural circumstances, i.e., some rooms may not have PIR sensors (requiring the cameras to run permanently) or there may be additional authentication devices such as fingerprint or card readers. Further, different smart environment devices need to react differently to new persons. This requires the application to adapt and scale. Most of those circumstances can be hidden to the application using a middleware system. Finally, networked applications are typically rather communication intensive. Middleware reduces the amount of time required to implement such an application and can also be used to transparently incorporate different networking technologies like WiFi and ZigBee. In addition, middleware systems which provide space decoupling (i.e., hide the exact location of application modules) enable a restructuring of the application to a single node without programming effort.

We will use this application scenario consistently throughout this chapter to explain self-awareness requirements, middleware paradigms and implementation concepts.

11.2 Middleware Requirements

In order to be useful for distributed applications, a middleware system must satisfy several general requirements. First, it must provide meaningful services to the application (e.g., communication and access to data). Second, it must be flexible enough to allow for application restructuring and quick exchange of functionality. Thus, applications should be constructed from several modules or plug-ins. Further, the middleware system should provide mechanisms to decouple individual modules, i.e., make them independent from each other wherever possible. Also, a middleware

system should ease application development by hiding complexity regarding heterogeneous hardware or networks. As an example, the size of a network can be hidden by a middleware system, i.e., a developer should not be concerned if the application runs on two or twenty nodes. Thus, a middleware system must transparently adapt its operation according to the network size. This also includes changing the network size, i.e., when nodes join or leave the network during the application runtime.

Finally, it is required that the middleware system be simple, both in structure and usage. This reduces the effort for understanding and using its concepts in developing applications, makes the execution faster and eases maintenance.

Distributed self-aware computing systems pose additional requirements to a middleware system. While it is not required for the middleware itself to be self-aware, it should support the application in being self-aware. It must provide facilities which make it easy for developers to asses the self-state of the system and to express it in the application behaviour.

We derive feature requirements for a SACS-supporting middleware from the architectural primitives (see Chapter 12) of a self-aware computing system as shown in Table 11.1.

Table 11.1 Middleware requirements derived from SACS architectural primitives

Architectural primitive	Supporting middleware feature
stimulus-awareness	Ability to push a new stimulus to the application
interaction-awareness	Ability to enable the application to participate in interactions with other application parts
time-awareness	Ability to relate stimuli or events to time instances or knowledge of historical/future phenomena
self-expression	Ability to re-route and modify data flows and ability to be split into components and exchange modules during runtime

For *stimulus-awareness*, the application needs a comfortable way to receive a stimulus. For a video-processing application, this might be a new image coming from a camera. Thus, a middleware system should provide a mechanism to push stimuli into the application or actively notify the application of a new stimulus.

To support *interaction-awareness*, the middleware system should make it easy for the application to participate in interactions like auctions, communication groups and multicast channels. This includes easy dissemination of messages and data and push-based message and data reception. Therefore, nodes need to know with whom they are communicating.

Supporting *time-awareness* means providing time information to the application. This can be done in various ways and on different levels: (i) time-stamping events, (ii) keeping the intervals between events proportional to the real time intervals, (iii) temporally ordering events and (iv) maintaining knowledge of historical data or predicting future phenomena. A middleware system can support each of these levels. For time-stamping the middleware system could by default add a header including

a time-stamp to each application message. Keeping the right time intervals and enabling temporal ordering would require the middleware system to buffer messages. To increase knowledge of historical data, the middleware system needs to store past events and enable nodes to access this data. Additionally, many distributed applications should run without any synchronisation. This means that the interacting parties do not need to be actively participating in the interaction at the same time. The middleware system should therefore support unsynchronised communication and interaction (known as time-decoupling).

Self-expression allows a node to take actions appropriate for learnt models or knowledge. A middleware system can support the application in re-routing data flows or by activating and deactivating system parts based on self-awareness. The latter can be realised in middleware systems which support a developer in modularising an application and thus make it easy to exchange modules at runtime.

Summarised, developing a distributed SACS application can be noticeably simplified by the support of a middleware system. At start-up of the application, node discovery is the main issue. A middleware system should take care of discovery of neighbouring nodes to enable later interaction between them and support interaction-awareness. Further, the middleware system should take care of the communication—forwarding of messages/events, communication channel, buffering of messages, etc. Taking care of delivery of messages and events means supporting stimulus-awareness of the application. In the sense of time-awareness, a middleware system could take care of temporal ordering of events (lowest form of time-awareness) or time-stamping of application messages (highest form of time-awareness). Also, self-expression can be supported by a middleware system.

11.3 Middleware Paradigms

Keeping the design issues and application goals in mind, middleware systems have to fulfill certain requirements in order to be useful in developing SACS. We classify middleware systems into the following two approaches: host-centric and content-centric systems.

A host-centric middleware system allows distributed applications to exchange general-purpose messages between specified hosts. This is used to remotely call functions, transport data and send notifications to known hosts. A key aspect of host-centric approaches is that for a single call typically there is a single previously known remote host. Those kinds of systems were originally intended to support only synchronous communication. However, language-specific extensions also allow asynchronous communication. Message delivery is enabled by knowing the host to communicate with and is ensured using reliable message queues.

In content-centric middleware, the nodes or applications are interested in specific data, rather than in the origin or sink of the data. A producer publishes data by making it accessible to the other nodes of the network. Consumers interested in this type of data can then consume the data. Content-oriented systems rely on asyn-

chronous communication, which means that processes do not need to be blocked while waiting for a reply. Using this approach, senders have no guarantee that their messages are read and no indication in the transmission time. The main advantage is that communication can be realised in either a one-to-one or a one-to-many manner, whereby neither producers nor consumers care about that.

11.3.1 Host-Centric Middleware

Host-centric systems are interested in the host they are communicating with. Those systems typically support synchronous communication with a possible extension to asynchronous communication. In synchronous communication, the sender is blocked until a reply message is received. This means that the sender cannot execute any other functions while waiting for a reply. Further, connection overhead is introduced, since each call requires, for instance, marshalling (transformation of the stored object to data which is suitable for transmission). Another drawback is that senders need to know receivers and their locations or addresses in advance. Using asynchronous communication, sender and receiver are loosely coupled, meaning that a sender continues its work after sending a message. When the call returns (i.e., it has been processed on the remote host), the sender can collect the return value.

11.3.1.1 Remote Procedure Calls (RPCs)

A well-known example of host-centric middleware systems is the so-called remote procedure call (RPC) [300]. RPC enables the invocation of a procedure or function on a remote node in the same way local procedures are called (as if they belong to the same process). RPC uses synchronous communication, meaning that the calling process blocks until the remote procedure replies to the call. Changing interfaces of procedures on one side will therefore result in the need for changes of the calls to the procedures, making RPC inflexible. A remote procedure call is a one-to-one communication, i.e., to call a procedure on multiple remote hosts, multiple calls are necessary. RPC calls work similarly to local function calls: the calling arguments are passed to the remote procedure and the caller waits for a response. A remote procedure needs to be uniquely identifiable within the whole network. An example implementation of RPC is the so-called message passing interface (MPI). In MPI, the sender sends a message to a process rather than invoking a function directly by name. This standard defines syntax and semantics of core library routines to ease application development. It supports one-to-one as well as collective communication and is the most widely used model in high-performance computing.

11.3.1.2 Object-Oriented Middleware (OOM)

Object-oriented middleware systems extend the pure method-oriented concepts of RPC to object-oriented programming. Here, not only a functional interface but whole objects are provided to a remote interaction partner. Within an object-oriented language, remote and local objects can ideally be used in the same way.

Prominent implementations of OOM systems are CORBA, JavaRMI, SOAP and .NETRemoting. Common Object Request Broker, known as CORBA [401], allows method calls between application objects of different applications. CORBA hides the complexity of different operating systems and hardware platforms from the application developer, enabling an easier development of distributed applications. An interface definition language (IDL) is used to specify the interfaces represented by objects to the network. Mappings from IDL to programming languages such as Java or C++ are also specified by CORBA. CORBA data such as integers or arrays are passed by value, while CORBA objects are passed by reference. Although CORBA supports flexible data types, data-by-value passing enforces strong data typing. Object Request Broker is the essential concept used by CORBA. ORB allows clients to make requests for services without the knowledge of the server's location or interface. To enable correct delivery of messages and replies, the IIOP protocol is used. The Internet Inter-ORB Protocol (IIOP) enables applications of different nodes and different operating systems to communicate via the Internet.

Java Remote Method Invocation (JavaRMI) [357] performs the object-oriented equivalent of RPC, with support for direct transfer of serialised Java classes and distributed garbage collection. Like CORBA and JavaRMI, .NET Remoting [331] allows an application to make an object available across remote boundaries, which includes different application domains, processes or even computers connected by a network.

11.3.1.3 Service-Oriented Architectures (SOAs)

Service-oriented architectures are a general pattern of structuring applications. Single parts of applications are modelled as independent reusable services. Nowadays, SOA is typically implemented using web technologies, but also other realisations of this concept exist.

Simple Object Access Protocol (SOAP) [152] is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses the XML information set for its message format, and relies on other application layer protocols, most notably hypertext transfer protocol (HTTP) or simple mail transfer protocol (SMTP), for message negotiation and transmission. A SOAP service typically provides a description of its methods and structures by means of a WSDL document (web service description language) [75]. This enables developer tools to generate client-side code for accessing a SOAP service automatically, which results in less error-prone and faster development.

However, the use of XML as transport envelope results in a large overhead in messages, thus making SOAP a technology rather unsuitable for embedded software. REST web-services (representational state transfer) [135] have proven to be a lightweight alternative to SOAP. REST uses standard HTTP methods such as GET, POST, PUT, DELETE to communicate to a remote server (i.e., GET is used to retrieve information, POST is used to send a certain payload, and so on). Methods to invoke are identified by URL paths. As an example, GETing *http://www.myservice.com/Items* would return a list of items available while POSTing to this URL may append one item. GETing *http://www.myservice.com/Items-/Item42* returns the item with index 42 while DELETEing would remove the corresponding data package. The payload in REST communication is typically serialised as XML or JSON data (Java Script Object Notation [86]). The advantage of REST is that nearly every connected device and platform has HTTP-accessing capabilities. Today, REST is the main communication paradigm used in smartphone apps.

11.3.1.4 Consequences of Host-Centric Middleware Systems for the Application Example

For a demonstrative explanation, we refer you to the example described in Section 11.1.2. Using a host-centric middleware system, the sensors need to be able to address the nodes they want to communicate with. If they want to share sensed data, they have to directly transmit them to a specific receiver. If synchronous communication is used, the sender is blocked until a response is received. This means that the sending node cannot continue sensing during message exchange. If for instance a PIR sensor wants to notify other parts of the system about detected motion, it needs to know the receivers and their addresses in advance. In this case, an application extension from cameras to cameras and audio sensors would explicitly affect the behaviour of PIR sensors.

The discovery of newly joining nodes is cost intensive, since those nodes have to distribute their address/location through the whole network and need to learn those of neighbouring nodes. Referring to our application example, supporting a changing smart environment within a room is cost-intensive (e.g., various devices are active at different times of the day, which is rather complicated to realise).

An RPC-based implementation of a sensor network first requires a definition of the functional interface of each node (i.e., the procedures available to be called by remote nodes). Further, the application needs mechanisms enabling nodes to find addresses of remote nodes. In a rather static network, this can be achieved by predefined host lists. Dynamic networks, in which nodes may join or leave, require a discovery mechanism or a centralised lookup registry (a previously known host where nodes can register on start-up and look up addresses of other nodes).

In an object-oriented implementation of our example sensor network, the middleware system first needs to define the object structures. Then the node addresses need to be resolved either by registry or pre-definition.

A sensor network which uses *REST* services in communication requires each sensor to run an HTTP server in order to receive incoming stimuli. A well-defined set of REST API calls has to be known already at design-time. To push a stimulus to multiple nodes, multiple HTTP requests have to be performed. This is hardly feasible for communication between all sensor nodes but may make sense in case multiple sensors report to the same sink node.

Summarised, host-centric systems were intended for synchronous operations which require sender and receiver to know each other and to block senders during data transfer. Further, synchronous operations support only one-to-one communication. Neighbour discovery is rather expensive as we can see on the example of REST, where services can only be accessed using fixed URLs. The benefit is the intuitive and transparent usage. Host-centric systems also support SACS in certain aspects. A detailed description of awareness requirements which are fulfilled by host-centric systems can be found in Section 11.3.3.

11.3.2 Content-Centric Middleware

In content-centric middleware systems, sending and receiving messages is not the central architectural viewpoint. Instead, the focus lies on accessing data independently of who is producing and who is consuming it. Typically, a data-oriented middleware system is used to transparently decouple content sources and content sinks (producers and consumers). While also host-centric paradigms are used to exchange data, they typically assume that the location of this data is known in advance (i.e., the URL of a SOAP service). In addition, they typically lack the capabilities to deal with multiple producers (i.e., multiple nodes would host a certain web service) compared to content-centric systems. Content-centric middleware systems can be realised in either a centralised or a distributed way.

11.3.2.1 Blackboard Systems

A blackboard system is a centralised content-centric middleware solution. Blackboard systems are distributed implementations of Linda spaces [8], where multiple processes exchange data via a centralised tuple matching system. In this approach, a producer provides its data by releasing it to a central point in the network (publishing the data on a blackboard). Consumers that are interested in that type of data can simply collect the data from the central point. A prominent example is the so-called tuple space. A tuple space can be interpreted as a distributed shared memory. It provides a repository of tuples that can be accessed concurrently. Producers post their data as tuples in the space, and the consumers then retrieve data from the space that match a certain pattern.

11.3.2.2 Message Queuing

Message queuing is another centralised host-centric middleware paradigm which uses the message queuing principle. Message queuing enables asynchronous communication and provides decoupling of senders and receivers, allowing them to be active at different times. Senders place their messages at a central point in a message queue, where they are stored until the receiver collects them. Since such a queue has restrictions on size, the number of messages in the queue and the size of messages are limited. Message queuing is a programming pattern allowing asynchronous applications in a distributed system to communicate with each other via message queues. A queue manager and the queues reside on a server node. Application nodes can send or receive messages only from message queues belonging to queue managers to which they are connected.

Message queue telemetry transport [26], known as MQTT, implements the message queuing pattern. MQTT is used for the transport of telemetry data and is well suited for wireless sensor networks or mobile-to-mobile communication since it is very lightweight [188]. In MQTT, the sender continuously transmits its sensor data to the message broker. Nodes which are interested in this kind of data collect it from the message broker. MQTT runs on connection-oriented transport (TCP).

11.3.2.3 Publish/Subscribe

Publish/subscribe [126] systems work in a similar way as blackboard systems do, especially if they are realised as a centralised solution. Publishers offer data and subscribers show interest in specific data by subscribing to that type of data. Thereby, publishers publish their data without any knowledge of the sink, and subscribers consume data without knowledge of the source. However, an important difference with blackboard systems is that in publish/subscribe the middleware system pushes data to subscribers whereas in a blackboard they have to pull data on demand. Publish/subscribe can be realised in a centralised as well as a distributed manner. In the centralised solution, a central unit is in charge of matching publishers with subscribers. The distributed solution does not need any central node to publish the data to or consume the data from. The middleware system is in charge of delivering the data to subscribers of matching data type. A detailed explanation of publish/subscribe can be found in Section 11.4.

11.3.2.4 Consequences of Content-Centric Middleware Systems for the Application Example

Referring to the example in Section 11.1.2, a sensor network based on the blackboard concept or on a message queue requires a central node. The blackboard system or the message queues and the queue manager would reside on this central node, which is responsible for delivering messages or data to the appropriate re-

ceivers. Thus, every node within the network has to communicate with the central node if it wants to interact with other nodes of the network. The central node may thus become a bottleneck of the system. In the case of message queuing, there has to be a queue for PIR notifications, a queue for camera images, a queue for authorisation information and a queue for user information. If the application is extended to more devices, they can use the already existing queues as input and define their own queues as output.

In publish/subscribe, there may or may not be a central entity, depending on the implementation. In any case, nodes only need to indicate their interest in specific data in order to start receiving it. However, without additional meta-data or middleware functionality, the node may not be able to distinguish between data of the same type received from different publishers. As an example, devices which want to process user information subscribe to this type of data and the middleware system will take care of finding suitable publishers and data transport.

Summarised, in content-centric middleware systems the focus lies on the data itself rather than on the data source or sink. This enables decoupling of data producers and consumers. Further, communication can be done in a one-to-one or one-to-many manner, since the data producers are not interested in who is consuming the data. As with host-centric systems, content-centric systems also support several SACS aspects, which can be found in Section 11.3.3.

11.3.3 Requirements Conformity of Middleware Paradigms

Table 11.2 shows which awareness requirements are fulfilled by the different types of middleware systems.

All middleware paradigms support stimulus-awareness whenever they push data into an application module. Using host-centric middleware, such as RPC, OOM and SOA, nodes have to directly address other nodes in order to push a new stimulus. In content-centric middleware concepts, nodes push their stimuli to the network by offering it to other nodes. Nodes which are interested in the information can then collect it. Since one of the main features of middleware systems is to provide communication and data exchange facilities to distributed applications, this makes them inherently interaction-aware.

Additionally to message exchange, a middleware system also enables nodes to express their selves. Host-centric paradigms explicitly model the data flow (they directly address each host) and can thus easily change this flow based on the self-awareness state. Data-centric approaches provide facilities to change the data produced and consumed by application modules and thus enable a change in the application behaviour. This means both types of middleware support applications in changing their behaviour based on the self-awareness state and are thus self-expressive.

Host-centric middleware solutions are not time-aware, since they typically do not care about time-stamping, temporal ordering of events or time-decoupling. How-

ever, content-centric solutions can support time-awareness in different ways. Publish/subscribe and message queuing systems can support time-awareness. In both paradigms, messages can be buffered in queues. This means they can be delivered in right intervals and/or in temporal order. A similar explanation holds for blackboard systems, since the data is stored at a central point where it can be buffered or time-stamped.

Interaction-awareness is supported by all host-centric middleware systems, since they need to know the host they are communicating with. In content-centric middleware systems, senders and receivers do not know each other, since they are only interested in the data which is exchanged. Without knowing other nodes, a sender is not able to interact with one specific node. Thus, content-centric solutions do not directly support interaction-awareness. To enable the application to participate in interactions with other application parts, a content-centric middleware system requires an additional communication channel. This channel could then be used by nodes to interact with specific neighbours.

Table 11.2 Support for awareness primitives in various middleware paradigms

SA/SE Primitive	RPC	OOM	SOA	Blackboard	Message queuing	Pub/Sub
stimulus-aware	✓	✓	✓	✓	✓	✓
interaction-aware	✓	✓	✓	✗	✗	✗
time-aware	✗	✗	✗	✓	✓	✓
self-expression	✓	✓	✓	✓	✓	✓

11.4 Publish/Subscribe

In this section we present the publish/subscribe paradigm in more detail to show how it can support SACS.

The publish/subscribe paradigm is a type of content-centric middleware, defining two different roles: a publisher and a subscriber. Publishers are modules which produce data (e.g., capturing images or fetching data from a specific source) to then pass it on to consumers, the so-called subscribers, which process the data. A module may be publisher and subscriber at the same time, i.e., it processes data from another publisher and publishes the results itself.

Publish/subscribe is a mechanism which allows for elegant decoupling of functional elements within applications. A component for publish/subscribe management takes care of the decoupling. Instead of directly connecting the publisher and subscriber modules, a publisher announces its events and a subscriber can announce interest in certain types of events. The publish/subscribe management takes care of matching published events and subscriptions and is also responsible for delivering

the published data to all interested subscribers. This means that publishers provide their data to the network, rather than sending their events to specific receivers.

A key requirement of publish/subscribe is that neither publishers nor subscribers need to be aware of each other. A publisher does not need to keep track of where its data is sent and how many subscribers exist for its events, and a subscriber does not need to care about where publishers are located and where their data is coming from (i.e., the local node or a remote node). All this is transparently handled by the publish/subscribe middleware system.

In the case of a centralised publish/subscribe solution, the publish/subscribe management resides on the central node. We prefer a decentralised solution since centralised networks are hardly scalable. Figure 11.3 illustrates a distributed publish/-subscribe architecture, where a small publish/subscribe management component resides on each node in the network and is responsible for its local modules and for connecting them to remote nodes.

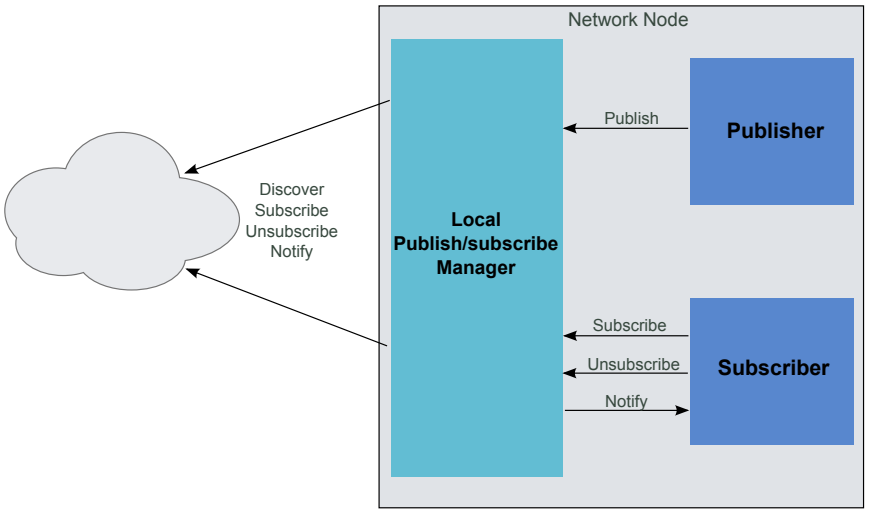

Fig. 11.3 A simple distributed publish/subscribe system. Each node keeps track of other nodes and subscriptions in the network.

The publish/subscribe management is responsible for notifying subscribers whenever data of their interest is published. To enable these notifications in this many-to-many interaction, publish/subscribe relies on asynchronous communication. The messaging model of publish/subscribe systems is very powerful, connecting senders and receivers of messages anonymously. It provides many-to-many and one-to-many interaction, enabling a sender to send its messages to either one or multiple receivers. There is no restriction on the role of a node within a publish/subscribe system. Each node can be a publisher, a subscriber or both.

There are three methods that are typically provided by a publish/subscribe middleware system: subscribe(), unsubscribe() and publish(). The subscriber calls the subscribe() method to show its interest in specific event data and the unsubscribe() method whenever it is no longer interested in this data. The publish() method is called by publishers to notify that there is new event data ready to be offered to subscribers. To enable the matching of published events and subscriptions without the need for the source to know the sink and vice versa, different subscription schemes had been defined. Those are explained in the following section.

11.4.1 Publish/Subscribe Flavours

Since subscribers are usually interested in some specific events, rather than in all events, a way of defining different events is required. Three different schemes can be used to define event classes, typically called subscription schemes.

11.4.1.1 Topic-Based Publish/Subscribe

Events in a topic-based scheme are defined by specific keywords, e.g., the name of the topic. This means that publishers publish events of a particular topic and all subscribers interested in this topic receive these events. The communication between publishers and subscribers is done via messages, including the command, the topic name and the event data. Thus, each topic can be seen as event service, offering a publish() and subscribe() method for its own specific topic. Topic-based publish/-subscribe is a rather static scheme with limited expressiveness, but can be efficiently implemented [126].

11.4.1.2 Content-Based Publish/Subscribe

Events in a content-based scheme are defined by its content, e.g., its properties. This means that subscribers subscribe to content (properties of an object) they would like to receive rather than to a topic name. Events are thus not predefined by keywords, which makes the scheme dynamic. Properties can be internal attributes or meta-data related to events. Subscribers subscribe to an event by typically using a key-value pair. Furthermore, some systems provide a mechanism for event correlation. In these systems, subscribers can subscribe themselves to different combinations of events and are only notified by the event service if this combination of events is provided by the publishers.

In the case of content-based publish/subscribe, the event service provides the subscribe() method with an additional argument. This argument defines the content to be subscribed to and can be either a string or a template object. By using template objects, the subscriber provides an object that it is interested in and the event service

notifies the subscriber only on occurrence of events of the same properties. The advantage over topic-based publish/subscribe is that events are no longer bound to fix criteria such as the topic name and that content-based publish/subscribe is highly expressive.

11.4.1.3 Type-Based Publish/Subscribe

In a type-based scheme, subscribers express their interest in events by subscribing to the type of an object. This scheme guarantees type safety at compile-time and encapsulation, which is an advantage in implementation simplicity over the other two schemes. By using public class members, type-based publish/subscribe would transform into content-based publish/subscribe. However, in type-based publish/-subscribe, private class members, which can only be accessed through public methods, are used.

11.4.2 Decoupling

A publish/subscribe system enables decoupling in the following dimensions:

- Space decoupling: Modules do not need to know where they and other modules are located in the network. This means that publishers do not hold any references to subscribers and vice versa. In a visual sensor network (VSN), a publisher of images does not need to care if they are delivered to one or more displays or other modules.
- Time decoupling: Publishers and subscribers do not need to participate in an interaction at the same time. The publisher might for instance publish an event while there is no subscriber connected. Publishers which start after a subscriber can still be matched to an earlier subscription request. In a VSN, cameras may not start up at the same time; still, they must form one distributed application.
- Synchronisation decoupling: Preparing events does not block the publishers, and subscribers can be notified of an event even though they are currently executing another activity. As an example, a publisher of images can capture the next image while the current image is still being delivered to subscribers.

11.4.3 Publish/Subscribe for SACS

Publish/subscribe enables self-awareness and self-expression, making this paradigm very suitable for SACS. As already described above, publish/subscribe supports an application in being stimulus-aware, time-aware and self-expressive.

Subscriptions of data consumers enable private and public self-awareness. Nodes perceive their environment and react properly by subscribing to specific information

and data. In tracking, for instance, a camera recognises whenever the object is going to leave the FOV (private self-awareness). The camera may then decide to hand over tracking responsibility to a neighbouring camera (public self-awareness).

By publishing data, self-expression is achieved. Since the publisher forgets about the data as well as the publishing process after publishing, we call it fire-and-forget self-expression.

11.5 Ella: A Publish/Subscribe-Based Hybrid Middleware

In the previous sections we have seen how different middleware paradigms can support different aspects of SACS. Of course, an application does not need to miss self-awareness features which are not supported by the middleware. However, they need to be implemented on the application layer.

In this section, we present a middleware implementation called Ella [100] which is available as an open source project[1]. Ella is designed as a hybrid middleware system combining features of host-centric and content-centric paradigms in order to provide an improved basis for self-aware applications. Ella in its basics is a distributed publish/subscribe middleware system. However, additional features have been added to allow for interaction- and self-awareness. Ella in its core builds on decoupled modules which do not have direct references to each other. This modularised architecture enables application modules to be exchanged at runtime.

11.5.1 Architecture

In this section, we present the architecture of the Ella middleware system. While there are several paradigms for middleware systems, like RPC or OOM, we chose a data-driven publish/subscribe approach. This allows for a high degree of flexibility since it provides decoupling in space and time. In contrast to other publish/subscribe implementations such as [388], Ella is completely distributed without the need for any central coordination. The publish/subscribe implementation of Ella is explained in the following sections.

11.5.1.1 Subscribing

Ella uses type-based subscriptions, which means that a subscriber specifies a certain data type to subscribe to. Additionally, Ella provides the possibility to request a template object. The middleware system will then ask each publisher (which is matching in type) to generate such a template object and will hand it to the prospec-

[1] https://ella.codeplex.com/

tive subscriber. The subscriber can then decide whether this specific publisher is accepted or not.

In addition, upon subscribing, the subscriber can decide to exclude remote publishers to obtain only subscriptions from the local node. This can be useful whenever node-specific information is requested, such as the current resource allocation.

Optionally, a subscriber can provide a callback method which is called whenever it is subscribed to a new publisher. The subscriber will be provided with a handle object which can be used to distinguish between multiple publishers but without making them identifiable.

11.5.1.2 Publishing

A publisher publishes events of a certain data type, which is then matched to subscribers by Ella. Similarly to subscribers, publishers can also provide callback methods which are called by Ella for each new subscriber to a certain event. Publishers can use this information to publish certain information to only a subset of subscribers. As an example, some subscribers to images of a camera may send an application message to the publisher asking for a reduced frame rate. The publisher can then publish at a lower rate to those subscribers which have requested this.

11.5.1.3 Realisation of Publishers and Subscribers

Creating a publisher or a subscriber for Ella is a straightforward task. Instead of forcing developers to implement a specific class hierarchy (i.e., subclassing a base class or implementing an interface), Ella uses code annotations to declare a certain class to be a publisher or a subscriber. These code annotations can be reflected by Ella at runtime to detect publishers and subscribers in code libraries. This approach has several advantages. First, it makes it very easy to adapt existing code to run on top of Ella. It basically requires annotating the existing code and adapting the way of data passing to the mechanism provided by Ella. Second, it makes the development of modules more flexible because developers are not bound to any inheritance hierarchy and thus it is easier to integrate Ella into any software architecture. Third, Ella-based code is easily readable and maintainable because the annotations directly inform about what the specific module does.

11.5.1.4 Network Management and Remote Operation

To support a convenient way of developing and deploying software modules for Ella, the middleware system provides a transparent node discovery mechanism which is used to detect any running Ella instances on other nodes in the network. This relieves the developer of the need for managing other nodes in the network. As soon as an Ella instance is detected, it is registered as a known host and it will also be checked

for suitable publishers of events requested by local subscribers. With this approach, it is much easier to scale an existing application without having to modify existing code. As soon as Ella detects other instances, it will include them in its operation.

On start-up, Ella tries to first discover other nodes in the network. By default discovery is realised with a UDP broadcast. This broadcast also contains connection information necessary to address this node in the network. However, this may be exchanged with any other suitable discovery provider (e.g., for non-IP compatible media, like ZigBee). Upon reception of a broadcast message, a node will send a unicast answer to the broadcasting node with its own connection information. Thus, each node keeps a local directory of known remote hosts. This directory is used when searching for matching publishers on other nodes.

Whenever a subscriber requests a new subscription, all remote hosts will be queried about matching publishers. If any matches are found, proxy objects at the remote node and stubs at the subscriber node will be created which act as transparent transport points for published event data. A proxy acts as subscriber at the remote node, serialises the event data and sends it to the stub. The stub deserialises it and publishes it as a local publisher for the original subscriber to receive.

The requested subscription types by each subscriber module are cached by the local Ella instance. Whenever a new node is discovered in the network which runs suitable publishers, they will start to deliver their events to the local node. In addition, if a publisher has delayed its start, the node sends out a notification to other nodes that a potential publisher has become available for their subscribers. These two functionalities enable soft time decoupling and do not require publishers and subscribers to start at the same time.

11.5.1.5 Communication

Ella instances on remote nodes use an efficient message structure to exchange data. A binary protocol is used to encode message types and to transport any necessary data. For any given message payload, only nine bytes of overhead are added, one byte for the message type, and four bytes each for the sender node ID and the message ID. For small networks this can be reduced by only using single bytes for the sender node ID.

Besides data communication, Ella provides also a unicast control channel between publishers and subscribers. This can be used to exchange application-specific messages as in RPC systems. Instead of transporting data, control commands can be sent between publishers and subscribers. As an example, an image processing module could instruct the image capturing module to adapt its frame rate. This unicast channel was introduced to enable support of interaction–awareness (see Section 11.2), for which nodes have to know each other in order to enable participation in interactions with specific neighbours.

11.5.1.6 Implementation Details

Ella has been developed in C#.Net. It is capable of running in the open source Mono[2] runtime and can thus be deployed on all major operating systems and many other platforms. Since it is only performing high-level tasks like I/O and management of subscriptions, its overhead compared to a native implementation is very low. In addition, it is easily possible to integrate native code components into any .Net application. Thus, performance-critical applications parts can be written in C++ and be integrated into Ella with low effort. Of course, pre-existing native code can be integrated as well.

Subscription Handling:
On each node, Ella keeps a list of all subscriptions relevant to this node, i.e., all subscriptions where modules of this node are publishers and/or subscribers (this is also true for subscriptions only on the local nodes). Whenever a publisher publishes a new event, all subscribers are found in this list. In the simplest case, this data is delivered to a local subscriber (which is on the same node). For remote subscribers, this list contains the proxy at the publisher node. A proxy serialises the event data and sends it to the receiver node. There, a stub reconstructs the data and publishes it locally for the intended subscriber to receive it. In cases where unreliable transport can be used to deliver data (i.e., where loss of data can be tolerated), a UDP multicast mechanism can be used in order to save communication costs.

Event Correlation:
In some cases, a user might want to indicate that two events are somehow correlated. For example, the image of a camera and the result of a tracking algorithm might correspond to each other. For this case, Ella provides a simple mechanism where a publisher can indicate such a correspondence. This is then delivered to all modules which subscribed to both events. This is also part of Ella's interaction-awareness support capabilities because it enables nodes to handle correlations of the stimuli they are emitting.

11.5.2 SACS-Specific Features in Ella

Ella provides several useful services to an application. It handles discovery and communication, provides a control channel and helps in modularising an application. It enables flexible reconfiguration of an application with its module-based architecture. The use of code annotations to declare Ella-specific code regions makes it very easy for developers to port their existing code. The transparent subscription mechanism of publish/subscribe enables decoupling in space and time and the synchronisation of modules.

[2] http://www.go-mono.org

Ella provides specific features for self-awareness and self-expression in various aspects described in Table 11.1. In an application which uses Ella, the nodes can communicate without any concerns on the communication channel. The middleware system cares about message and data delivery and additionally enables application-specific 1:1 messages.

Further, Ella enables context-awareness, reacting properly when the network is congested. It informs the appropriate publisher that it is congesting the network. The application developer can than decide what to do with this information. In the case of a VSN application, the cameras may reduce the resolution or the frame rate to achieve a continuous and reliable image stream without congesting the network.

Ella inherently deals with modularised applications since it connects individual modules by address-free communication mechanisms. Hence, a module does not need to have any reference to another module in order to exchange data. In addition, modules can start and terminate arbitrarily during application runtime. This not only increases robustness to module failures, it also enables the exchange of specific modules at runtime.

11.5.3 Ella in Practice

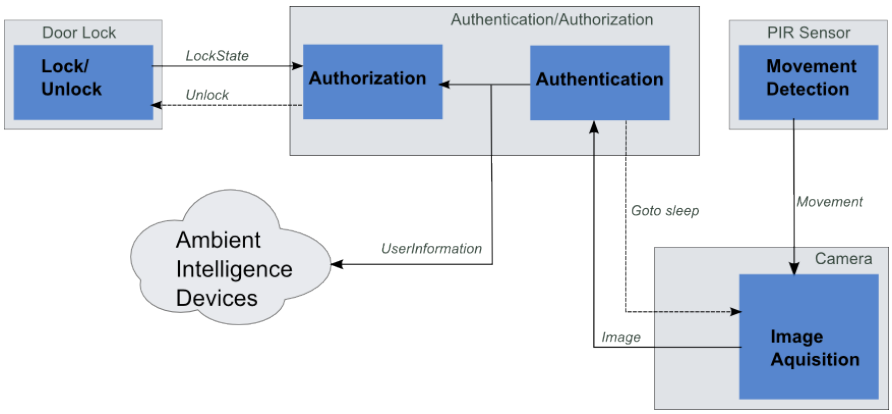

Fig. 11.4 The networked application consisting of PIR sensors, cameras, an authentication/authorisation module, door locks and ambient intelligence devices. Solid lines indicate published data, dotted lines indicate one-to-one messages.

Using the example introduced in Section 11.1.2 we now show how a self-aware application can be realised using Ella. We assume all components of the system to be on individual nodes. Figure 11.4 shows the architecture of the application.

PIR sensors act as pure publishers. Whenever a PIR sensor detects motion it will publish a new event of type *Movement*.

Cameras subscribe to this type of data. As soon as a camera receives this signal, it will start to record and publish the data type *Image*.

The authentication/authorisation module is subscribed to *Image* and performs face recognition. When it has received enough images from a camera, it sends a unicast message back to the corresponding camera telling it to go back into sleep mode and wait for the next *Movement*.

The lock in the door continuously publishes *LockState* information indicating if it is closed or open. The authentication module sends a control message to the lock to open it for an authorised user. Further, it publishes *UserInformation* to all devices which are concerned with adapting to specific user needs. Ambient intelligence devices like lighting controls, tablets, displays and others are subscribed to this data type and use the received events to change their state or display specific information.

Using Ella in this application brings various advantages. First, the application is very flexible in terms of data sources for stimuli input. As an example, also other sensor types may publish *Movement* events which activate cameras. This may be pressure mats on the floor or acoustic sensors. They can be integrated into the system without changing the application. The same holds true for bringing new subscribers into the system, a dedicated application part which consumes published data for archiving purposes.

Second, the possibility to use control messages in addition to publishing data enables feedback channels to publishers and interaction-aware behaviour of the application. The authentication/authorisation module can use this feature to disable a specific subset of cameras after an authentication fails and thus may record additional images of unauthorised persons.

Third, the subscription callback mechanism can be used by modules to dynamically adapt their behaviour. As an example, the authentication module may also subscribe to *Fingerprint* data. If it receives a subscription callback it knows that there is a fingerprint reader in the system. Upon identifying a person it can then wait for fingerprint information before authorising him.

11.6 Conclusion

This chapter provided an overview of distributed self-aware computing systems and and how middleware systems can support their development. Distributed SACS have additional requirements concerning communication, robustness and scalability as well as architectural primitives of self-aware applications.

A categorisation of middleware systems into host-centric and content-centric systems and their abilities to support SACS were presented. The publish/subscribe paradigm was discussed as a solid basis for a self-aware application since it enables decoupling of components along with scalability and robustness. However, it does not support all self-awareness aspects that can be emphasised by a middleware system.

We introduced Ella, a hybrid middleware system based on a distributed publish/-subscribe implementation and enhanced with additional concepts. Ella combines the best of different middleware paradigms in terms of self-awareness support. For our discussion, we used an example application to describe a possible implementation of a distributed SACS.

Acknowledgements

The research leading to these results was conducted in the EPiCS project (Engineering Proprioception in Computing Systems) and received funding from the European Union Seventh Framework Programme under grant agreement no. 257906.

The contributors would like to acknowledge additional financial support for research performed in individual chapters of this book.

- Chapters 6 and 7 were also supported by EPSRC Grants (Nos. EP/I010297/1, EP/K001523/1 and EP/J017515/1).
- Chapter 8 was also supported by the German Research Foundation (DFG) within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901) and the International Graduate School on Dynamic Intelligent Systems of Paderborn University.
- Chapter 9 was also supported in part by HiPEAC NoE, by the European Union Seventh Framework Programme under grant agreement numbers 287804 and 318521, by the UK EPSRC, by the Maxeler University Programme, and by Xilinx.
- Chapter 12 was also supported in part by the China Scholarship Council, by the European Union Seventh Framework Programme under grant agreement numbers 287804 and 318521, by the UK EPSRC, by the Maxeler University Programme, and by Xilinx.
- Chapter 13 was also supported by the research initiative Mobile Vision with funding from the Austrian Institute of Technology and the Austrian Federal Ministry of Science, Research and Economy HRSMV programme BGBl. II no. 292/2012.
- Chapter 14 was also supported by the Research Council of Norway under grant agreement number 240862/F20.
- Peter Lewis would like to thank the participants of the Dagstuhl Seminar "Model-Driven Algorithms and Architectures for Self-aware Computing Systems", Seminar Number 15041, for many insightful discussions on notions of self-aware computing.

References

1. Aberdeen, D., Baxter, J.: Emmerald: a fast matrix-matrix multiply using Intel's SSE instructions. Concurrency and Computation: Practice and Experience **13**(2), 103–119 (2001)
2. Abramowitz, M., Stegun, I.: Handbook of Mathematical Functions. Dover Publications (1965)
3. Agarwal, A., Harrod, B.: Organic computing. Tech. Rep. White paper, MIT and DARPA (2006)
4. Agarwal, A., Miller, J., Eastep, J., Wentziaff, D., Kasture, H.: Self-aware computing. Tech. Rep. AFRL-RI-RS-TR-2009-161, MIT (2009)

5. Agne, A., Hangmann, H., Happe, M., Platzner, M., Plessl, C.: Seven recipes for setting your FPGA on fire – a cookbook on heat generators. Microprocessors and Microsystems **38**(8), 911–919 (2014). DOI 10.1016/j.micpro.2013.12.001
6. Agne, A., Happe, M., Keller, A., Lübbers, E., Plattner, B., Platzner, M., Plessl, C.: ReconOS: An Operating System Approach for Reconfigurable Computing. IEEE Micro **34**(1), 60–71 (2014). DOI 10.1109/MM.2013.110
7. Agne, A., Platzner, M., Lübbers, E.: Memory virtualization for multithreaded reconfigurable hardware. In: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp. 185–188. IEEE Computer Society (2011). DOI 10.1109/FPL.2011.42
8. Ahuja, S., Carriero, N., Gelernter, D.: Linda and friends. IEEE Computer **19**(8), 26–34 (1986). DOI 10.1109/MC.1986.1663305
9. Al-Naeem, T., Gorton, I., Babar, M.A., Rabhi, F., Benatallah, B.: A quality-driven systematic approach for architecting distributed software applications. In: Proceedings of the 27th International Conference on Software Engineering, pp. 244–253. ACM (2005). DOI 10.1145/1062455.1062508. URL http://doi.acm.org/10.1145/1062455.1062508
10. Ali, H.A., Desouky, A.I.E., Saleh, A.I.: Studying and Analysis of a Vertical Web Page Classifier Based on Continuous Learning Naive Bayes (CLNB) Algorithm, pp. 210–254. Information Science (2009)
11. Alippi, C., Boracchi, G., Roveri, M.: Just-in-time classifiers for recurrent concepts. IEEE Transactions on Neural Networks and Learning Systems **24**(4), 620–634 (2013)
12. Amir, E., Anderson, M.L., Chaudhri, V.K.: Report on DARPA workshop on self-aware computer systems. Tech. Rep. UIUCDCS-R-2007-2810, UIUC Comp. Sci. (2007)
13. ANA: Autonomic Network Architecture. URL www.ana-project.org. (accessed March 8, 2016)
14. Angelov, P.: Nature-inspired methods for knowledge generation from data in real-time (2006). URL http://www.nisis.risk-technologies.com/popup/Mallorca2006_Papers/A333_13774_Nature-inspiredmethodsforKnowledgeGeneration_Angelov.pdf
15. Apache: Hadoop. http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html. (Accessed March 8, 2016)
16. Araya-Polo, M., Cabezas, J., Hanzich, M., Pericàs, M., Rubio, F., Gelado, I., Shafiq, M., Morancho, E., Navarro, N., Ayguadé, E., Cela, J.M., Valero, M.: Assessing accelerator-based HPC reverse time migration. IEEE Transactions on Parallel and Distributed Systems **22**(1), 147–162 (2011)
17. Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W., Yelick, K.A.: The landscape of parallel computing research: A view from Berkeley. Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley (2006)
18. Asendorpf, J.B., Warkentin, V., Baudonnière, P.M.: Self-awareness and other-awareness. II: Mirror self-recognition, social contingency awareness, and synchronic imitation. Developmental Psychology **32**(2), 313 (1996)
19. Athan, T.W., Papalambros, P.Y.: A note on weighted criteria methods for compromise solutions in multi-objective optimization. Engineering Optimization **27**(2), 155–176 (1996)
20. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2–3), 235–256 (2002)
21. Babaoglu, O., Binci, T., Jelasity, M., Montresor, A.: Firefly-inspired heartbeat synchronization in overlay networks. In: First International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 77–86 (2007)
22. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(8), 1619–1632 (2011)
23. Bader, J., Zitzler, E.: HypE: an algorithm for fast hypervolume-based many-objective optimization. Tech. Rep. TIK 286, Computer Engineering and Networks Laboratory, ETH Zurich, Zurich (2008)

24. Baena-García, M., Campo-Ávila, J.D., Fidalgo, R., Bifet, A.: Early drift detection method. In: Procedings of the 4th ECML PKDD International Workshop on Knowledge Discovery From Data Streams (IWKDDS), pp. 77–86. Berlin, Germany (2006)
25. Baker, S.: The identification of the self. Psyc. Rev. **4**(3), 272–284 (1897)
26. Banks, A., Gupta, R.: MQTT Version 3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html (2014)
27. Bartolini, D.B., Sironi, F., Maggio, M., Cattaneo, R., Sciuto, D., Santambrogio, M.D.: A Framework for Thermal and Performance Management. In: Proceedings of the Workshop on Managing Systems Automatically and Dynamically (MAD) (2012)
28. Basheer, I.A., Hajmeer, M.: Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods **43**(1), 3–31 (2000)
29. Basseur, M., Zitzler, E.: Handling uncertainty in indicator-based multiobjective optimization. International Journal of Computational Intelligence Research **2**(3), 255–272 (2006)
30. Basudhar, A., Dribusch, C., Lacaze, S., Missoum, S.: Constrained efficient global optimization with support vector machines. Structural and Multidisciplinary Optimization **46**(2), 201–221 (2012)
31. Baumann, A., Boltz, M., Ebling, J., Koenig, M., Loos, H.S., Merkel, M., Niem, W., Warzelhan, J.K., Yu, J.: A review and comparison of measures for automatic video surveillance systems. EURASIP Journal on Image and Video Processing **2008**(4) (2008). DOI 10.1155/2008/824726
32. Becker, T., Agne, A., Lewis, P.R., Bahsoon, R., Faniyi, F., Esterle, L., Keller, A., Chandra, A., Jensenius, A.R., Stilkerich, S.C.: EPiCS: Engineering proprioception in computing systems. In: Proceedings of the International Conference on Computational Science and Engineering (CSE), pp. 353–360. IEEE Computer Society (2012)
33. Ben-Hur, A., Weston, J.: A user's guide to support vector machines. Data Mining Techniques for the Life Sciences **609**, 223–239 (2010)
34. Betts, A., Chong, N., Donaldson, A.F., Qadeer, S., Thompson, P.: GPUVerify: a verifier for GPU kernels. In: Proceedings of the ACM International Conference on object-oriented Programming Systems Languages and Applications (OOPSLA) (2012)
35. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal on Operational Research **181**(3), 1653–1669 (2007)
36. Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guédy, F., Rasamimanana, N.: Continuous realtime gesture following and recognition. In: Gesture in embodied communication and human-computer interaction, pp. 73–84. Springer (2010)
37. Biehl, J.T., Adamczyk, P.D., Bailey, B.P.: Djogger: A mobile dynamic music device. In: Proceedings of CHI '06 Extended Abstracts on Human Factors in Computing Systems, pp. 556–561. ACM (2006)
38. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, United Kingdom (2005)
39. Bojic, I., Lipic, T., Podobnik, V.: Bio-inspired clustering and data diffusion in machine social networks. In: Computational Social Networks, pp. 51–79. Springer (2012)
40. Bongard, J., Lipson, H.: Evolved machines shed light on robustness and resilience. Proceedings of the IEEE **102**(5), 899–914 (2014)
41. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science **314**(5802), 1118–1121 (2006)
42. Borkar, S.: Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. IEEE Micro pp. 10–16 (2005)
43. Bouabene, G., Jelger, C., Tschudin, C., Schmid, S., Keller, A., May, M.: The Autonomic Network Architecture (ANA). IEEE Journal on Selected Areas in Communications **28**(1), 4–14 (2010). DOI 10.1109/JSAC.2010.100102
44. Boyd, J.: The Essence of Winning and Losing. http://dnipogo.org/john-r-boyd/ (1996). (Accessed March 8, 2016)
45. Bramberger, M., Doblander, A., Maier, A., Rinner, B., Schwabach, H.: Distributed Embedded Smart Cameras for Surveillance Applications. IEEE Computer **39**(2), 68–75 (2006)

46. Brdiczka, O., Crowley, J.L., Reignier, P.: Learning situation models in a smart home. IEEE Transactions on Systems, Man, and Cybernetics, Part B **39**, 56–63 (2009)
47. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
48. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
49. Brockhoff, D., Zitzler, E.: Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 2086–2093 (2007)
50. Buchanan, J.T.: A naive approach for solving MCDM problems: The GUESS method. Journal of the Operational Research Society **48**(2), 202–206 (1997)
51. Buck, J.: Synchronous rhythmic flashing of fireflies. The Quarterly Review of Biology **13**(3), 301–314 (1938)
52. Buck, J.: Synchronous rhythmic flashing of fireflies II. The Quarterly Review of Biology **63**(3), 265–289 (1988)
53. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society **206**(1), 241–264 (2013)
54. Buschmann, F., Henney, K., Douglas, S.C.: Pattern-oriented software architecture: On patterns and pattern languages. John Wiley and Sons (2007)
55. Buss, A.H.: Self-consciousness and social anxiety. W. H. Freeman, San Fransisco, CA, USA (1980)
56. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verication at runtime. Communications of the ACM **55**(9), 69–77 (2012)
57. Caramiaux, B., Wanderley, M.M., Bevilacqua, F.: Segmenting and parsing instrumentalists' gestures. Journal of New Music Research **41**(1), 13–29 (2012)
58. Carver, C.S., Scheier, M.: Attention and Self-Regulation: A Control-Theory Approach to Human Behavior. Springer (1981)
59. de Castro, L.N.: Fundamentals of natural computing: basic concepts, algorithms, and applications. Chapman & Hall/CRC Computer and Information Sciences (2006)
60. Chandra, A.: A methodical framework for engineering co-evolution for simulating socio-economic game playing agents. Ph.D. thesis, The University of Birmingham (2011)
61. Chandra, A., Nymoen, K., Volsund, A., Jensenius, A.R., Glette, K., Torresen, J.: Enabling participants to play rhythmic solos within a group via auctions. In: Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR), pp. 674–689 (2012)
62. Chandra, A., Yao, X.: Ensemble learning using multi-objective evolutionary algorithms. Journal of Mathematical Modelling and Algorithms **5**(4), 417–445 (2006)
63. Chang, C., Wawrzynek, J., Brodersen, R.W.: BEE2: a high-end reconfigurable computing system. IEEE Transactions on Design & Test of Computer **22**(2), 114–125 (2005)
64. Chen, J., John, L.K.: Efficient program scheduling for heterogeneous multi-core processors. In: Proceedings of the Design Automation Conference (DAC). ACM (2009)
65. Chen, R., Lewis, P.R., Yao, X.: Temperature management for heterogeneous multi-core FPGAs using adaptive evolutionary multi-objective approaches. In: Proceedings of the International Conference on Evolvable Systems (ICES), pp. 101–108. IEEE (2014)
66. Chen, S., Langner, C.A., Mendoza-Denton, R.: When dispositional and role power fit: implications for self-expression and self-other congruence. Journal of Personality and Social Psychology **96**(3), 710–27 (2009)
67. Chen, T., Bahsoon, R.: Self-adaptive and Sensitivity-aware QoS Modeling for the Cloud. In: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 43–52. IEEE (2013). URL http://dl.acm.org/citation.cfm?id=2487336.2487346
68. Chen, T., Bahsoon, R.: Symbiotic and Sensitivity-aware Architecture for Globally-optimal Benefit in Self-adaptive Cloud. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 85–94. ACM (2014). DOI 10.1145/2593929.2593931. URL http://doi.acm.org/10.1145/2593929.2593931

69. Chen, T., Bahsoon, R., Yao, X.: Online QoS Modeling in the Cloud: A Hybrid and Adaptive Multi-learners Approach. In: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC), pp. 327–336 (2014)
70. Chen, T., Faniyi, F., Bahsoon, R., Lewis, P.R., Yao, X., Minku, L.L., Esterle, L.: The handbook of engineering self-aware and self-expressive systems. Tech. rep., EPiCS EU FP7 project consortium (2014). URL http://arxiv.org/abs/1409.1793. Available via EPiCS website and arXiv
71. Chen, X., Li, X., Wu, H., Qiu, T.: Real-time Object Tracking via CamShift-based Robust Framework. In: Proceedings of the International Conference on Information Science and Technology (ICIST). IEEE (2012)
72. Chow, G.C.T., Grigoras, P., Burovskiy, P., Luk, W.: An efficient sparse conjugate gradient solver using a Beneš permutation network. In: Proceedings of the 24th International Conference on Field Programmable Logic and Applications, pp. 1–7 (2014)
73. Chow, G.C.T., Tse, A.H.T., Jin, Q., Luk, W., Leong, P.H.W., Thomas, D.B.: A mixed precision Monte Carlo methodology for reconfigurable accelerator systems. In: Proceedings of the ACM/SIGDA 20th International Symposium on Field Programmable Gate Arrays, FPGA 2012, Monterey, California, USA, February 22-24, 2012, pp. 57–66 (2012)
74. Christensen, A.L., O'Grady, R., Dorigo, M.: From fireflies to fault-tolerant swarms of robots. IEEE Transactions on Evolutionary Computation **13**(4), 754–766 (2009)
75. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. World Wide Web Consortium (2001)
76. Chu, F., Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. In: Proceedings of the Eighth Pacific-Asia Knowledge Discovery and Data Mining Conference (PAKDD), pp. 282–292. Sydney (2004)
77. Cichowski, A., Madden, C., Detmold, H., Dick, A., Van den Hengel, A., Hill, R.: Tracking Hand-off in Large Surveillance Networks. In: Proceedings of the International Conference Image and Vision Computing, pp. 276–281. IEEE Computer Society Press (2009). DOI 10.1109/IVCNZ.2009.5378396
78. Claus, C., Boutilier, C.: The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: Proceedings of the Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, pp. 746–752. American Association for Artificial Intelligence (1998)
79. Collins, N.: The analysis of generative music programs. Organised Sound **13**, 237–248 (2008)
80. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. IEEE Transactions on Pattern Analysis and Machine Intelligence **27**(10), 1631–1643 (2005). DOI 10.1109/tpami.2005.205
81. Colorni, A., Dorigo, M., Maniezzo, V., et al.: Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life, vol. 142, pp. 134–142. Elsevier (1991)
82. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(5) (2003). DOI 10.1109/tpami.2003.1195991
83. Connors, K.: Chemical kinetics: the study of reaction rates in solution. VCH Publishers (1990)
84. Cox, M.: Metacognition in computation: A selected research review. Artificial Intelligence **169**(2), 104–141 (2005)
85. Cramer, T., Schmidl, D., Klemm, M., an Mey, D.: OpenMP Programming on Intel Xeon Phi Coprocessors: An Early Performance Comparison. In: Proceedings of the Many-core Applications Research Community (MARC) Symposium, pp. 38–44. Aachen, Germany (2012)
86. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON). RFC 7159, RFC Editor (2014). URL http://tools.ietf.org/pdf/rfc7159.pdf
87. Curreri, J., Stitt, G., George, A.D.: High-level synthesis of in-circuit assertions for verification, debugging, and timing analysis. International Journal of Reconfigurable Computing **2011**, 1–17 (2011). DOI http://dx.doi.org/10.1155/2011/406857

88. Czajkowski, T.S., Aydonat, U., Denisenko, D., Freeman, J., Kinsner, M., Neto, D., Wong, J., Yiannacouras, P., Singh, D.P.: From OpenCL to high-performance hardware on FPGAs. In: Proceedings of the 22nd International Conference on Field Programmable Logic and Applications (FPL), pp. 531–534. Oslo, Norway (2012)
89. Datta, K., Murphy, M., Volkov, V., Williams, S., Carter, J., Oliker, L., Patterson, D., Shalf, J., Yelick, K.: Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2008)., p. 4. IEEE (2008)
90. Davidson, A.A., Owens, J.D.: Toward techniques for auto-tuning GPU algorithms. In: Proceedings of the 10th International Conference on Applied Parallel and Scientific Computing (PARA), Revised Selected Papers, Part II, pp. 110–119. Reykjavík (2010)
91. Day, J.: Patterns in Network Architecture: A Return to Fundamentals. Prentice Hall International (2008)
92. Day, J., Matta, I., Mattar, K.: Networking is IPC: A Guiding Principle to a Better Internet. In: Proceedings of the 2008 ACM CoNEXT Conference, pp. 67:1–67:6 (2008). DOI 10.1145/1544012.1544079. URL http://doi.acm.org/10.1145/1544012.1544079
93. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI), pp. 137–150. San Francisco, California, USA (2004)
94. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons, England (2001)
95. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
96. Denholm, S., Inoue, H., Takenaka, T., Luk, W.: Application-specific customisation of market data feed arbitration. In: Proceedings of the International Conference on Field Programmable Technology (ICFPT), pp. 322–325. IEEE (2013)
97. Denholm, S., Inouey, H., Takenakay, T., Becker, T., Luk, W.: Low latency FPGA acceleration of market data feed arbitration. In: Proceedings of the International Conference on Application-Specific Systems, Architectures, and Processors (ASAP), pp. 36–40. IEEE (2014). DOI 10.1109/ASAP.2014.6868628
98. Dennett, D.C.: Consciousness Explained. Penguin Science (1993)
99. Dennis, J.B., Misunas, D.: A preliminary architecture for a basic data flow processor. In: Proceedings of the 2nd Annual Symposium on Computer Architecture, pp. 126–132 (1974)
100. Dieber, B., Simonjan, J., Esterle, L., Rinner, B., Nebehay, G., Pflugfelder, R., Fernandez, G.J.: Ella: Middleware for multi-camera surveillance in heterogeneous visual sensor networks. In: Proceedings of the International Conference on Distributed Smart Cameras (ICDSC) (2013). DOI 10.1109/ICDSC.2013.6778223
101. Dietterich, T.G.: Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science, pp. 1–15. Springer-Verlag (2000)
102. Diguet, J.P., Eustache, Y., Gogniat, G.: Closed-loop–based Self-adaptive Hardware/Software-Embedded Systems: Design Methodology and Smart Cam Case Study. ACM Transactions on Embedded Computing Systems **10**(3), 1–28 (2011)
103. Dinh, M.N., Abramson, D., J. Chao, D.K., Gontarek, A., Moench, B., DeRose, L.: Debugging scientific applications with statistical assertions. Procedia Computer Science **9**(0), 1940––1949 (2012)
104. Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems **1**(2), 223–259 (2006)
105. Dobson, S., Sterritt, R., Nixon, P., Hinchey, M.: Fulfilling the vision of autonomic computing. IEEE Computer **43**(1), 35 –41 (2010)
106. Dobzhansky, T., Hecht, M., Steere, W.: On some fundamental concepts of evolutionary biology. Evolutionary Biology **2**, 1–34 (1968)
107. Dorigo, M.: Optimization, learning and natural algorithms. Ph.D. thesis, Politecnico di Milano (1992)

108. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoretical computer science **344**(2), 243–278 (2005)
109. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **26**(1), 29–41 (1996)
110. Dutta, R., Rouskas, G., Baldine, I., Bragg, A., Stevenson, D.: The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 1899–1904 (2007). DOI 10.1109/ICC.2007.316
111. Duval, S., Wicklund, R.A.: A theory of objective self awareness. Academic Press (1972)
112. Ehrgott, M.: Other Methods for Pareto Optimality. In: Multicriteria Optimization, Lecture Notes in Economics and Mathematical Systems, vol. 491, pp. 77–102. Springer (2000)
113. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer (2003)
114. Eigenfeldt, A., Pasquier, P.: Considering vertical and horizontal context in corpus-based generative electronic dance music. In: Proceedings of the Fourth International Conference on Computational Creativity, p. 72 (2013)
115. Eigenfeldt, A., Pasquier, P.: Evolving structures for electronic dance music. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO), pp. 319–326. ACM (2013)
116. Elkhodary, A., Esfahani, N., Malek, S.: FUSION: a framework for engineering self-tuning self-adaptive software systems. In: Proceedings of the eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 7–16. ACM (2010). DOI 10.1145/1882291.1882296. URL http://doi.acm.org/10.1145/1882291.1882296
117. Elliott, G.T., Tomlinson, B.: PersonalSoundtrack: context-aware playlists that adapt to user pace. In: Proceedings of CHI'06 Extended Abstracts on Human Factors in Computing Systems, pp. 736–741. ACM (2006)
118. Ellis, T., Makris, D., Black, J.: Learning a Multi-camera Topology. In: Proceedings of the Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 165–171. IEEE Computer Society Press (2003)
119. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**, 1517–1531 (2011)
120. Endo, T., Matsuoka, S.: Massive supercomputing coping with heterogeneity of modern accelerators. In: Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp. 1–10 (2008)
121. Erdem, U.M., Sclaroff, S.: Look there! Predicting Where to Look for Motion in an Active Camera Network. In: Proceedings of the IEEE Conference on Advanced Video and Signal-based Surveillance, pp. 105–110. Como, Italy (2005)
122. Esterle, L., Lewis, P.R., Bogdanski, M., Rinner, B., Yao, X.: A socio-economic approach to online vision graph generation and handover in distributed smart camera networks. In: Proceedings of the International Conference on Distributed Smart Cameras (ICDSC), pp. 1–6. IEEE (2011). DOI 10.1109/ICDSC.2011.6042902
123. Esterle, L., Lewis, P.R., Caine, H., Yao, X., Rinner, B.: CamSim: A distributed smart camera network simulator. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 19–20. IEEE Computer Society Press (2013). DOI 10.1109/SASOW.2013.11
124. Esterle, L., Lewis, P.R., Rinner, B., Yao, X.: Improved adaptivity and robustness in decentralised multi-camera networks. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 1–6. ACM (2012)
125. Esterle, L., Lewis, P.R., Yao, X., Rinner, B.: Socio-economic vision graph generation and handover in distributed smart camera networks. ACM Transactions on Sensor Networks **10**(2), 20:1–20:24 (2014). DOI 10.1145/2530001
126. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The Many Faces of Publish/Subscribe. ACM Computing Surveys **35**(2), 114–131 (2003)

127. Faniyi, F., Lewis, P.R., Bahsoon, R., Xao, X.: Architecting self-aware software systems. In: Proceedings of the IEEE/IFIP Conference on Software Architecture (WICSA), pp. 91–94. IEEE (2014)
128. Farrell, R., Davis, L.S.: Decentralized discovery of camera network topology. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 1–10. IEEE Computer Society Press (2008). DOI 10.1109/ICDSC.2008.4635696
129. Fels, S., Hinton, G.: Glove-talk: A neural network interface between a data-glove and a speech synthesizer. IEEE Transactiona on Neural Networks **4**(1), 2–8 (1993)
130. Feng, W.: Making a case for efficient supercomputing. ACM Queue **1**(7), 54–64 (2003)
131. Fenigstein, A., Scheier, M.F., Buss, A.H.: Public and private self-consciousness: Assessment and theory. Journal of Consulting and Clinical Psychology **43**(4), 522–527 (1975)
132. Fern, A., Givan, R.: Online ensemble learning: An empirical study. Machine Learning **53**(1–2), 71–109 (2003)
133. Fette, B.: Cognitive radio technology. Academic Press (2009)
134. Fiebrink, R., Trueman, D., Cook, P.R.: A meta-instrument for interactive, on-the-fly machine learning. In: Proceedings of the International Conference on New Interfaces for Musical Expression. Pittsburgh (2009)
135. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Transactions on Internet Technology **2**(2), 115–150 (2002). DOI 10.1145/514183.514185. URL http://doi.acm.org/10.1145/514183.514185
136. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, pp. 148–156 (1996)
137. Froming, W.J., Walker, G.R., Lopyan, K.J.: Public and private self-awareness: When personal attitudes conflict with societal expectations. Journal of Experimental Social Psychology **18**(5), 476 – 487 (1982). DOI 10.1016/0022-1031(82)90067-1
138. Fu, H., Sendhoff, B., Tang, K., Yao, X.: Finding robust solutions to dynamic optimization problems. In: Proceedings of the 16th European conference on Applications of Evolutionary Computation (EvoApplications), pp. 616–625 (2013)
139. Funie, A., Salmon, M., Luk, W.: A hybrid genetic-programming swarm-optimisation approach for examining the nature and stability of high frequency trading strategies. In: Proceedings of the 13th International Conference on Machine Learning and Applications (ICMLA), pp. 29–34. Detroit, USA (2014). DOI 10.1109/ICMLA.2014.11. URL http://dx.doi.org/10.1109/ICMLA.2014.11
140. Gallup, G.G.: Chimpanzees: self-recognition. Science (1970)
141. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Proceedings of the 7th Brazilian Symposium on Artificial Intelligence (SBIA) - Lecture Notes in Computer Science, vol. 3171, pp. 286–295. Springer, São Luiz do Maranhão, Brazil (2004)
142. Gao, J., Fan, W., Han, J.: On appropriate assumptions to mine data streams: Analysis and practice. In: Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM), pp. 143–152 (2007)
143. Garlan, D., Cheng, S.W., Huang, A.C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. IEEE Computer **37**(10), 46–54 (2004)
144. Gelenbe, E., Loukas, G.: A self-aware approach to denial of service defence. Computer Networks **51**(5), 1299–1314 (2007)
145. Goto, M.: Active music listening interfaces based on signal processing. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 1441–1444 (2007)
146. Gouin-Vallerand, C., Abdulrazak, B., Giroux, S., Mokhtari, M.: Toward autonomic pervasive computing. In: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services, iiWAS '08, pp. 673–676. ACM, New York, NY, USA (2008)
147. Goukens, C., Dewitte, S., Warlop, L.: Me, myself, and my choices: The influence of private self-awareness on preference-behavior consistency. Tech. rep., Katholieke Universiteit Leuven (2007)

148. Grabner, H., Bischof, H.: On-line Boosting and Vision. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 260–267 (2006)
149. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line Boosting for Robust Tracking. In: Proceedings of the European Conference on Computer Vision, Lecture Notes in Computer Science, vol. 5302, pp. 234–247 (2008)
150. Group, K.: The OpenCL specification, version: 1.1. http://www.khronos.org/registry/cl/specs/opencl-1.1.pdf. (Accessed March 8, 2016)
151. Gudger, E.W.: A historical note on the synchronous flashing of fireflies. Science **50**(1286), 188–190 (1919)
152. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Nielsen, H.F., Karmarkar, A., Lafon, Y.: SOAP Version 1.2. World Wide Web Consortium (2007)
153. Guo, C., Luk, W.: Accelerating Maximum Likelihood Estimation for Hawkes Point Processes. In: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp. 1–6. IEEE (2013)
154. Guo, C., Luk, W.: Accelerating parameter estimation for multivariate self-exciting point processes. In: Proceedings of the International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 181–184. ACM (2014). DOI 10.1145/2554688.2554765
155. Haikonen, P.O.: Reflections of consciousness: The mirror test. In: Proceedings of the AAAI Fall Symposium on Consciousness and Artificial Intelligence, pp. 67–71 (2007)
156. Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., Isbell, C.: A novel sequence representation for unsupervised analysis of human activities. Artificial Intelligence **173**(14), 1221–1244 (2009). DOI DOI: 10.1016/j.artint.2009.05.002
157. Hansen, N.: The CMA evolution strategy: A comparing review. In: J. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea (eds.) Towards a New Evolutionary Computation, *Studies in Fuzziness and Soft Computing*, vol. 192, pp. 75–102. Springer Berlin Heidelberg (2006)
158. Happe, M., Agne, A., Plessl, C.: Measuring and Predicting Temperature Distributions on FPGAs at Run-Time. In: Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 55–60. IEEE Computer Society (2011). DOI 10.1109/ReConFig.2011.59
159. Happe, M., Huang, Y., Keller, A.: Dynamic Protocol Stacks in Smart Camera Networks. In: Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig), pp. 1–6. IEEE (2014)
160. Happe, M., Traber, A., Keller, A.: Preemptive Hardware Multitasking in ReconOS. In: Proceedings of the International Symposium on Applied Reconfigurable Computing (ARC), Springer (2015)
161. Hart, J.W., Scassellati, B.: Robotic self-modeling. In: J. Pitt (ed.) The Computer After Me, pp. 207–218. Imperial College Press / World Scientific Book (2014)
162. Heath, D., Jarrow, R., Morton, A.: Bond pricing and the term structure of interest rates: A new methodology for contingent claims valuation. Econometrica **60**(1), 77–105 (1992)
163. Hernandez, H., Blum, C.: Distributed graph coloring in wireless ad hoc networks: A lightweight algorithm based on Japanese tree frogs' calling behaviour. In: Proceedings of the 4th Joint IFIP Wireless and Mobile Networking Conference (WMNC), pp. 1–7 (2011)
164. Herzen, B.V.: Signal Processing at 250 MHz Using High-Performance FPGAs. In: Proceedings of the ACM Fifth International Symposium on Field-programmable Gate Arrays, pp. 62–68 (1997)
165. Ho, T.K.: The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(8), 832–844 (1998)
166. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision Combination in Multiple Classifier Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(1), 66–75 (1994)
167. Ho, T.S.Y., Lee, S.B.: Term Structure Movements and Pricing Interest Rate Contingent Claims. Journal of Finance **41**(5), 1011–1029 (1986)
168. Hoare, C.A.R.: An axiomatic basis for computer programming. Communications of the ACM **12**(10), 576–580 (1969)

169. Hockman, J.A., Wanderley, M.M., Fujinaga, I.: Real-time phase vocoder manipulation by runner's pace. In: Proceedings of the International Conference on New Interfaces for Musical Expression (2009)
170. Hoffmann, H., Eastep, J., Santambrogio, M., Miller, J., Agarwal, A.: Application heartbeats for software performance and health. In: ACM SIGPLAN Notices, vol. 45, pp. 347–348. ACM (2010)
171. Hoffmann, H., Eastep, J., Santambrogio, M.D., Miller, J.E., Agarwal, A.: Application Heartbeats: A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments. In: Proceedings of the International Conference on Autonomic Computing (ICAC) (2010)
172. Hoffmann, H., Holt, J., Kurian, G., Lau, E., Maggio, M., Miller, J.E., Neuman, S.M., Sinangil, M., Sinangil, Y., Agarwal, A., Chandrakasan, A.P., Devadas, S.: Self-aware computing in the Angstrom processor. In: Proceedings of the 49th Annual Design Automation Conference, DAC '12, pp. 259–264. ACM, New York, NY, USA (2012)
173. Hoffmann, H., Maggio, M., Santambrogio, M.D., Leva, A., Agarwal, A.: SEEC: A general and extensible framework for self-aware computing. Tech. Rep. MIT-CSAIL-TR-2011-046, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (2011)
174. Holland, B., George, A.D., Lam, H., Smith, M.C.: An analytical model for multilevel performance prediction of Multi-FPGA systems. ACM Transactions on Reconfigurable Technology and Systems **4**(3), 27–28 (2011)
175. Holland, O., Goodman, R.B.: Robots with internal models: A route to machine consciousness? Journal of Consciousness Studies **10**(4), 77–109 (2003)
176. Holopainen, R.: Self-organised sound with autonomous instruments: Aesthetics and experiments. Ph.D. thesis, University of Oslo (2012)
177. Hölzl, M., Wirsing, M.: Towards a system model for ensembles. In: Formal Modeling: Actors, Open Systems, Biological Systems, pp. 241–261. Springer (2011)
178. Hölzl, M., Wirsing, M.: Issues in engineering self-aware and self-expressive ensembles. In: J. Pitt (ed.) The Computer After Me, pp. 37–54. Imperial College Press/World Scientific Book (2014)
179. Horn, J., Nafpliotis, N., Goldberg, D.E.: A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 82–87 (1994)
180. Horn, P.: Autonomic computing: IBM's perspective on the state of information technology. Armonk, NY, USA. International Business Machines Corporation. (2001)
181. Hosseini, M.J., Ahmadi, Z., Beigy, H.: Using a classifier pool in accuracy based tracking of recurring concepts in data stream classification. Evolving Systems **4**(1), 43–60 (2013)
182. Hsu, C.H., Feng, W.C.: Reducing overheating-induced failures via performance-aware CPU power management. In: Proceedings of the 6th International Conference on Linux Clusters: The HPC Revolution (2005)
183. Hu, F., Evans, J.J.: Power and environment aware control of Beowulf clusters. Cluster Computing **12**, 299–308 (2009)
184. Hu, W., Tan, T., Wang, L., Maybank, S.: A Survey on Visual Surveillance of Object Motion and Behaviors. IEEE Transactions on Systems, Man and Cybernetics, Part C **34**(3), 334–352 (2004)
185. Huang, T., Russell, S.: Object Identification in a Bayesian Context. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1276–1283 (1997)
186. Huebscher, M., McCann, J.: Simulation Model for Self-Adaptive Applications in Pervasive Computing. In: Proceedings of the 15th International Workshop on Database and Expert Systems Applications, pp. 694–698. IEEE Computer Society (2004)
187. Hume, D.: A Treatise of Human Nature. Gutenberg eBook (1739). URL http://www.gutenberg.org/ebooks/4705. (Accessed March 8, 2016)
188. Hunkeler, U., Truong, H.L., Stanford-Clark, A.: MQTT-S–A publish/subscribe protocol for Wireless Sensor Networks. In: Proceedings of the Third International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE), pp. 791–798. IEEE (2008)

189. Hunt, A., Wanderley, M.M., Paradis, M.: The importance of parameter mapping in electronic instrument design. In: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 1–6. National University of Singapore (2002)
190. IBM: An architectural blueprint for autonomic computing (2003). URL http://www-03.ibm.com/autonomic/pdfs/AC Blueprint White Paper V7.pdf. (Accessed March 8, 2016)
191. Iglesia, D.: MobMuPlat (iOS application). Iglesia Intermedia (2013)
192. Intel: Sophisticated library for vector parallelism. http://software.intel.com/en-us/articles/intel-array-building-blocks/. (Accessed March 8, 2016)
193. Investigating RINA as an Alternative to TCP/IP. URL http://irati.eu. (Accessed March 8, 2016)
194. Ishibuchi, H., Murata, T.: A multiobjective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **28**(3), 392–403 (1998)
195. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Iterative approach to indicator-based multiobjective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3967–3974 (2007)
196. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization. In: Proceedings of the 3rd International Workshop on Genetic and Evolving Systems (GEFS), pp. 47–52. IEEE (2008)
197. James, W.: The principles of psychology. Henry Holt & Co. (1890)
198. Janusevskis, J., Riche, R.L., Ginsbourger, D., Girdziusas, R.: Expected Improvements for the Asynchronous Parallel Global Optimization of Expensive Functions: Potentials and Challenges. In: Y. Hamadi, M. Schoenauer (eds.) Learning and Intelligent Optimization, pp. 413–418. Springer (2012)
199. Javed, O., Khan, S., Rasheed, Z., Shah, M.: Camera Handoff: Tracking in Multiple Uncalibrated Stationary Cameras. In: Proceedings of the Workshop on Human Motion, pp. 113–118. IEEE Computer Society Press (2000). DOI 10.1109/HUMO.2000.897380
200. Javed, O., Rasheed, Z., Shafique, K., Shah, M.: Tracking across Multiple Cameras Disjoint Views. In: Proceedings of IEEE International Conference on Computer Vision, p. 952–957 (2003)
201. Jia, J., Veeravalli, B., Ghose, D.: Adaptive load distribution strategies for divisible load processing on resource unaware multilevel tree networks. IEEE Transactions on Computers **56**(7), 999–1005 (2007)
202. Jin, Q., Becker, T., Luk, W., Thomas, D.: Optimising explicit finite difference option pricing for dynamic constant reconfiguration. In: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp. 165–172 (2012)
203. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation **6**(5), 481–494 (2002)
204. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**(4), 455–492 (1998)
205. Jones, P., Cho, Y., Lockwood, J.: Dynamically optimizing FPGA applications by monitoring temperature and workloads. In: Proceedings of the International Conference on VLSI Design (VLSID). IEEE (2007)
206. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-Learning-Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1409–1422 (2012)
207. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Journal of Fluids Engineering **82**(1), 35–45 (1960)
208. Kamil, S., Chan, C., Oliker, L., Shalf, J., Williams, S.: An auto-tuning framework for parallel multicore stencil computations. In: Proceedings of the IEEE International Symposium on Parallel & Distributed Processing (IPDPS), pp. 1–12 (2010)
209. Kang, J., Cohen, I., Medioni, G.: Continuous Tracking within and across Camera Streams. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 267–272 (2003)

210. Kant, I.: The critique of pure reason. Gutenberg eBook (1781). URL http://www.gutenberg.org/ebooks/4280. Digital edition 2003 (Accessed March 8, 2016)
211. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Marca, D.: Accelerometer-based gesture control for a design environment. Personal and Ubiquitous Computing **10**(5), 285–299 (2006)
212. Keller, A., Borkmann, D., Neuhaus, S., Happe, M.: Self-Awareness in Computer Networks. International Journal of Reconfigurable Computing pp. 1–10 (2014). DOI 10.1155/2014/692076
213. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer **36**(1), 41–50 (2003)
214. Kettnaker, V., Zabith, R.: Bayesian Multi-Camera Surveillance. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, pp. 117–123 (1999)
215. Khan, M.I., Rinner, B.: Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning. In: Proceedings of the International Conference on Communications Workshops (ICCW). IEEE (2014). DOI 10.1109/ICCW.2014.6881310
216. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, vol. 2632, pp. 376–390. Springer (2003)
217. Kim, H.S., Sherman, D.K.: Express yourself: Culture and the effect of self-expression on choice. Journal of Personality and Social Psychology **92**(1), 1–11 (2007). DOI 10.1037/0022-3514.92.1.1
218. Kim, J., Seo, S., Lee, J., Nah, J., Jo, G., Lee, J.: OpenCL as a unified programming model for heterogeneous CPU/GPU clusters. In: Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPOPP), pp. 299–300 (2012)
219. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(3), 226–239 (1998)
220. Klinglmayr, J., Bettstetter, C.: Self-organizing synchronization with inhibitory-coupled oscillators: Convergence and robustness. ACM Transactions on Autonomous and Adaptive Systems **7**(3), 30:1–30:22 (2012)
221. Klinglmayr, J., Kirst, C., Bettstetter, C., Timme, M.: Guaranteeing global synchronization in networks with stochastic interactions. New Journal of Physics **14**(7), 1–13 (2012)
222. Knutzen, H., Nymoen, K., Torresen, J.: PheroMusic [iOS application]. URL http://itunes.apple.com/app/pheromusic/id910100415
223. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research **8**, 2755–2790 (2007)
224. Koski, J., Silvennoinen, R.: Norm methods and partial weighting in multicriterion optimization of structures. International Journal for Numerical Methods in Engineering **24**(6), 1101–1121 (1987)
225. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: Future of Software Engineering (FoSE), pp. 259–268. IEEE (2007)
226. Krishnamoorthy, S., Baskaran, M., Bondhugula, U., Ramanujam, J., Rountev, A., Sadayappan, P.: Effective automatic parallelization of stencil computations. In: Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 235–244 (2007)
227. Kuhn, H.W., Yaw, B.: The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly pp. 83–97 (1955)
228. Kuncheva, L.I.: A theoretical study on six classifier fusion strategies. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(2), 281–286 (2002)
229. Kurek, M., Becker, T., Chau, T.C., Luk, W.: Automating Optimization of Reconfigurable Designs. In: Proceedings of the International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 210–213. IEEE (2014). DOI 10.1109/FCCM.2014.65
230. Kurek, M., Becker, T., Luk, W.: Parametric Optimization of Reconfigurable Designs Using Machine Learning. In: Proceedings of the International Conference on Reconfigurable Computing: Architectures, Tools and Applications (ARC), *Lecture Notes in Computer Science*, vol. 7806, pp. 134–145. Springer (2013)

231. Legrain, L., Cleeremans, A., Destrebecqz, A.: Distinguishing three levels in explicit self-awareness. Consciousness and Cognition **20**, 578–585 (2011)
232. Legrand, D.: Pre-reflective self-as-subject from experiential and empirical perspectives. Consciousness and Cognition **16**(3), 583–599 (2007)
233. Leidenfrost, R., Elmenreich, W.: Firefly clock synchronization in an 802.15. 4 wireless network. EURASIP Journal on Embedded Systems **2009**, 7:1–7:17 (2009)
234. Leland, W., Taqqu, M., Willinger, W., Wilson, D.: On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Transactions on Networking **2**(1), 1–15 (1994). DOI 10.1109/90.282603
235. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: Binary robust invariant scalable keypoints. In: Proceedings of the International Conference on Computer Vision, pp. 2548–2555. IEEE (2011). DOI 10.1109/iccv.2011.6126542
236. Lewis, P.R., Chandra, A., Faniyi, F., Glette, K., Chen, T., Bahsoon, R., Torresen, J., Yao, X.: Architectural aspects of self-aware and self-expressive computing systems: From psychology to engineering. IEEE Computer **48**(8), 62–70 (2015)
237. Lewis, P.R., Chandra, A., Parsons, S., Robinson, E., Glette, K., Bahsoon, R., Torresen, J., Yao, X.: A Survey of Self-Awareness and Its Application in Computing Systems. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 102–107. IEEE Computer Society, Ann Arbor, MI, USA (2011)
238. Lewis, P.R., Esterle, L., Chandra, A., Rinner, B., Torresen, J., Yao, X.: Static, Dynamic, and Adaptive Heterogeneity in Distributed Smart Camera Networks. ACM Transactions on Autonomous and Adaptive Systems **10**(2), 8:1–8:30 (2015). DOI 10.1145/2764460
239. Lewis, P.R., Esterle, L., Chandra, A., Rinner, B., Yao, X.: Learning to be Different: Heterogeneity and Efficiency in Distributed Smart Camera Networks. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 209–218. IEEE Computer Society Press (2013). DOI 10.1109/SASO.2013.20
240. Lewis, P.R., Marrow, P., Yao, X.: Resource Allocation in Decentralised Computational Systems: An Evolutionary Market Based Approach. Autonomous Agents and Multi-Agent Systems **21**(2), 143–171 (2010)
241. Lewis, P.R., Platzner, M., Yao, X.: An outlook for self-awareness in computing systems. Awareness Magazine (2012). DOI 10.2417/3201203.004093
242. Li, B., Li, J., Tang, K., Yao, X.: An improved Two Archive Algorithm for Many-Objective Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 2869–2876 (2014)
243. Li, G., Gopalakrishnan, G.: Scaleable SMT-based verification of GPU kernel functions. In: Proceedings of the Eighteenth International Symposium on the Foundations of Software Engineering (FSE-18) (2010)
244. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Transactions on Evolutionary Computation **13**(2), 284–302 (2009)
245. Li, Y., Bhanu, B.: Utility-based Camera Assignment in a Video Network: A Game Theoretic Framework. Sensors Journal **11**(3), 676–687 (2011)
246. Liang, C.J.M., Liu, J., Luo, L., Terzis, A., Zhao, F.: RACNet: A High-Fidelity Data Center Sensing Network. Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems pp. 15–28 (2009)
247. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing **5**(6), 657–675 (2009)
248. Liu, Y., Yao, X.: Ensemble learning via negative correlation. Neural Networks **12**(10), 1399–1404 (1999)
249. Lübbers, E., Platzner, M.: Cooperative multithreading in dynamically reconfigurable systems. In: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp. 1–4. IEEE (2009)
250. Lübbers, E., Platzner, M.: ReconOS: Multithreaded programming for reconfigurable computers. ACM Transactions on Embedded Computing Systems **9** (2009)

251. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 674–679 (1981)
252. Maggio, M., Hoffmann, H., Santambrogio, M.D., Agarwal, A., Leva, A.: A comparison of autonomic decision making techniques. Tech. Rep. MIT-CSAIL-TR-2011-019, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology (2011)
253. Makris, D., Ellis, T., Black, J.: Bridging the Gaps between Cameras. In: Proceedings of Conference on Computer Vision and Pattern Recognition, vol. 2 (2004)
254. Marler, R.T., Arora, J.S.: Function-transformation methods for multi-objective optimization. Engineering Optimization **37**(6), 551–570 (2005)
255. Marrow, P.: Nature-inspired computing technology and applications. BT Technology Journal **18**(4), 13–23 (2000)
256. Marsaglia, G., Bray, T.A.: A convenient method for generating normal variables. SIAM Review **6**(3), 260–264 (1964)
257. Masahiro, N., Takaesu, H., Demachi, H., Oono, M., Saito, H.: Development of an automatic music selection system based on runner's step frequency. In: Proceedings of the 2008 International Conference on Music Information Retrieval, pp. 193–8 (2008)
258. Massie, M.L., Chun, B.N., Culler, D.E.: The Ganglia distributed monitoring system: design, implementation, and experience. Parallel Computing **30**, 817–840 (2004)
259. Mathar, R., Mattfeldt, J.: Pulse-coupled decentral synchronization. SIAM Journal on Applied Mathematics **56**(4), 1094–1106 (1996)
260. Max [computer software]. URL http://cycling74.com. (Accessed March 8, 2016)
261. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics pp. 55–61 (2000)
262. Mehta, N.R., Medvidovic, N.: Composing architectural styles from architectural primitives. In: Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 347–350 (2003). URL http://dblp.uni-trier.de/db/conf/sigsoft/fse2003.html#MehtaM03
263. Menasce, D.A., Sousa, J.a.P., Malek, S., Gomaa, H.: QoS Architectural Patterns for Self-architecting Software Systems. In: Proceedings of the 7th International Conference on Autonomic Computing (ICAC), pp. 195–204. ACM (2010). DOI 10.1145/1809049.1809084
264. Metcalfe, J., Shimamura, A.P. (eds.): Metacognition: Knowing about knowing. MIT Press, Cambridge, MA, USA (1994)
265. Michalski, R.S.: A Theory and Methodology of Inductive Learning. In: Machine Learning, Symbolic Computation, pp. 83–134. Springer Berlin Heidelberg (1983)
266. Miettinen, K., Mäkelä, M.M.: Interactive bundle-based method for nondifferentiable multiobjective optimization: nimbus. Optimization Journal **34**(3), 231–246 (1995)
267. Minku, L.L.: Online ensemble learning in the presence of concept drift. Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, UK (2010)
268. Minku, L.L., Yao, X.: DDD: A new ensemble approach for dealing with concept drift. IEEE Transactions on Knowledge and Data Engineering **24**(4), 619–633 (2012)
269. Minku, L.L., Yao, X.: Software Effort Estimation as a Multi-objective Learning Problem. ACM Transactions on Software Engineering and Methodology **22**(4), 35:1–32 (2013)
270. Miranda, E.R., Wanderley, M.: New Digital Musical Instruments: Control and Interaction Beyond the Keyboard. A-R Editions, Inc., Middleton, WI (2006)
271. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics **50**(6), 1645–1662 (1990)
272. Mitchell, M.: Self-awareness and control in decentralized systems. In: Proceedings of the AAAI Spring Symposium on Metacognition in Computation (2005). Available at http://www.cs.pdx.edu/ mm/self-awareness.pdf
273. Modler, P.: Neural networks for mapping hand gestures to sound synthesis parameters, vol. 18, p. 14. IRCAM — Centre Pompidou (2000)

274. Moens, B., van Noorden, L., Leman, M.: D-Jogger: Syncing music with walking. In: Proceedings of the Sound and Music Computing Conference, pp. 451–456. Barcelona, Spain (2010)
275. Morin, A.: Levels of consciousness and self-awareness: A comparison and integration of various neurocognitive views. Consciousness and Cognition **15**(2), 358–71 (2006)
276. Morin, A., Everett, J.: Conscience de soi et langage interieur: Quelques speculations. [Self-awareness and inner speech: Some speculations]. Philosophiques **XVII**(2), 169–188 (1990)
277. Müller-Schloer, C., Schmeck, H., Ungerer, T.: Organic computing: a paradigm shift for complex systems. Springer (2011)
278. Nakashima, H., Aghajan, H., Augusto, J.C.: Handbook of ambient intelligence and smart environments. Springer (2009)
279. Narukawa, K., Tanigaki, Y., Ishibuchi, H.: Evolutionary many-objective optimization using preference on hyperplane. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation Companion, pp. 91–92. ACM (2014)
280. Natarajan, P., Atrey, P.K., Kankanhalli, M.: Multi-camera coordination and control in surveillance systems: A survey. ACM Transactions on Multimedia Computing, Communications and Applications **11**(4), 57:1–57:30 (2015). DOI 10.1145/2710128
281. Nebehay, G., Chibamu, W., Lewis, P.R., Chandra, A., Pflugfelder, R., Yao, X.: Can diversity amongst learners improve online object tracking? In: Z.H. Zhou, F. Roli, J. Kittler (eds.) Multiple Classifier Systems, *Lecture Notes in Computer Science*, vol. 7872, pp. 212–223. Springer (2013). DOI 10.1007/978-3-642-38067-9_19
282. Nebehay, G., Pflugfelder, R.: Consensus-based matching and tracking of keypoints for object tracking. In: Proceedings of the Winter Conference on Applications of Computer Vision (WACV). IEEE (2014)
283. Nebro, A.J., Luna, F., Alba, E., Beham, A., Dorronsoro, B.: AbYSS: adapting scatter search for multiobjective optimization. Tech. Rep. ITI-2006-2, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, Malaga (2006)
284. Neisser, U.: The Roots of Self-Knowledge: Perceiving Self, It, and Thou. Annals of the NY AoS. **818**, 19–33 (1997)
285. netem. URL http://www.linuxfoundation.org/collaborate/workgroups/networking/netem. (Accessed March 8, 2016)
286. Nguyen, A., Satish, N., Chhugani, J., Kim, C., Dubey, P.: 3.5-D blocking optimization for stencil computations on modern CPUs and GPUs. In: Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13 (2010)
287. Niezen, G., Hancke, G.P.: Evaluating and optimising accelerometer-based gesture recognition techniques for mobile devices. In: Proceedings of AFRICON, pp. 1–6. IEEE (2009)
288. Nishida, K.: Learning and detecting concept drift. Ph.D. thesis, Hokkaido University (2008). URL http://lis2.huie.hokudai.ac.jp/ knishida/paper/nishida2008-dissertation.pdf
289. Nishida, K., Yamauchi, K.: Detecting concept drift using statistical testing. In: Proceedings of the Tenth International Conference on Discovery Science (DS) - Lecture Notes in Artificial Intelligence, vol. 3316, pp. 264–269. Sendai, Japan (2007)
290. Niu, X., Chau, T.C.P., Jin, Q., Luk, W., Liu, Q.: Automating elimination of idle functions by run-time reconfiguration. In: Proceedings of the 21st IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 97–104 (2013)
291. Niu, X., Coutinho, J.G.F., Luk, W.: A scalable design approach for stencil computation on reconfigurable clusters. In: Proceedings of the 23rd International Conference on Field programmable Logic and Applications (FPL), pp. 1–4 (2013)
292. Niu, X., Jin, Q., Luk, W., Liu, Q., Pell, O.: Exploiting run-time reconfiguration in stencil computation. In: Proceedings of the 22nd International Conference on Field programmable Logic and Applications (FPL), pp. 173–180 (2012)
293. Niu, X., Tsoi, K.H., Luk, W.: Reconfiguring distributed applications in FPGA accelerated cluster with wireless networking. In: Proceedings of the 21st International Conference on Field Programmable Logic and Applications (FPL), pp. 545–550 (2011)

294. NVIDIA: Cuda zone. http://www.nvidia.com/object/cuda_home_new.html. (Accessed March 8, 2016)
295. Nymoen, K., Chandra, A., Glette, K., Torresen, J.: Decentralized harmonic synchronization in mobile music systems. In: Proceedings of the International Conference on Awareness Science & Technology (iCAST), pp. 1–6 (2014)
296. Nymoen, K., Chandra, A., Glette, K., Torresen, J., Voldsund, A., Jensenius, A.R.: PheroMusic: Navigating a Musical Space for Active Music Experiences. In: Proceedings of the International Computer Music Conference (ICMC) joint with the Sound and Music Computing Conference, pp. 1715–1718 (2014)
297. Nymoen, K., Song, S., Hafting, Y., Torresen, J.: Funky Sole Music: Gait recognition and adaptive mapping. In: Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), pp. 299–302 (2014)
298. Okuma, K., Taleghani, A., de Freitas, N., Little, J., Lowe, D.: A Boosted Particle Filter: Multitarget Detection and Tracking. In: Proceedings of 8th European Conference on Computer Vision, vol. 3021, pp. 28–39 (2004)
299. Olfati-Saber, R.: Distributed Kalman filtering for sensor networks. In: Proceedings of the Conference on Decision and Control, pp. 5492–5498 (2007). DOI 10.1109/CDC.2007.4434303
300. Olsson, R.A., Keen, A.W.: Remote procedure call. The JR Programming Language: Concurrent Programming in an Extended Java pp. 91–105 (2004)
301. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal **41**(4), 689–696 (2003)
302. Ontañón, S., Plaza, E.: Multiagent Inductive Learning: An Argumentation-based Approach. In: J. Fürnkranz, T. Joachims (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML), pp. 839–846. Omnipress, Haifa, Israel (2010)
303. Oxford: Oxford dictionaries: Adapt. http://www.oxforddictionaries.com/definition/english/adapt. (Accessed March 8, 2016)
304. Oza, N.C.: Online bagging and boosting. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 2340–2345 (2005)
305. Oza, N.C., Russell, S.: Experimental comparisons of online and batch versions of bagging and boosting. In: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 359–364 (2001)
306. Özuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast Keypoint Recognition Using Random Ferns. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(3), 448–461 (2010). DOI 10.1109/tpami.2009.23
307. Page, I., Luk, W.: Compiling occam into Field-Programmable Gate Arrays. In: Proceedings of the International Conference on Field programmable Logic and Applications (FPL) (1991)
308. Papakonstantinou, A., Liang, Y., Stratton, J.A., Gururaj, K., Chen, D., Hwu, W.W., Cong, J.: Multilevel Granularity Parallelism Synthesis on FPGAs. In: Proceedings of the 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 178–185. IEEE (2011)
309. Parashar, M., Hariri, S.: Autonomic computing: an overview. In: Proceedings of the International Conference on Unconventional Programming Paradigms, pp. 257–269. Springer-Verlag, Berlin (2005)
310. Parsons, S., Bahsoon, R., Lewis, P.R., Yao, X.: Towards a better understanding of self-awareness and self-expression within software systems. Tech. Rep. CSR-11-03, University of Birmingham, School of Computer Science, UK (2011)
311. Paul, C., Bass, L., Kazman, R.: Software Architecture in Practice. MA: Addison-Wesley (1998)
312. Paul, C., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley (2002)
313. Paulson, L.: DARPA creating self-aware computing. IEEE Computer **36**(3), 24 (2003). DOI 10.1109/MC.2003.1185213
314. Peleg, A., Wilkie, S., Weiser, U.C.: Intel MMX for Multimedia PCs. Communications of the ACM **40**(1), 24–38 (1997)

315. Perkowitz, M., Philipose, M., Fishkin, K., Patterson, D.J.: Mining models of human activities from the Web. In: Proceedings of the 13th International Conference on World Wide Web, pp. 573–582 (2004)
316. Perrone, M., Liu, L.K., Lu, L., Magerlein, K., Kim, C., Fedulova, I., Semenikhin, A.: Reducing Data Movement Costs: Scalable Seismic Imaging on Blue Gene. In: Proceedings of the 26th International Parallel & Distributed Processing Symposium (IPDPS), pp. 320–329 (2012)
317. Perrone, M.P., Cooper, L.N.: When networks disagree: Ensemble methods for hybrid neural networks. Neural Networks for Speech and Image Processing, Chapman-Hall, New York pp. 126–142 (1993)
318. Peskin, C.S.: Mathematical aspects of heart physiology. Courant Institute of Mathematical Sciences, New York University New York (1975)
319. Pflugfelder, R., Bischof, H.: People Tracking across Two Distant Self-calibrated Cameras. In: Proceedings of International Conference on Advanced Video and Signal-based Surveillance. IEEE Computer Society Press (2006)
320. Pflugfelder, R., Bischof, H.: Tracking across Non-overlapping Views Via Geometry. In: Proceedings of the International Conference on Pattern Recognition (2008)
321. Phelps, S., McBurney, P., Parsons, S.: Evolutionary mechanism design: A review. Autonomous Agents and Multi-Agent Systems **21**(2), 237–264 (2010)
322. Piciarelli, C., Esterle, L., Khan, A., Rinner, B., Foresti, G.: Dynamic Reconfiguration in Camera Networks: a short survey. IEEE Transactions on Circuits and Systems for Video Technology **PP**(99), 1–13 (2015). DOI 10.1109/TCSVT.2015.2426575. (early access)
323. Pilato, C., Loiacono, D., Tumeo, A., Ferrandi, F., Lanzi, P.L., Sciuto, D.: Speeding-up expensive evaluations in highlevel synthesis using solution modeling and fitness inheritance. In: Y. Tenne, C.K. Goh (eds.) Computational Intelligence in Expensive Optimization Problems, vol. 2, pp. 701–723. Springer (2010)
324. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine **6**(3), 21–45 (2006)
325. Polikar, R., Udpa, L., Udpa, S., Honavar, V.: Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **31**(4), 497–508 (2001)
326. Puckette, M.: Pure Data (PD) (software). URL http://puredata.info. (Accessed March 8, 2016)
327. Pylvänäinen, T.: Accelerometer based gesture recognition using continuous HMMs. In: Pattern Recognition and Image Analysis, pp. 639–646. Springer (2005)
328. Quaritsch, M., Kreuzthaler, M., Rinner, B., Bischof, H., Strobl, B.: Autonomous Multicamera Tracking on Embedded Smart Cameras. EURASIP Journal on Embedded Systems **2007**(1), 35–45 (2007)
329. Rajko, S., Qian, G., Ingalls, T., James, J.: Real-time gesture recognition with minimal training requirements and on-line learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8. IEEE (2007)
330. Ramamurthy, S., Bhatnagar, R.: Tracking recurrent concept drift in streaming data using ensemble classifiers. In: Proceedings of the Sixth International Conference on Machine Learning and Applications (ICMLA), pp. 404–409. Cincinnati, Ohio (2007)
331. Rammer, I., Szpuszta, M.: Advanced .NET Remoting. Springer (2005)
332. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems **23**(2), 15–18 (2008). DOI 10.1109/MIS.2008.19
333. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press (2006)
334. Reason [computer software]. URL https://www.propellerheads.se. (Accessed March 8, 2016)
335. ReconOS: A programming model and OS for reconfigurable hardware (2013). URL http://www.reconos.de/. (Accessed March 8, 2016)
336. Reisslein, M., Rinner, B., Roy-Chowdhury, A.: Smart camera networks. IEEE Computer **47**(5), 26–28 (2014)
337. Reyes, R., de Sande, F.: Automatic code generation for gpus in llc. The Journal of Supercomputing **58**(3), 349–356 (2011)

338. Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., Schmeck, H.: Towards a generic observer/controller architecture for organic computing. In: C. Hochberger, R. Liskowsky (eds.) INFORMATIK 2006 – Informatik für Menschen, *LNI*, vol. P-93, pp. 112–119. Bonner Köllen Verlag (2006)
339. Rietmann, M., Messmer, P., Nissen-Meyer, T., Peter, D., Basini, P., Komatitsch, D., Schenk, O., Tromp, J., Boschi, L., Giardini, D.: Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC) (2012)
340. Rinner, B., Esterle, L., Simonjan, J., Nebehay, G., Pflugfelder, R., Fernandez, G., Lewis, P.R.: Self-Aware and Self-Expressive Camera Networks. IEEE Computer **48**(7), 33–40 (2015)
341. Rinner, B., Winkler, T., Schriebl, W., Quaritsch, M., Wolf, W.: The evolution from single to pervasive smart cameras. In: Proceedings of the Second ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), pp. 1–10 (2008). DOI 10.1109/ICDSC.2008.4635674
342. Rinner, B., Wolf, W.: Introduction to Distributed Smart Cameras. Proceedings of the IEEE **96**(10), 1565–1575 (2008). DOI 10.1109/JPROC.2008.928742
343. RNA: Recursive Network Architecture. URL http://www.isi.edu/rna. (Accessed March 8, 2015)
344. Rochat, P.: Five levels of self-awareness as they unfold in early life. Consciousness and Cognition **12**, 717–731 (2003)
345. Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach, 3 edn. Pearson Education (2010)
346. Saaty, T.L.: The Analytical Hierarchical Process. McGraw-Hill (1980)
347. Sakellari, G.: The cognitive packet network: A survey. The Computer Journal **53** (2010)
348. SanMiguel, J.C., Shoop, K., Cavallaro, A., Micheloni, C., Foresti, G.L.: Self-Reconfigurable Smart Camera Networks. IEEE Computer **47**(5), 67–73 (2014)
349. Santambrogio, M., Hoffmann, H., Eastep, J., Agarwal, A.: Enabling technologies for self-aware adaptive systems. In: 2010 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 149–156. IEEE (2010)
350. Santner, J., Leistner, C., Saffari, A., Pock, T., Bischof, H.: PROST: Parallel robust online simple tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 723–730 (2010)
351. Schaumeier, J., Jeremy Pitt, J., Cabri, G.: A tripartite analytic framework for characterising awareness and self-awareness in autonomic systems research. In: Proceedings of the Sixth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 157–162 (2012)
352. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a Wii controller. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction, pp. 11–14. ACM (2008)
353. Schmeck, H.: Organic computing - a new vision for distributed embedded systems. In: Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), pp. 201–203 (2005)
354. Schmickl, T., Thenius, R., Moslinger, C., Timmis, J., Tyrrell, A., Read, M., Hilder, J., Halloy, J., Campo, A., Stefanini, C., Manfredi, L., Orofino, S.: CoCoRo–The Self-Aware Underwater Swarm. In: Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 120–126. IEEE Computer Society, Ann Arbor, MI, USA (2011)
355. Schnier, T., Yao, X.: Using negative correlation to evolve fault-tolerant circuits. In: Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware (ICES'2003) – Lecture Notes in Computer Science, vol. 2606, pp. 35–46. Springer-Verlag (2003)
356. Scholz, M., Klinkenberg, R.: Boosting classifiers for drifting concepts. Intelligent Data Analysis **11**(1), 3–28 (2007)
357. Sharan, K.: Java remote method invocation. In: Beginning Java 8 APIs, Extensions and Libraries, chap. 7, pp. 525–548. Springer (2014)

358. Shaw, M.J., Sikora, R.: A distributed problem-solving approach to inductive learning. Tech. Rep. CMU-RI-TR-90-262, School of Computer Science, Carnegie Mellon University (1990)
359. Shipp, C.A., Kuncheva, L.I.: Relationships between combination methods and measures of diversity in combining classifiers. Information Fusion **3**(2), 135–148 (2002)
360. Showerman, M., Enos, J., Pant, A., Kindratenko, V., Steffen, C., Pennington, R., mei Hwu, W.: QP: A heterogeneous multi-acceleator cluster. In: Proceedings of the International Conference on High-Performance Clustered Computing (2009)
361. Shukla, S.K., Yang, Y., Bhuyan, L.N., Brisk, P.: Shared memory heterogeneous computation on PCIe-supported platforms. In: Proceedings of the 23rd International Conference on Field programmable Logic and Applications (FPL), pp. 1–4 (2013)
362. Simonjan, J., Esterle, L., Rinner, B., Nebehay, G., Dominguez, G.F.: Demonstrating autonomous handover in heterogeneous multi-camera systems. In: Proceedings of the International Conference on Distributed Smart Cameras, pp. 43:1–43:3 (2014). DOI 10.1145/2659021.2669474
363. Sironi, F., Bartolini, D.B., Campanoni, S., Cancare, F., Hoffmann, H., Sciuto, D., Santambrogio, M.D.: Metronome: Operating System Level Performance Management via Self-adaptive Computing. In: Proceedings of the Design Automation Conference (DAC). ACM (2012)
364. Sironi, F., Cuoccio, A., Hoffmann, H., Maggio, M., Santambrogio, M.: Evolvable Systems on Reconfigurable Architecture via Self-aware Adaptive Applications. In: Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS) (2011). DOI 10.1109/AHS.2011.5963933
365. Sironi, F., Triverio, M., Hoffmann, H., Maggio, M., Santambrogio, M.: Self-aware Adaptation in FPGA-based Systems. In: Proceedings of the International Conference on Field Programmable Logic and Applications. IEEE (2010)
366. Smallwood, J., McSpadden, M., Schooler, J.: The lights are on but no one's home: meta-awareness and the decoupling of attention when the mind wanders. Psychonomic Bulletin and Review **14**(3), 527–533 (2007)
367. Song, S., Chandra, A., Torresen, J.: An ant learning algorithm for gesture recognition with one-instance training. In: Proceedings of the International Congress on Evolutionary Computation (CEC), pp. 2956–2963. IEEE (2013)
368. SRC Computers, L.: SRC-7 MAPstation. Tech. rep., SRC Computers (2009)
369. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation **2**(3), 221–248 (1994)
370. Stanley, K.O.: Learning concept drift with a committee of decision trees. Tech. Rep. UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin (2003)
371. Sterritt, R., Parashar, M., Tianfield, H., Unland, R.: A concise introduction to autonomic computing. Advanced Engineering Informatics **19**(3), 181–187 (2005)
372. Steuer, R.E., Choo, E.U.: An interactive weighted Tchebycheff procedure for multiple objective programming. Mathematical Programming **26**(3), 326–344 (1983)
373. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press (2000)
374. Strassen, V.: Gaussian elimination is not optimal. Numerische Mathematik pp. 13:354–356 (1969)
375. Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (KDD), pp. 377–382. New York (2001)
376. Strenski, D.: The Cray XD1 computer and its reconfigurable architecture. Tech. rep., Cray Inc. (2005)
377. Strey, A., Bange, M.: Performance Analysis of Intel's MMX and SSE: A Case Study. In: Proceedings of 7th International Euro-Par Conference on Parallel Processing (Euro-Par), pp. 142–147. Manchester, UK (2001)
378. Susanto, K.W., Todman, T., Coutinho, J.G.F., Luk, W.: Design Validation by Symbolic Simulation and Equivalence Checking: A Case Study in Memory Optimization for Image Manipulation, *LNCS*, vol. 5404, pp. 509–520. Springer (2009)

379. Sutter, H.: The free lunch is over: A fundamental turn toward concurrency in software. Dr. Dobb's Journal (2005)
380. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
381. Taj, M., Cavallaro, A.: Distributed and decentralized multi-camera tracking. IEEE Signal Processing Magazine **28**(3), 46–58 (2011)
382. Tawney, G.A.: Feeling and self-awareness. Psyc. Rev. **9**(6), 570 – 596 (1902)
383. Tesauro, G.: Reinforcement learning in autonomic computing: A manifesto and case studies. IEEE Internet Computing **11**(1), 22–30 (2007)
384. Thomas, D., Luk, W.: Non-uniform random number generation through piecewise linear approximations. In: Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pp. 1–6 (2006)
385. Thomas, D.B., Luk, W.: Credit Risk Modelling using Hardware Accelerated Monte-Carlo Simulation. In: Proceedings of the 16th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 229–238 (2008)
386. Todman, T., Boehm, P., Luk, W.: Verification of streaming hardware and software codesigns. In: Proceedings of the International Conference on Field Programmable Technology (ICFPT), pp. 147–150. IEEE (2012)
387. Todman, T., Stilkerich, S.C., Luk, W.: Using Statistical Assertions to Guide Self-Adaptive Systems. International Journal of Reconfigurable Computing **2014**, 1–8 (2014). DOI 10.1155/2014/724585
388. Tong, X., Ngai, E.: A ubiquitous publish/subscribe platform for wireless sensor networks with mobile mules. In: Proceedings of the IEEE Eighth International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 99–108 (2012)
389. Torresen, J., Hafting, Y., Nymoen, K.: A new Wi-Fi based platform for wireless sensor data collection. In: Proceedings of the International Conference on New Interfaces for Musical Expression, pp. 337–340 (2013)
390. Torresen, J., Plessl, C., Yao, X.: Special Issue on "Self-Aware and Self-Expressive Systems". IEEE Computer **48**(7), 45–51 (2015)
391. Touch, J., Pingali, V.: The RNA Metaprotocol. In: Proceedings of the International Conference on Computer Communications and Networks, pp. 1–6 (2008). DOI 10.1109/ICCCN.2008.ECP.46
392. Trucco, E., Plakas, K.: Video Tracking: A Concise Survey. Journal of Oceanic Engineering **31**(2), 520–529 (2006)
393. Tse, A.H.T., Chow, G.C.T., Jin, Q., Thomas, D.B., Luk, W.: Optimising performance of quadrature methods with reduced precision. In: Proceedings of the International Conference on Reconfigurable Computing: Architectures, Tools and Applications (ARC), *Lecture Notes in Computer Science*, vol. 7199, pp. 251–263. Springer (2012). DOI 10.1007/978-3-642-28365-9_21
394. Tse, A.H.T., Thomas, D.B., Tsoi, K.H., Luk, W.: Dynamic scheduling Monte-Carlo framework for multi-accelerator heterogeneous clusters. In: Proceedings of the International Conference on Field-Programmable Technology (FTP), pp. 233–240 (2010)
395. Tsoi, K.H., Luk, W.: Axel: A Heterogeneous Cluster with FPGAs and GPUs. In: Proceedings of the 18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays, pp. 115–124 (2010)
396. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion **9**(1), 56–68 (2008)
397. Vassev, E., Hinchey, M.: Knowledge representation and awareness in autonomic service-component ensembles – state of the art. In: 14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing, pp. 110–119 (2011)
398. Vasudevan, S.: What is assertion-based verification? SIGDA E-News **42**(12) (2012)
399. Vermorel, J., Mohri, M.: Multi-Armed Bandit Algorithms and Empirical Evaluation. In: Proceedings of the European Conference on Machine Learning, pp. 437–448. Springer (2005)
400. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. The Journal of Finance **16**(1), 8–37 (1961)

401. Vinoski, S.: CORBA: Integrating diverse applications within distributed heterogeneous environments. IEEE Communications Magazine **35**(2), 46–55 (1997)
402. Volker, L., Martin, D., El Khayaut, I., Werle, C., Zitterbart, M.: A Node Architecture for 1000 Future Networks. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 1–5 (2009). DOI 10.1109/ICCW.2009.5207996
403. Volker, L., Martin, D., Werle, C., Zitterbart, M., El-Khayat, I.: Selecting Concurrent Network Architectures at Runtime. In: Proceedings of the IEEE International Conference on Communications (ICC), pp. 1–5 (2009). DOI 10.1109/ICC.2009.5199445
404. Wang, J., Brady, D., Baclawski, K., Kokar, M., Lechowicz, L.: The use of ontologies for the self-awareness of the communication nodes. In: Proceedings of the Software Defined Radio Technical Conference (SDR), vol. 3 (2003)
405. Wang, S., Minku, L.L., Yao, X.: A learning framework for online class imbalance learning. In: Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL), pp. 36–45 (2013)
406. Wang, S., Minku, L.L., Yao, X.: Online class imbalance learning and its applications in fault detection. International Journal of Computational Intelligence and Applications **12**(1340001), (1–19) (2013)
407. Wang, S., Minku, L.L., Yao, X.: A multi-objective ensemble method for online class imbalance learning. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 3311–3318. IEEE (2014). DOI 10.1109/IJCNN.2014.6889545
408. Wang, S., Minku, L.L., Yao, X.: Resampling-based ensemble methods for online class imbalance learning. In: IEEE Transactions on Knowledge and Data Engineering, vol. 27, pp. 1356–1368. IEEE (2015). DOI 10.1109/TKDE.2014.2345380
409. Wang, Z., Tang, K., Yao, X.: A memetic algorithm for multi-level redundancy allocation. IEEE Transactions on Reliability **59**(4), 754–765 (2010)
410. Watson, R.: The Delta-t Transport Protocol: Features and Experience. In: Proceedings 14th Conference on Local Computer Networks, pp. 399–407 (1989). DOI 10.1109/LCN.1989.65288
411. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-inspired sensor network synchronicity with realistic radio effects. In: Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems, pp. 142–153 (2005)
412. Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H., Gäschka, K.M.: On patterns for decentralized control in self-adaptive systems. In: R. Lemos, H. Giese, H. Müller, M. Shaw (eds.) Software Engineering for Self-Adaptive Systems II, *Lecture Notes in Computer Science*, vol. 7475, pp. 76–107. Springer Berlin Heidelberg (2013)
413. Wikipedia: Adaptation (computer science). http://en.wikipedia.org/wiki/Adaptation. (Accessed March 8, 2016)
414. Winfield, A.: Robots with internal models: a route to self-aware and hence safer robots. In: J. Pitt (ed.) The Computer After Me. Imperial College Press / World Scientific Book (2014)
415. Wolf, W., Ozer, B., Lv, T.: Smart Cameras as Embedded Systems. IEEE Computer **35**(9), 48–53 (2002)
416. Wright, M.: Open Sound Control: an enabling technology for musical networking. Organised Sound **10**(3), 193–200 (2005)
417. Xiao, L., Zhu, Y., Ni, L., Xu, Z.: GridIS: An Incentive-Based Grid Scheduling. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, p. 65b (2005). DOI 10.1109/IPDPS.2005.237
418. Xilinx: SDAccel Development Environment. http://www.xilinx.com/products/design-tools/sdx/sdaccel.html. (Accessed March 8, 2016)
419. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: A review. Pervasive and Mobile Computing **8**(1) (2012)
420. Yiannacouras, P., Steffan, J.G., Rose, J.: VESPA: portable, scalable, and flexible FPGA-based vector processors. In: Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, pp. 61–70 (2008)

421. Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. ACM Computing Surveys **38**(4), 1–45 (2006)
422. Yin, F., D., M., Velastin, S.: Performance evaluation of object tracking algorithms. In: Proceedings of the International Workshop on Performance Evaluation of Tracking and Surveillance (2007)
423. Yin, L., Dong, M., Duan, Y., Deng, W., Zhao, K., Guo, J.: A high-performance training-free approach for hand gesture recognition with accelerometer. Multimedia Tools and Applications pp. 1–22 (2013)
424. Yu, X., Tang, K., Chen, T., Yao, X.: Empirical analysis of evolutionary algorithms with immigrants schemes for dynamic optimization. Memetic Computing **1**(1), 3–24 (2009)
425. Zadeh, L.: Optimality and non-scalar-valued performance criteria. IEEE Transactions on Automatic Control **8**(1), 59–60 (1963)
426. Zagal, J.C., Lipson, H.: Towards self-reflecting machines: Two-minds in one robot. In: Advances in Artificial Life. Darwin Meets von Neumann, *Lecture Notes in Computer Science*, vol. 5777, pp. 156–164. Springer (2011)
427. Zambonelli, F., Bicocchi, N., Cabri, G., Leonardi, L., Puviani, M.: On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In: Proceedings of the Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 108 –113 (2011)
428. Zarezadeh, A.A., Bobda, C.: Hardware Middleware for Person Tracking on Embedded Distributed Smart Cameras. Hindawi International Journal of Reconfigurable Computing (2012)
429. Zeppenfeld, J., Bouajila, A., Stechele, W., Bernauer, A., Bringmann, O., Rosenstiel, W., Herkersdorf, A.: Applying ASoC to Multi-core Applications for Workload Management. In: C. Müller-Schloer, H. Schmeck, T. Ungerer (eds.) Organic Computing – A Paradigm Shift for Complex Systems, *Autonomic Systems*, vol. 1, pp. 461–472. Springer Basel (2011)
430. Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N., Zhang, Q.: Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation **1**(1), 32–49 (2011)
431. Ziliani, F., Velastin, S., Porikli, F., Marcenaro, L., Kelliher, T., Cavallaro, A., Bruneaut, P.: Performance evaluation of event detection solutions: the CREDS experience. In: Proceedings of the International Conference on Advanced Video and Signal Based Surveillance, pp. 201–206 (2005)
432. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithm: Empirical Results. Evolutionary Computation **8**(2), 173–195 (2000)
433. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. In: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN), vol. 3242, pp. 832–842 (2004)
434. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich (2001)
435. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation **3**(4), 257–271 (1999)
436. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)