The CASCADAS Framework for Autonomic Communications

Luciano Baresi, Antonio Di Ferdinando, Antonio Manzalini, and Franco Zambonelli

Abstract An interesting approach to the design and development of the future Internet foresees a networked service *eco-system* capable of seamlessly offering services for human-to-human, human-to-machine and machine-to-machine interactions.
This chapter builds in this direction by describing a distributed component-ware framework for autonomic and situation-aware communication developed within the CASCADAS project. The core of this framework is the *Autonomic Communication Element* (ACE), an innovative software abstraction capable of providing dynamically adaptable services that can be built, composed, and let evolve according to autonomic principles. Services are capable of adapting their logic to the dynamically changing context they operate in without human intervention. As a result, whenever the need arises, ACEs can be federated autonomously and produce new services on a situation-aware basis. Systems and, in particular, eco-systems can thus be conceived as collections of ACEs.
The chapter introduces the concept of ACE and its different facets. It also presents the architecture of a prototype ACE-based platform and exemplifies the different concepts through a future *Pervasive Behavioral Advertisement* scenario.

Luciano Baresi
Dipartimento di Elettronica e Informazione, Politecnico di Milano, e-mail: Baresi@elet.polimi.it

Antonio Di Ferdinando
Electrical and Electronic Engineering Department, Imperial College London, e-mail: A.Di-Ferdinando@imperial.ac.uk

Antonio Manzalini
Telecom Italia Lab, e-mail: Antonio.Manzalini@telecomitalia.it

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, e-mail: Franco.Zambonelli@unimore.it

A.V. Vasilakos et al. (eds.), *Autonomic Communication*, DOI: 10.1007/978-0-387-09753-4_6,

1 Introduction

Today's Internet is rapidly evolving towards a collection of highly distributed, pervasive, communication-intensive services [26]. In the next future, such services will be expected to (i) autonomously detect and organize the knowledge necessary to understand the context in which they operate, and (ii) self-adapt and self-configure, to exploit at their best any situation, to meet the needs of diverse users, in diverse situations, without explicit human intervention. These features will enable a wide range of new activities that are simply not possible or impractical now. However, the achievement of such capabilities requires a deep re-thinking of the current way of developing and deploying distributed systems and applications.

In this direction, a promising approach consists in conceiving services as part of an "ecology" within which they can prosper and thrive at the service of users (i.e., an *eco-system*). This vision is attractive because would provide better services to end-users while, at the same time, meeting the emerging economic urge for service provision and system management deriving by the higher level of dynamism and variability of communication systems. In addition, systems built in respect of this view are characterized by a flatter architecture, where services at Network and Transport level of the classical ISO OSI architecture (levels 3 and 4 respectively) are provided at the same level of application-oriented services, that is levels 5 to 7, and cross-layer interactions are a natural part of the ecology itself.

In this context, this chapter presents the CASCADAS Autonomic Service Framework (or the *Framework*, for short), capable of enabling the conception outlined above through the development of *autonomic applications*, that is, applications capable of dynamically adapting their plans to cope well with situations where the environment changes in uncertain ways [22]. The Framework represents the major outcome of the CASCADAS EU-IST project [1], and advances the field of autonomic communications with at least the two contributions of (i) providing a novel component model that facilitates the development of services as autonomic applications, and (ii) providing an environment that supports the evolution of such services in an autonomous fashion.

The Framework is conceived around a set of complementary key enabling features, namely *situation-awareness*, *semantic self-organization* and *self-similarity*, around which we believe any future communication services infrastructure should be conceived. The identification of these features, described in the following, starts from key state-of-the-art concepts in the area of modern distributed computing and communication systems, and tries to advance and generalize them to properly account for the specific characteristics of autonomic and situation-aware communication services. The features above are blended in a sound component model, which provides a robust and dynamic modular conceptual framework for building autonomic semantic services. This *Autonomic Component-ware* provides a high-level reference model for the production of a new generation of programmable communication elements that can be reused at different levels of the stack.

The Framework is centered around the fundamental abstraction of *Autonomic Communication Elements* (ACEs), and supports the vision of advanced autonomic ser-

vices as developed and deployed in the form of ACEs and networks of them. To this extent, the Framework provides an environment where basic services can be created and executed. This environment allows services to evolve autonomously, through the enhancement of the enabling features mentioned above, and according the local needs arising on a timely basis. Particular emphasis is put on the support for autonomously organizing services to compose more sophisticated ones: through this feature, in fact, it is possible to build systems (i.e. collections of ACEs) that "specialize" in a topic. Then, self-similarity allows these to be easily integrated into other systems recursively. This enhances modular flexibility in the Framework, which can be easily extended with components offering specific composite services.
The Framework is offered as a Java-oriented open source software development toolkit for situated autonomic communications (hereafter, the *Toolkit*), under *GNU General Public License* (GPL), and can be freely downloaded [34] through the Sourceforge website [15]. The rest of the chapter is structured as follows. Section 2 frames the problem of a framework for autonomic communications to set the motivations of our research. Section 3 introduces the CASCADAS Framework, its basic self-organization, self-awareness and self-similarity features, and describes the ACE component model. Sections 4 and 5 provide details on such basic features, while Sections 6 and 7 describe the way they have been used to equip the Framework with more sophisticated features. These sections exploit a *Pervasive Behavioral Advertisement* as running case study to better illustrate and exemplify the features introduced. Section 8 sketches the dynamics of a fully working ACE-based prototype eco-system for the case study. Finally, Section 9 concludes the chapter and outlines some future research directions.

2 Autonomic Communication Frameworks

In recent years, the body of research work in the area of autonomic systems has been growing [22], and a number of frameworks have been proposed. Each of these has different characteristics, with, for instance, some narrowing the target scenario to clustered services [5] or grid environments [23, 25, 31], while others designing containers through which non-functional properties can be injected and managed in legacy systems [4, 17, 28]. All of these, however, originate from the view of autonomic behaviour as a mean for reducing the management costs of complex IT systems achieved, in turn, by enhancing autonomous reconfiguration, optimization and management of network elements in the system [16].
This view, perfectly in line with IBMs *autonomic computing* initiative [24], implies autonomic behavior to be supported only at the level of computing resources. As a consequence, all of the frameworks above are (more or less) effective in enhancing network resources with self-configuration and self-management capabilities. However, none of them targets, nor foresees, the provision of "smart" services to the users, for which they all result ineffective. Quitadamo and Zambonelli [30] affirm that the main reason for this is the lack of a broader support for autonomic behavior

in the stack. This, in turn, motivates a broader approach known under the name of *autonomic communication*.

Autonomic communication takes the key motivations of autonomic computing and extends them to conceive the creation and provision of a new generation of autonomic services that can be made available, and usable, by end users. Dobson [16] define autonomic communication as "a more general thrust aimed at a deep foundational rethinking of communication, networking and distributed computing paradigms to face the increasing complexities and dynamics of modern network scenarios". Despite the evident similarities, the autonomic computing approach is significantly different from the autonomic communication one. In fact, while the former is more oriented towards the direct management of network resources, the latter is concerned with the provision of services and the management of resources at both infrastructure and user levels.

Let us take an example, in order to clarify. Let us consider a crowded venue, such as a museum, an airport, or a rail/metro station, with a number of public screens used to advertise events as well as third party commercial contents. As of today, the advertisement policy typically consists in a cyclical, sequential, display of content. This latter is typically preloaded, anticipately to the start of advertisement process, in a static fashion so that any modification aimed at either modifying the content itself or the sequence with which it is displayed is subject to explicit human intervention. More importantly, the advertisement process is carried out independently of the context it operates in, and therefore the content displayed on a screen remains the same regardless of the differences among people passing by. Exceptions can be eventually found in marginal cases, where the advertisement is based on timed priorities (i.e., higher priority to food advertisements when the time for a meal approaches). We call this category of advertisement techniques *audience-insensitive*, as they are insensitive to actual audience. Indeed, this highly static scenario results ideal to show the advantages that the introduction of autonomic technology might bring. However, as described below, these advantages change significantly according to the approach used.

According to its *manifesto* [24], an autonomic computing approach "would aim at facilitating configuration and management of the IT infrastructure to the extent of limiting, or even avoiding, explicit human intervention while also reducing the costs for its maintenance". As a result, enhancement of autonomic technology through the autonomic computing approach would produce an IT platform with interacting software components capable of reconfiguring and managing themselves in an autonomous fashion at runtime. However, no improvement would be seen at user level, where the service offered would be exactly the same. In particular, no enhancement would be observed in terms of autonomy of the service, which would still need explicit human intervention to, for instance, re-sequence and update contents to be shown.

On the other hand, a smart service might consider the presence of infrastructures such as wireless networks, RFID receptors, etc. These might enable access to pervasive services, such as for instance downloadable maps or events program for the venue, which can be accessed by personal mobile devices carried by the audience in

the venue. This might constitute an incentive for device holders to provide publicly accessible information (e.g. age slot, interests, gender, etc), which might be used to the extent of adapting the displayed contents on the basis of the peculiar interests of people detected. This would enhance effectiveness of the advertisement process, as the exposition of a product would be optimized towards people known to be interested (and thus more receptive), while enhancing the validity of the advertisers' business investment, since the system would be in the position to provide guarantees on the impact of the content. We call the category of strategies allowing for such enhancement *audience-sensitive*, as they are sensitive to the audience they are shown to. As a side remark, it is worth noting that audience-sensitive strategies would also provide an effective way of avoiding the display of inappropriate content to particularly sensitive audience, such as children, without repercussions on the actual cost of the infrastructure.

An autonomic communication approach would aim at providing all the necessary support for the construction of such a smart audience-sensitive service, and therefore its application has the capabilities to impact the scenario in such a way to provide a service whose benefits can be immediately made available to all actors involved. Unfortunately, to the best of our knowledge no framework fulfilling the characteristics of the autonomic communication area can currently be found in literature.

The considerations above constitute the rationale and leading motivation behind our work. The CASCADAS ACE Framework aims at filling the lack highlighted above by providing an environment, and all necessary tools, to build complex and highly dynamic applications in respect of the paradigms of situated autonomic communication. Central to the Framework is the concept of *Autonomic Communication Element* (ACE), an innovate software engineering abstraction that allows the construction, and supports, the provision of highly dynamic services through the development of autonomic applications. In the remainder of this chapter, we will use the terms 'application' and 'service' interchangeably.

With respect to our context, the scenario introduced above can be realized as a collection of ACEs, each of which providing basic services, autonomously evolving in the system according to changes in context they operate in, and the needs that arise locally at each ACE, in a way similar to living organisms in natural eco-systems do. A working prototype eco-system for such scenario, that we named *Pervasive Behavioural Advertisement* (PBA), has been fully developed. We will use its structure to better explain the characteristics of the Framework throughout this document, and finally detail the prototype.

3 CASCADAS Framework

The CASCADAS autonomic service framework foresees services to interact with service users according to motivations driven by the context the former ones operate in. Services are developed and deployed as according to the component-ware paradigm offered by the specifications of ACEs. Therefore, ACEs are effectively

system components, and in the remainder of this document we will use the terms 'component' and 'ACE' interchangeably.
ACEs are capable of autonomously interacting to the extent of providing the desired functionality in a situation-aware way with no, or very limited, configuration efforts. Interactions may be aimed at forming more sophisticated services than the ones currently available and, in this case, may involve the spontaneous formation of new components in the system through aggregation of the original ones. Indeed, enabling of these highly dynamic interactions requires the components to own a set of features, which we retain of paramount importance to enable the vision of eco-systems of services.
Self-organization is one of the key design principles underpinning the autonomic management of large populations, and is therefore an essential property for the Framework. **Semantic self-organization** aims at exploiting the potential of "classical" self-organization and self-aggregation as key enablers for service composition and aggregation, and allows the identification of self-organization models of semantic (i.e. cognitive) nature. In the Framework, semantic self-organization allows ACEs to identify and document local rule-sets through which they can aggregate and form more sophisticated services. This is achieved according to *clustering*, *differentiation*, and *synchronization* techniques aimed at autonomously reinforcing organizational links, differentiating according to resource management strategies and select ideal aggregation partners respectively. By doing so, the desired collective behavior can be promoted in groups of ACEs.
Situation-awareness takes from *context-awareness*, that is, the capability of sensing the environment and reacting accordingly [32, 33], and advances it with techniques to organize the amount of distributed information in proper, strongly distributed "knowledge networks" to support situated and adaptive service provisioning. This enables services to autonomously adapt their logic to the context from which they are requested and in which they execute.
In the Framework, this capability is offered through a dedicated service, accessible through aggregation, which allows components to obtain situation-aware information. Internally, this information is made available by a *Knowledge Network* (KN) [8], which is in charge of correlating small bits of local context-aware information, typically received through interaction with pervasive technology receptors, and organizing it into global knowledge. Other ACEs can, then, become situation-aware by aggregating to the KN *façade* ACE and observe the evolution of the system by receiving and interpreting the information made available through the aggregation.
Internally, the KN ACE is itself an aggregate of a number of ACEs, each dealing with isolated local views, which semantically self-organize in order to provide a global snapshot on the present situation. Aggregation of the ACEs involved in the KN service is made in full respect of self-similar constraints, and thus the ACE resulting from aggregation exports standard communication interfaces that hide its aggregate nature to the eyes of other ACEs, also facilitating further aggregations to other ACEs in the eco-system.
Self-similarity refers to the capability of individual components to self-organize and self-aggregate to reproduce nearly identical structures over multiple scales

[2, 14]. Besides scalability, self-similarity is indeed a key enabler for the composition of complex communication-intensive services and for the structuring of the possibly enormous and multi-faceted items they will have to exploit in the knowledge networks. The Framework supports self-similarity by imposing that ACEs formed as aggregation of other ACEs provide the same set of interfaces as the original ones, so that service invocations to aggregate ACEs do not need to be treated in a specific way by the invoker. Properties above represent fundamental building blocks for the Framework. These, in turn, have been used, combined and exploited in the process of equipping the Framework with more sophisticated self-* features.

- **Self-healing** refers to special-purpose management techniques aimed at detecting ACEs entering faulty states and finding corresponding reasons to restore the correct operation of the element. This principle takes from semantic self-organization and situation-awareness to build, configure, and drive a control infrastructure over interoperating ACEs. Again, self-healing materializes in the Framework as (logically) dedicated ACEs responsible for monitoring one or more ACEs to detect faulty states and undertake recovery actions, when needed.
- **Self-Preservation** is the natural instinct of preserving oneself from harm and destruction. In our context, this is translated into the capability of self-recognize the emergence of anomalies, and self-repair them, in a scenario populated by heterogeneous entities and characterized by the lack of a centralized organization. The approach towards these issues focuses on the provision of a framework for basic common security mechanisms, which can be aggregated in order to provide more complex mechanisms. In doing so, the **security** elements effectively work in synergy with the supervision structure above, which provides an ideal ground for enhancing *confidentiality*, *integrity*, *authentication*, and *non-repudiation* features in a distributed fashion. Then, more sophisticated services can be provided through structured aggregation, semantic self-organization, of these simple components.

3.1 ACE Component Model

The ACE (Autonomic Communication Element) forms the core of the Framework, and its component model enables the design of applications in a self-similar manner. Central to the component model is the concept of *organ*. An organ is an ACE internal operative component that, as the name suggests, behaves as an organ in the human body. Specifically, organs are able to harmonize their own behavior to the execution of other interacting organs and, more in general, to the context in which the ACE is operating. Each organ is responsible for a specific type of tasks, and the interaction among them allows the constitution of an ACE as a standalone component. As example, Figure 1 depicts the assembly of an ACE from a set of organs with clearly defined tasks that leads to a well-structured modularized component model.

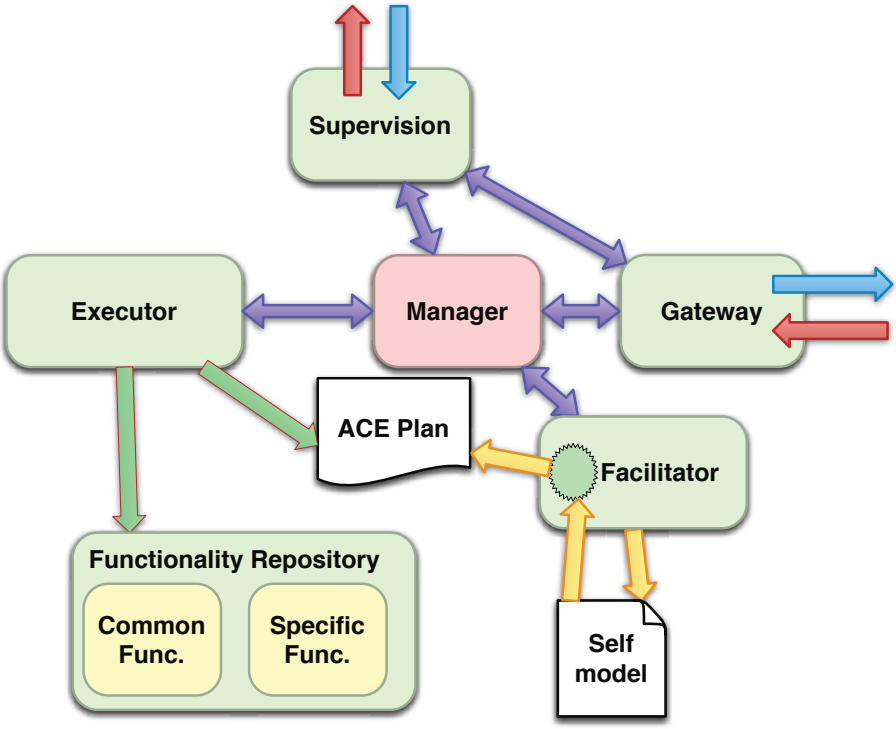

Fig. 1 ACE Component Model.

The *Gateway* organ is in charge of handling interactions with the external world. To this end, two different communication protocols are used: a connectionless protocol is used for initial service discovery through a publish-subscribe paradigm [18], supported by the *REconfigurable Dispatching System* [9] (REDS), under the name of *GN-GA* protocol [21]. On the other hand, a connection-oriented communication protocol is used for all other communications, where (one-to-one or one-to-many) communication channels are established through a contracting technique. The DIET [27] agent framework supports this communication.

The *Manager* organ is in charge of handling the internal communication among the organs and is responsible for the ACEs lifecycle management. With respect to this activity, any ACE must be in one of the four states of Figure 2: *inactive*, *running*, *prepared to move*, and *destroyed*. Different lifecycle actions are possible from these individual states.

The *Facilitator* is the organ that provides an ACE with capabilities for autonomously adapting its behavior to changes detected in the context. Adaptation is achieved through modification of the existing capabilities and/or addition of new ones. In more detail, ACEs are provided an original behaviour through a *self-model*, which defines the ACE's behavior with a number of *plans* each containing a set of states

along with rules to switch from one state to another. Plans can also include a set of *modification rules*, which allow to change the original behavior by modifying states and transitions in, or disabling, existing plans. In addition, modification rules might enable plans originally disabled or even create new plans. It is worth noting that modification rules are a major achievement in terms of situation-awareness, as in fact modifications are typically based on events occurring in the context the ACE is operative in. Semantic interpretation of plans, performed during initialization, leads to the creation of the original *ACE plan* that, in turn, allows creation of the ACE's initial behavior. During its operation, the Facilitator continuously evaluates the environmental conditions and the operations the ACE performs. Based on the outcome of the reasoning process, which are in turn influenced by the occurrence of contextual events, it subsequently performs, if needed, the specified modification actions on the basis of the peculiar events occurred. Changes made according to these, then, are submitted to the executor, described below, where they are made effective.

The *Executor* organ governs the evolution of the ACE according to the actions taken as per self-model. Its main role is to ensure that any reasoning and decision taken by the Facilitator is put in place in an effective and efficient way by ensuring that conditions are verified, actions are executed, and appropriate messages are exchanged.

Execution of plans may involve the use of specific capabilities provided by the

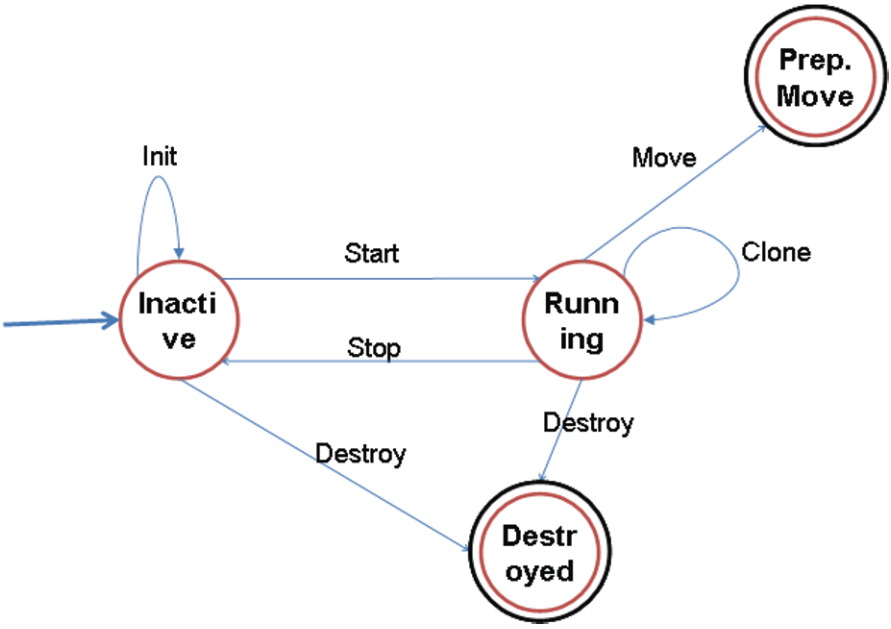

Fig. 2 State diagram of ACE's lifecycle management.

ACE. To this extent, the Executor may query the *Functionality Repository* organ to obtain them. The purpose of this repository is store the capabilities deployed.

ACEs provide services in form of functionality. These are stored in the repository, which is thus split into *Common* (i.e., guaranteed to be available in each ACE) and *Specific* (i.e., peculiar to a specific ACE) functionality. Common functionality is a major feature in terms of self-similarity. In fact, each and every ACE, regardless of its simple or aggregated nature, is capable of offering this set of capabilities albeit the set is strictly limited to the provision of basic operations. On the other hand, a service developer may create any arbitrary service and store it into the Specific Functionality Repository to equip the ACE the service is deployed in with more sophisticated and distinctive capabilities.
In addition to the organs described above, the original component model has been extended towards the provision of self-healing features through a dedicated organ aimed at interacting with a pervasive supervision framework. Activation of this organ, named *Supervision*, determines the "supervisable" nature of the ACE, and triggers monitoring activities at specific (configurable) points in the ACE aimed at verifying its own operative state. The structure of the pervasive supervision framework, as well as the description of ACEs self-healing features will be the subject of Section 6.

4 Semantic Self-Organization

The Framework supports semantic self-organization through algorithms for organized self-aggregation of ACEs in the form of autonomous *clustering*, *differentiation* and *synchronization*. The red wire among all the three self-aggregation techniques is that the choice of aggregation partners is subject to knowledge, and evaluation, of the context they are operating in, and that the self-organization will impact. This enables ACEs to conduct the aggregation according to a selection process of cognitive (i.e., semantic) nature, as deriving from the consideration of situation-aware information.
Clustering foresees an ACE initiating a "rewiring" procedure upon detection of a discrepancy in its list of required/available functionality (resulting from many different events such as the breaking of an existing collaboration link or a change in the local load due to a surge in demand). Depending on the circumstances, the initiator of the algorithm can choose one or more of its (contracted) first ACE neighbors as match-makers, and the constraint on the conservation of the total number of links can be relaxed or not. Simulation results [29] show that successful self-organization can take place, at a predictable rate, provided that well-identified conditions are met. The resulting aggregate ACEs will reflect the presence of durable complementariness between functions provided/encapsulated by individual ACEs (i.e., long-lasting GN/GA matches) and their ability to collectively identify and realize these functional clusters through local interactions. This can be generalized to any complex web of interdependencies, with individual ACEs potentially belonging to more than one functional cluster, and including "single type" aggregates designed for load-balancing rather than complementariness.

Differentiation allows ACEs to decide to "self-terminate" locally when facing an inappropriate workload. This automates the transfer of resources, between applications and according to fluctuations in the demand for a variety of co-hosted services, in a way that resources released can be re-assigned to other applications. The name *differentiation* derives from biological morphogenesis with which it shares some characteristics. Results from preliminary simulations [29] show that differentiation can help maximize the throughput of a distributed processing infrastructure while also making the system more responsive to heterogeneities in the workload when combined with the above "on demand" clustering. At the same time, results also emphasize that even the simplest set of rules tends to combine a large number of parameters to make interpretation difficult, therefore making selection of appropriate values a non-trivial problem. The algorithms integrated in the Framework enhance ACEs with capabilities to self-assess their own type according to the local observable workload, along with means for locally deciding when to change type.

Synchronization aims at finding or creating partnerships that adequately take into account the time activity pattern of individual constituting ACEs. This, in turn, involves (i) establishing a collaborative overlay that aggregates components featuring activity patterns that are *a priori* compatible starting from a random bootstrap configuration, and (ii) seeking ways of adjusting individual time-cycles so as to create opportunities for collaborations that would not exist if every individual activity pattern were "frozen" from the onset.

When the "rewiring" algorithm is employed, simulations show that the use of random time signatures allows the formation of a scale-free overlay when pruning from a complete graph and trying to secure a target number of active neighbors at any one time. In addition, they also show that distributed heuristics based on the "on-demand" clustering algorithm can be found so that near-optimal configurations obtained by pruning can be approximated.

In the economy of the Framework, and of the applications developed through it, self-organization is used as key enabler for composition of more sophisticated services, typically through aggregation of ACEs offering basic services. Furthermore, the semantic nature of the self-organization process brings a number of added values such as, for instance, the reinforcement of collaboration links in the aggregation process. As an example, consider the ACE that governs an advertisement screen in the PBA application. In order to conduct the advertisement process appropriately, it will have to aggregate to, among others, the ACE providing the multimedia contents, the ACEs providing self-healing features and, for instance, an ACE providing encryption services. Upon the rise of instabilities in the system, for instance due to a hardware failure causing a reduction of the available resources, it might enter an "emergency mode" whereby sensitive neighbors might be "clustered" to reinforce collaboration links critical to the service provision (e.g. the ACE providing multimedia content) while the self-healing infrastructure takes some action, also "synchronizing" to the "clustered" neighbors so as to optimize the aggregation's internal configuration. At the same time, it might "differentiate" with non-critical aggregations (e.g. the ACE providing encryption), to reduce the workload, thus allocating newly available resources to links retained critical. It is worth noting, finally, that

the employment of these algorithms is not automatic, but rather triggered by the situation-aware detection of well recognized instabilities.

5 Situation-Awareness

Situation-awareness refers to the ability of refining decisions according to the specific contextual situation. This capability requires models and tools for analyzing and organising pieces of contextual information into structured collections. To this end, the Framework offers a *Knowledge Network* (KN) [8] service, accessible, through aggregation, by ACEs as a system-wide service.

The KN is in charge of gathering and processing information to form a collection of *Knowledge Atoms* (KAs), which structure the latter in a data model conceived around the consideration that any bit of contextual knowledge is produced as a consequence of an event occurring in the context. Accordingly, a dedicated data model was created to represent any such fact, in a simple and expressive way, by means of a 4-tuple of the form *(Who, What, Where, When)*. This 4-tuple represents the basic unit of information in KAs and allows to account an entity (*Who*) involved in some activity (*What*) at a certain location (*Where*) at a certain moment (*When*).

The use of the data model enhances knowledge networking so as to make it possible to identify relations between atoms on the basis of content similarity (e.g. two atoms having the "who" field set to the same value corresponds to facts related to the same entity). Once relationships between atoms are identified and organized, it is possible to process them, to produce views, on the basis of equality of values in the same fields for different 4-tuples. For instance, data from a sensor network in an environment can be clustered according to the geographic closeness of sensors so that a concise perspective on an activity occurring in region larger than the one sensed by a single sensor can be generated.

Although it appears as a single ACE to the eyes of ACEs using it, the KN is internally composed by a collection of ACEs structurally organised in the architecture depicted in Figure 3. ACEs at bottom-most layer realize the concept of KA from heterogeneous data sources, from GPS devices to the Web. Data sources with very limited power, or too dynamic and ephemeral, are represented within so-called *Knowledge Repositories*. This might be the case, for instance, of data from RFID readers and sensor networks, which can be accessed via a repository rather than via individual KAs. It is worth noting that non-ACE applications may still be part of the Atom Repository through a simple interface.

On top of this layer, a number of components organize the KA so as to verify the concept of knowledge networks. The *Knowledge Organization* is in charge of organizing data in containers and exporting an interface for concept-based querying (i.e., by keyword) enabling higher level ACEs to access concept-based information. *Knowledge aggregators* allow establishing meaningful relationships among KAs, also storing data in a very expressive format and processing upon request. Results can thus be made available via a pattern-matching interface *à la Linda* [1]. The

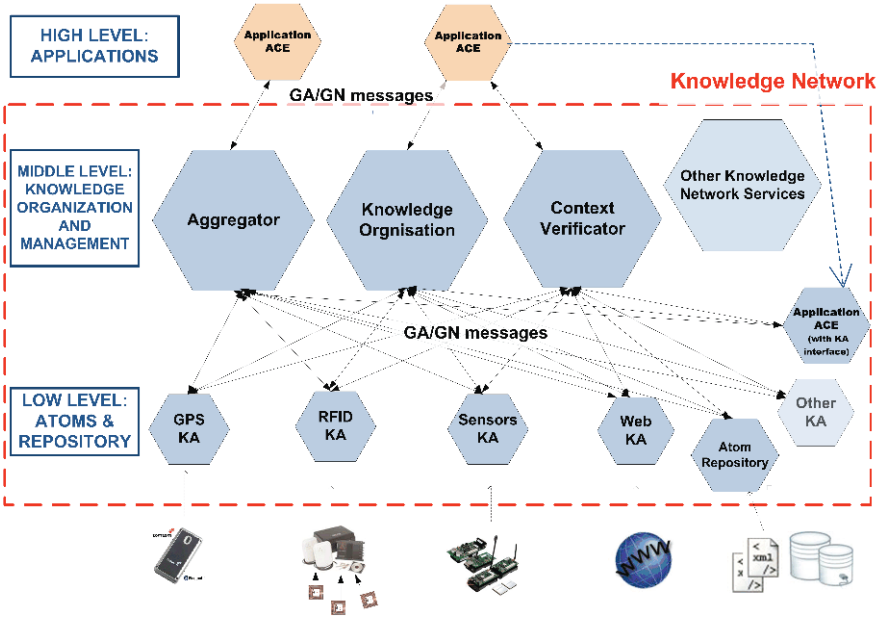

Fig. 3 Architecture of the KN.

idea behind the aggregator component is to provide specific information via pattern matching to form knowledge *views*. The *Context Verificator* verifies consistency of information in the KN according to (application- or user-) configurable parameters. This is done by accessing the KA(s) under verification and/or querying knowledge organization components to verify consistency in the knowledge. According to the outcome of such verification, then, it can notify the application about problems. An understanding of the advantages brought by the use of the KN can be obtained by looking at the PBA scenario. There, situation-awareness is the real enabler for self-adaptation in the advertisement process, as it allows the content allocation process to be driven according to contextual views of the composition of the current audience. In more detail, the ACE that takes the decision on the type of content to show, at a certain time, evaluates its appropriateness against the most recent view. Thus, by doing so it can make sure that the decision respects, in the best possible way, the nature of the interests that emerge as dominant from the view, and propose content that meets such interests. This process is realized by ACEs, representing potential content advertisers, aggregating to the KN in order to acquire situation-awareness features. By doing so, the former ones can come acquainted of the "situation by the screen", that is, the dominant interests among the audience currently in front of the screen, so as to ponder about relevance to own business, which would potentially lead to notification of interest.

6 Pervasive Supervision

Supervision refers to the ongoing observation of ACEs and the issuing of corrective measures upon detection of hazardous situations. Figure 4 shows the basic pattern for pervasive supervision, where all components are ACEs themselves. The *Sen-*

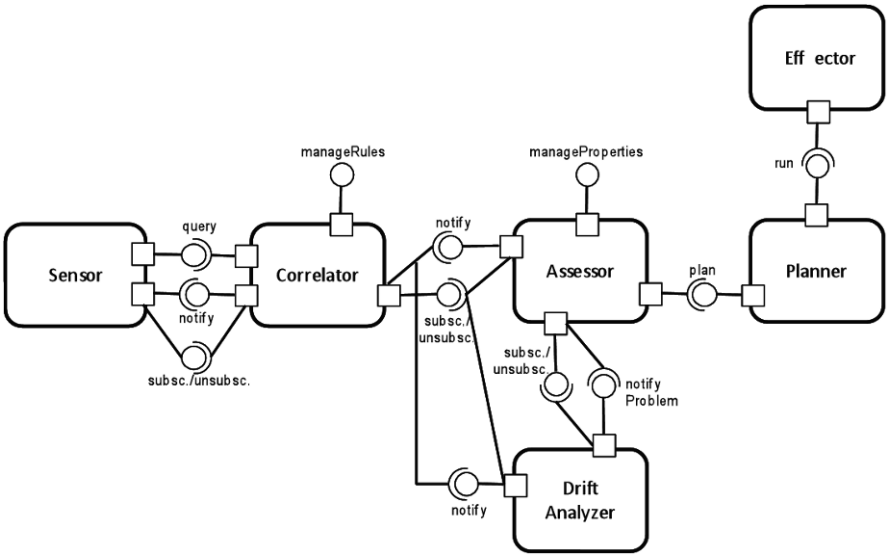

Fig. 4 ACEs for pervasive supervision

sor links the supervision system with the ACE (configuration) under supervision in order for this pattern to be monitored internally. Each ACE exports, through the dedicated *Supervision* organ described in section 3.1. a management interface through which internal state and session objects can be accessed and verified. Monitoring data gathered in this way is aggregated by the *Correlator*, to extract meaningful indicators of current health conditions of the system under supervision, and, in turn, analyzed by *Drift Analyzers* that try to anticipate future problematic situations in the system under supervision. Additionally, information from the environment may be used to supplement the analytical process. The outcome of this analysis constitutes the input for *Assessors*, which make predictions on the current (or future) system health, on the basis of raw data or output of correlators and drift analyzers.

The above components are concerned with detection of potential health threats for the configuration supervised. According to the outcome of this activity, a reaction is eventually decided and put in place as follows. The *Planner* tries to compute a solution plan for a detected problem, on the basis of the assessments generated by the Assessor and the actions described in the self-model of the ACE (or ACEs) under supervision. These are finally executed, through interception and modification of internal processes of the supervised ACE(s), by *Effectors*. The actual state of

the system under supervision, its potential future behaviors and countermeasures on problem situations is derived from an analysis of the composition of the self-models of the ACEs that constitute the supervised system itself through the use of a mathematical framework for model composition, abstraction, and local refinement [11].
Automation of supervision functions to limit human intervention, referred to as *Self-management*, can benefit of self-organization techniques, already employed in the Framework, to enable highly distributed supervision of large collection of ACEs.
Thus, a higher-granularity supervision system accounts for the self-adaptive nature of ACEs behavior, all of which implement the same supervision logic. Their cooperation enables for highly distributed supervision logic. Clustering and differentiation techniques are considered to foster information exchange and support of application-specific logical neighborhood to the extent of detecting and reacting to events related to ACE's behavior and their interactions through contracts.
The use of the pervasive supervision framework in the PBA scenario has the main purpose of enriching the prototype eco-system with capabilities of dynamic detection of ACEs entering faulty states. While, on one hand, this allows for a prompt heal, as per definition of self-healing, on the other hand its utilization also allows other operative ACEs to be notified of relevant anomalies as these occur. This, in turn, facilitates the design of modification rules to allow ACEs to cope with uncertain environments arising as a consequence of such anomalies. This is the case, for instance, of the ACE governing a display, which might ask allocation of an advertisement slot of time to other allocation ACEs upon notification, from the supervisor, that the one currently employed has entered a faulty state. It is worth noting that by doing so, the display ACE is also enhancing self-survivability of the display service, as the reaction guarantees the continuation of the service even though part of the aggregation is not in an operative state.

7 Security and Self-Preservation

The distinctive feature of a system built as a collection of ACEs is the absence of a centralized authority. As a consequence, *a-priori* trust relationships between ACEs belonging to different administrative domains cannot be assumed.
ACEs can show selfish, uncooperative or, in the worst-case, malicious behavior. Therefore, it becomes of paramount importance to address security issues in two distinct directions to cope with this wide range of attacks. Indeed, the behavior of ACEs needs to be monitored, and it is necessary to provide mechanisms to secure the communication infrastructure with confidentiality, integrity, and authentication. The Framework approaches these through security elements enhancing *hard-* and *soft-security* mechanisms, concerned with cryptographic mechanisms and behavioral attacks respectively. In the course of its operations, the security elements exploit the supervision infrastructure and employs semantic self-organization techniques to guarantee survivability of the system while ensuring self-preservation of (both simple and aggregate) ACEs and resources.

Hard security mechanisms aim at protecting the system from threats such as impersonation, eavesdropping, spoofing, and data modification. However, deploying cryptographic algorithms directly on ACEs would make them cumbersome, thus potentially prejudicing execution on pervasive devices. For this reason, the Framework exploits aggregation functionality and the GN/GA communication protocol to provide security in an adaptive and flexible way. Security features are thus provided through ACEs that can equip, through on demand aggregation, requesting ACEs with self-preservation and security features.

These ACEs provide a set of libraries containing basic cryptographic algorithms that can be combined on demand in order to compose more sophisticated security features. As an example consider, in the PBA scenario, the moment when the ACE governing the display communicates with the one governing the advertisement allocation service. There, the requesting ACE can be instructed to evaluate the availability of bandwidth as a parameter for deciding the type of encryption algorithms to use. The rational behind this check is that more complex algorithms will require altogether longer transmission times, which contrasts with the real-time requirements of inter-ACE communication in the PBA scenario. Then, once decided which encryption service to use, the requesting ACE might aggregate with the ACE providing the desired encryption algorithm in order to acquire the desired encryption capabilities. If at a later time the availability changes, the same ACE might want to employ a more complex encryption service, in which case the aggregation with the previous encryption ACE might be untied and another one might be undertaken with another ACE providing the service. This solution enables for flexible adaptation of ACEs to the security requirements of the application and type of service by implementing reusability of components for different purposes.

Giving ACEs responsibilities for the survivability of the system creates new challenges, whereby they are expected to react to eventual attacks, from malicious nodes targeting disruption of the system, by autonomously reconfiguring so as to exclude malicious ACEs. *Social control*, in the form of trust and reputation mechanisms, is employed in the Framework to this extent. Heuristics, or aggregation functions, can be injected so that ACEs' behavior can be captured and exclusion of malicious (selfish, or rational) entities can be enabled. Social control enables analysis of system evolution and interactions, and techniques borrowed from the game theory are used to minimize uncertainty in service provision and composition. This allows deriving conclusions on system performance in presence of selfish or uncooperative ACEs, and the cooperation level when a reputation management scheme drives interaction decisions [7]. System-wide self-preservation can also be enhanced, through a similar analysis, by exploiting semantic self-organization through the use of policies and strategies for the selection of specific partners on the basis of an evaluation of the level of involvement of an ACE. For instance, the use of such polices might enable an ACE to maximize its outcome in participating to system. This leads to an enhancement of the system survivability when the structure so formed is used to detect, and remove, malfunctioning components in order to keep the same service level[18].

The Framework targets protection against Distributed Denial-of-Service (DDoS),

perhaps the most difficult type of attack to deal with. Development of feasible protection mechanisms involving both detection and reaction are under development, where two possible response mechanisms derive from generic DoS detection schemes. The first allows ACEs to self-protect by filtering detected unwanted traffic, and it is applicable to DoS attacks within the ACE communication domain. The second, exploits service migration (self-configuration) to escape malicious flows.

8 Pervasive Behavioral Advertisement Scenario

The case study scenario mentioned throughout the chapter has been fully developed as a prototype platform named *Pervasive Behavioral Advertisement* (PBA). The platform is conceived as an eco-system of ACEs developed through the Toolkit, and we believe it might represent a first step in the direction of an application with a potentially immediate industrial spin-off.

The platform is populated by a number of originally disjoint ACEs, which self-organize in *Regions* upon start as depicted in Figure 5,. Regions are formed as a result of each ACE's behavior as specified in its own self-model, and are characterized by the diversity of the service(s) offered. In other words, each region offers different services, which are made available to other regions so as to enhance on demand cooperation to the extent of delivering the promised service in the promised terms and conditions. Cooperation is then exploited through further inter-region aggregation that, in turn, fosters a system-wide self-organization optimized towards the behavioral pervasive advertisement service to be offered.

Referring to Figure 5, the *Profiling Region* makes available services for obtaining user information. This region is composed by a collection of ACEs truly interacting with badges equipped with RFID tags via RFID antennas positioned by the screen in such a way to detect the presence (and gather information) of people appearing and disappearing from the screen range. This activity is carried out on a continuous basis, as the shape of arrow denotes in the figure. Bits of information so gathered are exchanged with the *Knowledge Network Region*, whose ACEs give it the shape of structurally organized knowledge in the way previously described. To the eyes of other ones, this region appears as a single ACE providing a situation-aware information provision service. In reality, as described in previous sections, the region is composed by a number of aggregated ACEs that seamlessly provide the service in full respect of the self-similar constraints highlighted previously.

This process outcomes availability of contextual information that other regions of the system obtain and use in order to acquire high degrees of situation-awareness. The *Display Region* provides the system with displaying capabilities for the advertisement to be shown. Allocation of the actual content to be shown takes place by invoking the slot allocation service provided by the *Allocation Region*. With this respect, in the presence of a large number of screens and parties interested in buying time slots on them, solutions for allocating time slots and generating added value for interested parties must be identified. From this point of view, auctions appear

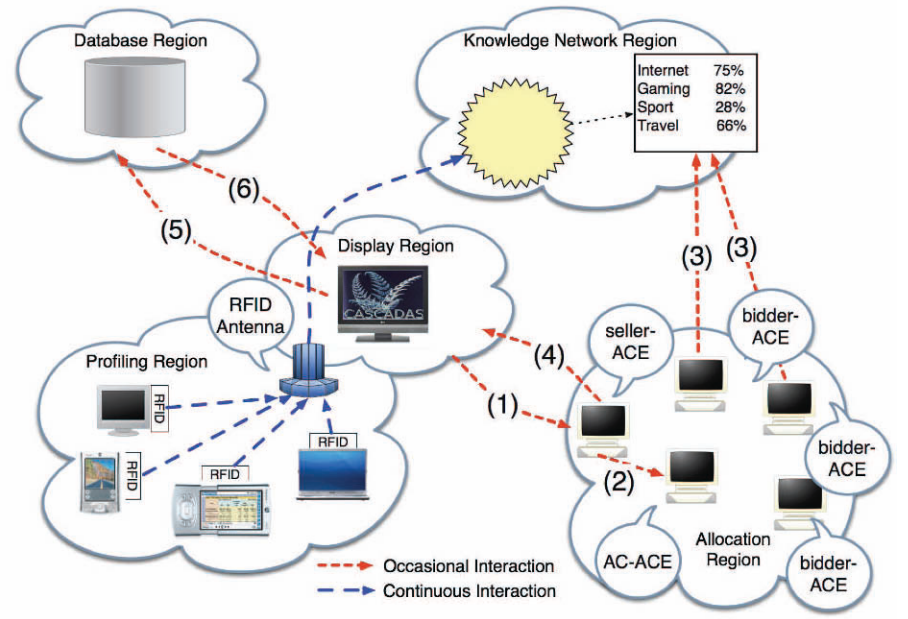

Fig. 5 The Pervasive Behavioral Advertisement scenario

an excellent solution as they prioritize allocation to advertisers who value them the most. Therefore, in our prototype the allocation of advertisement slots is decided by employing an auction-based paradigm whereby advertisers compete in a situation-aware fashion in order to acquire the rights of advertising on a specific screen at a specific time. The allocation service exported by the Allocation Region (interaction (1) in the figure) employs an iterative English auction, where ACE advertisers compete to the extent of acquiring the rights to expose own products in the slot of time under auction. Internally, the slot under auction is advertised by an *Auction Center* (interaction (2) towards the AC-ACE in the figure) through which is made available to advertisers (i.e. bidder-ACEs in figure) whose decisions on whether to submit a bid or not make intrinsic use the querying services offered by the Knowledge Network Region (interaction (3)). Thus, a bid is submitted if and only if the trends of interests reported in the (situation-aware) result of the query show sensitive relevance with the range of products the advertiser is competing to display. As in the case of the Knowledge Network Region, the Allocation Region has a single-ACE *façade* that, in turn, involves a number of aggregated, self-similar, ACEs to cooperate.

Upon auction termination, communication of the auction winner is returned back to the Display Region (interaction (4)), and is used by the ACEs there contained to select the right advertisement to display. The selection is offered as a service by the *Database Region*, which contains an advertisement database repository where advertisements are tagged based on owner and dominant relevance of interests. Thus,

selection of the right advertisement is done by simply invoking the service with auction winner and the interest the winning bid was submitted for, returned by the Auction Region, as input (interaction (5) for the query, and (6) for the response).
Seamlessly from the interaction model just described, which ensures that the contents advertised on the display evolve in the same way as the interests in the audience, the pervasive supervision framework enriches the platform with self-healing features. In the current prototype, however, self-healing capabilities are used for proof of concept, and therefore the only Auction Region is put under supervision. Specifically, relevant ACEs aggregate with the supervisor, which starts its monitoring activity towards all of them. If an ACE, say, the seller, enters a faulty state, a notification is sent to all other supervised ACEs, which can thus escape the faulty state by aggregating to other sellers, and a healing action aimed at reverting the ACE back to an operational state is started.
The prototype platform has been deployed on a distributed testbed, and executions show that ACEs developed are stable and capable of supporting the interaction model described above in extensive long-term sessions. In addition, preliminary performance evaluation tests have been conducted to the extent of evaluating effectiveness of the platform in terms of impact of the advertisement currently shown on the current crowd as based on a matching of relevant interests. Results show that our system enables high impact on the current crowd, as compared with the classical "round-robin". Analysis on such data allowed inferring that the effective investment cost for the advertiser, intended as the cost per-matching-person, decreases even though the price paid for advertising results many times increased (with a consequent increase in the screen owner's revenue).

9 Conclusions

This chapter describes the distributed component-ware framework for autonomic and situation-aware communication designed and developed in the context of the CASCADAS project. The use of the toolkit as a mean for creating computer networks enables the development of these latter ones as ecologies of services, or ecosystems, according to a vision that many foresee as a natural future evolution of the current Internet.
Eco-systems are characterized by a relatively flat architecture, which makes it easy to inject new services without, or with very limited, configuration efforts, and provide an environment where services can thrive to satisfy users' needs, while accounting for the actual operative context, as these both evolve. The framework embraces this vision by enhancing the creation of dynamically adaptable services through the innovative abstraction of Autonomic Communication Elements (ACEs), which are made available through an open source development Toolkit. This latter is fully usable and capable of exploiting highly innovative networking and application services through provision of features such as situation-awareness, which enriches the system with the ability to take into consideration the context evolution in local de-

cisions by providing system-wide knowledge in a highly expressive and flexible format, and semantic self-organization, which allows aggregating and organizing ACEs on the basis of dynamic contextual changes. These features are federated into a sound component model, which, among other things, ensures uniformity of the interfaces exported by composite ACEs resulting after aggregation of ACEs offering basic services. This results in a natural tendency to show self-similar behavior.
The above features have been used to equip the framework with other, composite, features that enrich the framework itself. In detail, security and self-preservation features are provided through a dedicated set of ACEs, where confidentiality, integrity, non-repudiation and authentication mechanisms are provided in form of on-demand services. Similarly, another dedicated set of ACEs brings self-healing features through a distributed approach towards detection of changes in the health of one or more ACEs, while also anticipating reactions based on projections of the way the state will evolve; A Pervasive Behavioral Advertisement application was selected fully developed as proof of concept scenario. The structure and interactions of the collection of ACEs that form it served as a way to better illustrate the features above, and how these support such a highly dynamic application. The scenario anticipates a potential future industrial scenario, and relies on the sole CASCADAS framework in order to build a robust and effective platform in a fully distributed fashion.

Acknowledgements The material here presented is based on a research funded by the CASCADAS (*Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services*) project (EU FP6-027807). Authors would like to thank Nermin Brgulja, Roberto Cascella, Peter Deussen, Elisabetta Di Nitto, Sandra Hasselhof, Ricardo Lent, Corrado Moiso and Fabrice Saffre for their help.

References

1. Ahuja S., Carriero N., Gelernter D., “Linda and Friends”. In *IEEE Computer*, Vol. 19, No. 8, pp. 26-34, 1986.
2. Albert R. and Barabasi A. “Statistical Mechanics of Complex Networks”. In Review of Modern Physics Vol 74(47), 2002.
3. Bicocchi N., Mamei M., Zambonelli F. “Self-organising Spatial Regions for Sensor Network Infrastructures”. In *Proc. of the* 21st *International Conference on Advanced Information Networking and Applications Workshops* (AINAW’07), Niagara Falls, Sept. 2007.
4. Bigus J. P., Schlosnagle D. A., Pilgrim J. R., Mills III W. N. and Diao Y. “ABLE: a toolkit for building multiagent autonomic systems”. In *Proc. of the* 1st *International Workshop on Enterprise Distributed Object Computing* (EDOC 2007), Annapolis, Oct. 2007.
5. Bouchenak S., De Palma N. and Hagimont D. “Autonomic Management of Clustered Applications”. In *Proc. of the IEEE International Conference on Cluster Computing*, Barcelona, Sept. 2006.
6. The Cascadas Consortium. “CASCADAS White Paper”. Available at `http://www.cascadas-project.org/Deliverable.html`.
7. Cascella R. G., “The "Value" of Reputation in Peer-to-Peer Networks”. In *Proc. of the Fifth IEEE Consumer Communications & Networking Conference* (CCNC 2008), Las Vegas, Jan. 2008.

8. Castelli G., Mamei M., Zambonelli F.. “Engineering Contextual Knowledge for Pervasive Autonomic Services”. To appear in *International Journal of Information and Software Technology*.
9. Cugola G. and Picco G.P. “REDS A Reconfigurable Dispatching System”. In *Proc. of the 6th International Workshop on Software Engineering and Middleware* (SEM2006), Portland, Nov. 2006.
10. Deussen P. H. “Model based reactive planning and prediction for autonomic systems”. In *Proc. of the INSERTech (Innovative SERvice Technologies) workshop*, Rome, Oct. 2007.
11. Deussen P. H., Valetto G., Din G., Kivimaki T., Heikkinen S., and Rocha A. “Continuous online validation for optimized service management”. In *EURESCOM Summit*, Heidelberg, Oct. 2002.
12. Di Ferdinando A., Rosi A., Lent R., Zambonelli F. and Gelenbe E. “A Platform for Pervasive Combinatorial Trading With Opportunistic Self-Aggregation”. In *Proc. of the 2nd IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications* (AOC’08), Newport Beach, Jun. 2008.
13. Di Ferdinando A., Lent R. and Gelenbe E.. “A Framework for Autonomic Networked Auctions”. In *Proc. of the 1st International Conference on Autonomic Computing and Communication Systems* (AUTONOMICS 2007), Rome, Oct. 2007.
14. S. Dill, R. Kumar, K. Mccurley, S. Rajagopalan, D. Sivakumar, A. Tomkins, “Self-Similarity in the Web”. In *ACM Transactions on Internet Technology* Vol. 2(3), pp. 205-223, 2003.
15. Sourceforge Website. `http://www.sourceforge.net`.
16. Dobson S., Denazis S., Fernandez A., Gaiti D., Gelenbe E., Massacci F., Nixon P., Saffre F., Schmidt N. and Zambonelli F. “A Survey of Autonomic Communications”. In *ACM Transactions on Autonomous Adaptive Systems*, Vol. 1(2), pp. 223-259, 2006.
17. Escoffier C., Hall R. S. and Lalanda P. “iPOJO: an Extensible Service-Oriented Component Framework”. In *Proc. of the 2007 IEEE International Conference on Service Computing* (SCC’07), Salt Lake City, Jul. 2007.
18. Eugster P., Felber P., Guerraoui R. and Kermarrec A.-M.. “The Many Faces of Publish/Subscribe”. In *ACM Computing Surveys*, 35(2):114-131, 2003.
19. Greer K., Baumgarten M., Mulvenna M., Curran K., Nugent C., “Autonomic Supervision of Stigmergic Self-Organisation for Distributed Information Retrieval”. In *Proc of the Workshop on Technologies for Situated and Autonomic Communications* (SAC), Budapest, Dec. 2007.
20. Greer K., Baumgarten M., Mulvenna M., Curran K., Nugent C. “Knowledge-Based Reasoning through Stigmergic Reasoning”. In *Proc. of the International Workshop on Self-Organising Systems* (IWSOS07). In: Hutchison D. and Katz R. H. (Eds.). *Lecture Notes in Computer Science*, LNCS 4725, 2007.
21. Höfig H., Benko B. K., Di NItto E., Mamei M., Mannella A., Wuest B. “On Concepts for Autonomics Communication Elements”. In *Proc. of the 1st IEEE International Workshop on Modeling Autonomic Communication Environments* (MACE 2006), Dublin, Oct. 2006.
22. Huebscher M. and McCann J. “A Survey of Autonomic Computing - degrees, modelas and applications”. In *ACM Computing Surveys*, December 2007.
23. Khargharia B., Hariri S., Parashar M., Ntaimo L. and Uk Kim B. “vGrid: A Framework for Building Autonomic Applications”. In *Proc. of the International Workshop on Challenges of Large Applications in Distributed Environments* (CLADE2003), Seattle, June 2003.
24. Kephart J. and Chess D. “The vision of autonomic computing”. In *IEEE Computer*, 36(1):4152, 2003.
25. Liu H., Parashar M. and Hariri S. “A Component Based Programming Framework for Autonomic Applications”. In *Proc. of the 1st IEEE International Conference on Autonomic Computing* (ICAC-04), New York, May 2004.
26. Manzalini A., Zambonelli F, “Towards Autonomic and Situation-Aware Communication Services: the CASCADAS Vision”. In *Proc. of the 2006 IEEE Workshop on Distributed Intelligent Systems* (DIS 2006), Prague, Jun. 2006.
27. Marrow P., Koubarakis M., Van Lengen R.H., Valverde-Albacete F., Bonsma E., Cid-Suerio J., Figueiras-Vidal A.R., Gallardo-Antolin A., Hoile C., Koutris T., Molina-Bulla H., Navia-Vasquez A., Raftopoulou P., Skarmeas N., Tryfonopoloulos C., Wang F. and Xiruhaki C.

"Agents in Decentralised Information Ecosystems: the DIET Approach". In *Proc. of the Artificial Intelligence and Simulation Behaviour Convention* (AISB 2001), York, Mar. 2001.
28. Melcher B. and Mitchell B. "Towards an Autonomic Framework: Self-Configuring Network Services and Developing Autonomic Applications". In *Intel Technology Journal*, Vol. 8(4), pp. 279-290, 2004.
29. Michiardi P., Marrow P., Tateson R. and Saffre F. "Aggregation Dynamics in Service Overlay Networks". In *Proc. of the* 1st *IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, Boston, Jul. 2007.
30. Quitadamo, R. and Zambonelli, F. "Autonomic communication services: a new challenge for software agents". In *Journal of Autonomous Agents and Multi-Agent Systems*, 2007.
31. Sajjad A., Jameel H., Kalim U., Lee Y.-K. and Lee S. "A Component-Based Architecture for an Autonomic Middleware Enabling Mobile Access to Grid Infrastructure". In *Proc. of the Conference on Embedded And Ubiquitous Computing* (EUC 2005), Nagasali, Dec. 2005.
32. Shilit B. N., Adams N. and Recker J. "Context-aware computing applications". In *Proc. of the IEEE Workshop on Computing Systems and Applications* (WMCSA94), Santa Cruz, Dec. 1994.
33. Shilit B. N. and Theimer M.M. "Disseminating Active Map Information to Mobile Hosts". In *IEEE Network* Vol. 8(5), pp. 22-32, 1994.
34. The ACE Autonomic Toolkit. `http://sourceforge.net/projects/acetoolkit`, or `http://acetoolkit.sourceforge.net`.
35. Vassilakis C. C., Laoutaris N., and Stavrakakis I., "The impact of playout policy on the performance of P2P live streaming". In *Proc. of the* 5th *Annual Multimedia Computing and Networking* (MMCN '08), San Jose, Jan. 2008.