A Design Framework for Ultra-Large-Scale Autonomic Systems

Michele Amoretti

Distributed Systems Group, Università degli Studi di Parma,
Via Usberti, 181a, Parma, Italy
michele.amoretti@unipr.it
http://www.ce.unipr.it/amoretti

Abstract. The origins of ultra-large-scale (ULS) systems derive from social problems that are getting more and more complex, such as climatic monitoring, transportation, citizens protection and security. These factors imply a continuous increase of information systems that evolve towards ultra-dimension systems, requiring digital communication networks that allow for communication between people, between objects, and objects and people.
The aim of this paper is to present novel approaches for the engineering of highly adaptive ULS systems, with the focus on computer-supported evolution, adaptable structure, emergent behaviors as well as advanced monitoring and control techniques. We illustrate the Networked Autonomic Machine (NAM), a framework for the characterization of the elements of self-*, highly dynamic ULS systems. Moreover, we recall the Adaptive Evolutionary Framework (AEF), for the implementation of distributed evolutionary strategies. Finally, we describe an example scenario of large peer-to-peer network under targeted attacks, showing the benefits of the NAM-AEF design.

Keywords: ultra-large-scale, autonomic, peer-to-peer, evolutionary

1 Introduction

A well-known study published by the Software Engineering Institute (SEI) [14] defines an *ultra-large-scale (ULS)* system as a system of unprecedented scale in some of the following dimensions: lines of code, amount of data (stored, accessed, manipulated, and refined), number of connections and interdependencies, number of hardware elements, number of computational elements, number of system purposes and user perception of these purposes, number of routine processes, interactions, and emergent behaviors, number of (overlapping) policy domains and enforceable mechanisms, number of people involved in some way. Foreseen ULS systems are, for example, super-national power grid management systems, healthcare infrastructures, e-markets, global ambient intelligence systems.

Breakthrough research is necessary to face fundamental challenges in the design and evolution, orchestration and control, and monitoring and assessment

of ULS systems. Specific challenges in ULS system design and implementation include legal issues, enforcement mechanisms and processes, definition of common services (infrastructural services), rules and regulations, handling of change, integration, user-controlled evolution, computer-supported evolution, adaptable structure and emergent quality.

The aim of this paper is to present novel approaches for the engineering of highly adaptive ULS systems, with the focus on computer-supported evolution, adaptable structure, emergent behaviors as well as advanced monitoring and control techniques. The overall vision is that of a ULS networked world in which software entities will be able to self-design, self-configure, self-monitor, self-deploy, self-adapt and self-heal (self-* properties, from now on). In other words, such ULS autonomic systems will be able to perform (live) in a totally unsupervised manner. One key aspect that will be investigated is the ability of foreseen self-refactoring systems that deploy and undeploy functional modules according to context and reasoning, in order to maintain desired quality of service in a varying and challenging environment. The latter may include users, whose interaction will affect the self-adaptation process of ULS autonomic systems.

Three important objectives need to be pursued to fulfill this vision. The first one is the **lack of centralized goals and control**. Decentralization increases the robustness of the services at a microscale, and encourages new applications at a macroscale. The second objective is **meaningful adaptation**. The proposed ULS autonomic systems are able to adapt correctly to given stimuli, maintain key behaviors and avoid deleterious ones - using evolutionary computation. Designers have to take into account emergent behaviors, for which it may happen that local optimizations lead to global performance improvements, but also to global unexpected failures. The third objective is **cooperation in the face of competition**, for which free-riding and selfish behaviors must be detected and rendered harmless. These challenges cannot be independently dealt with. On the contrary, they need to be woven into a coherent solution to be evaluated with respect to well-established performance metrics for distributed systems.

In this context, the peer-to-peer (P2P) paradigm appears as a highly appealing solution for scalable and high-throughput resource sharing among decentralized computational entities. In a P2P system, all participating processes are equally important, because they all contribute to the functioning of the whole system. A P2P system is a complex system, being composed of several interconnected parts that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts.

Thus, we depict ULS systems as autonomic P2P systems, being able to detect, diagnose and repair failures and adapt their behavior to changes in the environment. By increasing the context-awareness of monitoring data exchanged by autonomic peers, it would be possible to efficiently sense network conditions and the level of provided services and perform corrective actions. Sharing context-based information can be realized through dissemination of specific data among different nodes or through cross-module and cross-layer messages inside the same node. For example, a QoS entity responsible for allocating network

resources may exchange context-aware information with other nodes in order to identify changes in the network conditions. Context-based distributed self-monitoring should be finalized to automatic refactoring of software entities that compose the ULS system.

The rest of the paper is organized as follows. Section 2 presents the state of the art of autonomic computing, with particular emphasis on evolutionary adaptivity. Section 3 introduces our modeling framework for ULS autonomic systems, also discussing the problem of the characterization of complexity and emergent behaviors in ULS systems. Section 4 recalls the Adaptive Evolutionary Framework we introduced in previous works ([3] [4]), and sets it in the context of ULS autonomic systems. Section 5 applies the proposed techniques to the scenario of a large peer-to-peer network that runs through targeted attacks that attempt to break the topology in separated clusters. Finally, section 6 concludes the paper with a discussion on achieved results, and proposals for future work.

2 Related Work

The previously mentioned SEI study [14] brings together experts in software and other fields to examine the consequences of rapidly increasing scale in software-reliant systems. The report details a broad, multi-disciplinary research agenda for developing the ultra-large-scale systems of the future. Here we focus on computer-supported evolution, adaptable structure and emergent quality, that can be placed under the umbrella of Autonomic computing (AC).

AC is based on the assumption that the increasing complexity of distributed systems is becoming a limiting factor for further development. The solution proposed by the AC research community is to provide systems with four key properties: self-configuration, self-healing, self-optimization, self-protection [10]. Efforts to design self-managing systems have yielded many impressive achievements, yet the original vision of AC remains largely unfulfilled. As suggested in [6], researchers should adopt a comprehensive systems engineering approach to create effective solutions for next-generation large-scale distributed systems, for example by merging networking, software engineering and artificial intelligence. In our research activity, we take into account related concepts like context awareness, policies, ontologies, evolutionary algorithms, etc. combined with the peer-to-peer paradigm.

Until now, few significant attempts have been done for defining a specification language for autonomic systems. IBM has suggested a reference model for autonomic control loops, which is sometimes called the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) loop [8]. This model is being widely used to communicate the architectural aspects of autonomic systems. A rather interesting approach is based on the chemical programming paradigm [5], that captures the intuition of a collection of cooperative components which freely evolve, according to some predefined constraints (reaction rules). More general Bigraphical Reactive Systems (BRSs) [11] are a visual model of computation in which both locality and connectivity are prominent. Recognizing the increas-

ingly topographical quality of global computing, they take up the challenge to base all distributed computation on graphical structure. Such a graph is reconfigurable, and its nodes (the ovals and circles) may represent a great variety of computational objects: a physical location, an administrative region, a data constructor, a π-calculus input guard, an ambient, a cryptographic key, a message, a replicator, and so on.

With respect to evolutionary adaptivity in autonomic peer-to-peer systems, only few approaches have been proposed. Hales introduced an algorithm called SLAC, which means selfish link and behavior adaptation to produce cooperation [7]. SLAC is based on the copy and rewire approach, whose basic algorithm assumes that peer nodes have the freedom to change the way they handle and dispatch requests to and from other nodes, and drop and make links to nodes they know about. Another interesting approach for peer restructuring has been proposed by Tyson et al. [15], that show how survival of the fittest has been implemented into the Juno middleware. On receipt of a superior component, Juno dynamically reconfigures the internal architecture of the peer, by replacing the existing component with the new one.

3 Modeling ULS autonomic systems

Previously cited modeling framework MAPE-K [8] makes the strong (and quite inflexible) assumption of an ungainly Autonomic Manager for each autonomic element. On the other side, using an extremely general modeling tool like Bigraphical Reactive Systems [11] is questionable for ULS autonomic systems. To overcome these issues, we are developing a formal tool called Networked Autonomic Machine (NAM), that we introduced in [12]. NAM allows to model any software entity able to

- communicate with other NAMs;
- execute a number of functional modules that may provide/consume services or context events;
- dynamically deploy, undeploy and migrate functional modules and services.

These features support local self-management and self-healing activities. Conversely, the achievement of global autonomicity, in a ULS system made of NAMs, depends on the policies adopted at the functional level. For example, by providing all NAMs with a peer-to-peer overlay management module, it may be possible to enable their cooperation in routing messages for discovering new services or functional modules to use or download and execute.

In the previously cited chemical programming paradigm [5], system self-management arises as a result of interactions between members, in the same way as intelligence emerges from cooperation in colonies of biological agents. NAM follows the same approach, but uses a simpler formalism. With respect to other modeling tools, NAM allows to specify the migration of functional modules and services among nodes. The NAM formalism can be used to semantically

characterize and compare the elements of a self-*, highly dynamic distributed system.

Formally, a NAM node is a tuple $NAM = \langle R, F \rangle$, where R is a set of physical *resources*, such as CPU cycles, storage, bandwidth, and F is a set of *functional modules*. Each functional module $f \in F$ plays one or more of the following roles: context provider (CP), context consumer (CC), service provider (SP), service consumer (SC). Formally, $f = \langle S_f, S_r, C_{in}, C_{out}, POL \rangle$ where S_f is the set of provided services, S_r is the set of services the module can consume, C_{in} is the set of consumed context events, C_{out} is the set of provided context events, and POL is the set of policies according to which the functional module can react to the sensed environment. More precisely, when receiving a set of context events in C_{in}, the functional module may react by publishing a set of context events in C_{out}, by executing a set of services in S_f, or by calling a set of services in S_r. By means of self-* policies, NAMs are able to dynamically reconfigure their structure, by adding new functional modules or services, or discarding those that are no more necessary.

We envision NAM-based ULS autonomic systems that are peer-to-peer (P2P) networks whose environment is the set of users, that directly or indirectly act on peers and on resources that each node can use (running environment) and share with other nodes. The local environment is perceived by the peer by means of its input interfaces and sensors, providing direct messages from users, but also contextual information. The peer and its neighbors are part of the P2P system which is immersed in the environment. The response of the peer to the inputs that come from the local environment is usually contingent on interactions with neighbors (which in turn may involve their neighbors, etc.). The other way round, a peer can receive requests from its neighbors, and its response in general may depend and affect its local environment. Thus, environmental inputs usually target a very limited number of peers. When a peer receives an input (from the environment or from other peers), its internal structure maps the input to an output. The mapping process could require the peer to cooperate with other peers, exchanging messages in order to discover and eventually consume resources. In any case, localized reactions follow localized inputs. Thus, NAM-based ULS autonomic systems are complex adaptive systems.

4 Implementation of the Adaptive Evolutionary Framework for ULS systems

We propose to implement self-* mechanisms for ULS systems using the Adaptive Evolutionary Framework (AEF) that we have defined in some recent papers ([3] [4]), a distributed strategy, based on soft computing, to reconfigure nodes, *i.e.* to dynamically change the value of their parameters, leaving unchanged their structure. In the proposed project, major effort will be devoted to investigate the use of the AEF not only to reconfigure, but also to refactor nodes, *i.e.* to change their structure. AEF-based node refactoring will have the objective of enabling efficient, predictable, dependable, safe, and secure ULS systems.

With respect to the state of the art (*e.g.* [7] [15]), foreseen ULS autonomic systems will be made of nodes (modeled as NAMs) that may change their structure at run time in order to acquire new capabilities (by dynamically loading software components that implement specific abilities), or to lose obsolete capabilities (by undeploying components). Each node has a basic behavior that implements the node life cycle, specified as a set of actions to perform, by a set of Goals to use or achieve, and by a set of autonomic rules supporting application dependent self-configuration. Each node may have a knowledge base, for which adaptation would be epigenetic, *i.e.* based on learning and knowledge transmission. Conversely, adaptive re-structuring of nodes may be based on a phylogenetic approach, i.e. memoryless transformations.

In [4] we sketched the main features of the AEF and proposed a resource sharing system based on AEF. For the sake of clarity, it is necessary to distinguish between structure and configuration of a peer. The structure is the set of functional modules and connections among functional modules that compose the peer. Changing the structure means removing or adding functional modules and/or connections. Given a structure, there may be different configurations, since usually functional modules are characterized by parameters whose values may change over time. According to the AEF, the internal structure and/or configuration of a peer may change according to an adaptive plan τ, responding to modifications of the environment (figure 1). In our previous works [3] [4], we focused on the case of fixed structures, each one consisting of the same number of elements, and applied AEF to dynamically reconfigure nodes. Here we investigate the more general case of evolving structures. In that case, the search space has p dimensions where p is the number of known functional modules. Each element of C is binary, representing the presence (or absence) of a functional module in the peer.

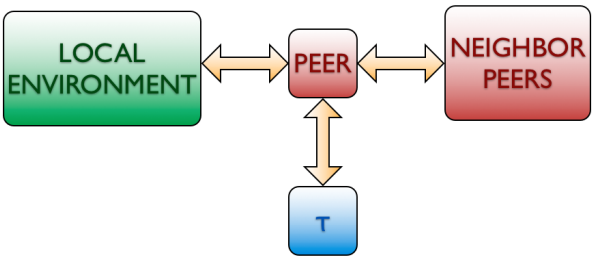

Fig. 1. Interactions between a peer, its local environment and neighbors in the Adaptive Evolutionary Framework.

Being all elements of C binary, there are 2^p possible solutions at the level of single peer. Considering the whole ULS system as a network of N peers sharing the same knowledge (set of functional modules) the number of possible solutions

is 2^{pN}. Fortunately, the environment usually affects a limited set of peers at a time. For example, a resource request is usually triggered by a user interacting with a peer, or by an application running on the same host (or in the same process space) of a peer which receives the request. The peer is directly affected by the environments input. The peer may own the requested resource, or may propagate the request to other peers. It is meaningful to say that these peers are indirectly affected by the environments input. When the system receives an input, not all its nodes must be considered for evolution, but only those which are directly or indirectly affected by the environments input. Thus, to design τ plans at the level of single peer is a reasonable tradeoff. Each node can evolve either autonomously or by merging its configuration with those of other nodes, *e.g.* of its k neighbors.

Genetic algorithms (GA), introduced by John Holland in 1975, have been the first phylogenetic evolutionary computing paradigm to be developed and applied [9]. We are currently adopting GA for solving a number of problems in ULS autonomic systems (one is illustrated in next section). In our approach, each peer compares its chromosome (encoding the peer's structure and configuration) with that of its neighbors. The best fitting neighbor is selected for crossing its chromosome with the one of the peer. The offspring is mutated with conditional probability. Finally, mutated offspring is compared with previous generation, in order to select the best structure for the peer.

5 Simulation experiments

We have applied the proposed NAM-AEF framework to the scenario of a large peer-to-peer network where nodes with higher node degree k run through targeted attacks that have the purpose of breaking the topology in separated clusters. All nodes are provided with a functional module that manages connections, selecting (by means of a periodically applied distributed genetic algorithm) one of the following strategies:

- Multiple Random Connections (MRC): the peer selects r existing nodes and connects to them (r is a random variable in $[1, R]$);
- Exponential Topology (ET): the peer selects m existing nodes and connects to them (m is fixed);
- Single Random Connection (SRC): the peer selects 1 existing node and connects to it.

In terms of robustness against random attacks, MRC is better than ET, which is better than SRC. On the contrary, in terms of robustness against targeted attacks, SRC is better than ET, which is better than MRC [1]. We assume that all nodes join the network using the MRC strategy, with $R = 10$. Then, the distributed adaptation process may lead to a change of strategy (either ET with $m = 3$ or SRC). For simplicity, the network is fixed, with 10^4 nodes. Targeted attacks are performed against nodes that have $k > 10$, *i.e.* more than

ten connections. Once attacked, a peer immediately reconnects to the network using the strategy that the adaption process has chosen as the best fitting.

The analysis has been performed by means of DEUS, a general-purpose tool for creating simulations of complex systems [2]. DEUS provides a Java API which allows to implement

- nodes (*i.e.* the parts which interact in a complex system, leading to emergent behaviors.. humans, pets, cells, robots, intelligent agents, etc.);
- events (e.g. node births/deaths, interactions among nodes, interactions with the environment, logs, etc);
- processes (stochastic or deterministic, they regulate the timeliness of events).

Once simulation classes have been implemented, the dynamics of their instances in a specific simulation can be defined by means of a XML document (using the DEUS XML Schema), or a visual editor that generates the XML schema. Simulations have been averaged over several execution runs with 20 different seeds, giving a very narrow 95% confidence interval.

Figure 2 illustrates how the node degree distribution changes over time, considering two different attack rates ($1/T_a$, with $T_a = 10^2, 10^4$), when adaptation is not applied. The "initial" distribution refers to the network before attacks start, while the "final" distribution refers to a steady-state condition, with ongoing attacks. Every attacked node reconnects using the MRC strategy, for which the observed node degree distribution does not change its shape over time. At the end of the simulation (virtual time $VT = 10^7$), the number of successful attacks, for the two considered attack rates, is 15500 and 450 respectively.

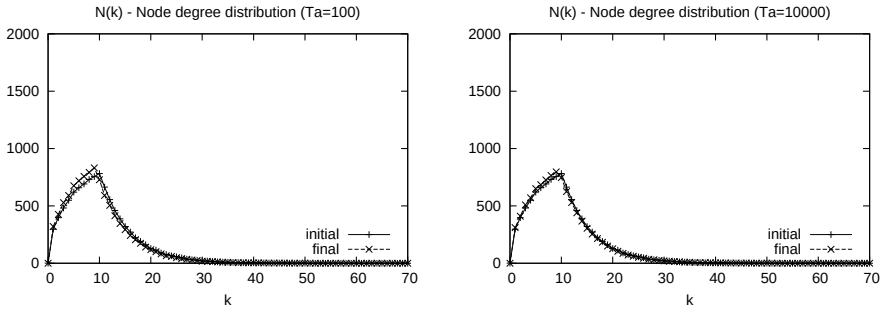

Fig. 2. Evolution of the node degree distribution for different attack rates (without adaptation).

Figure 3 refers to the same kind of analysis - change of the node degree distribution considering two different attack rates - when peers execute the adaptation protocol. When the attack rate is low, few peers have to reconnect and the node degree distribution does not change. On the contrary, when the attack rate is high, peers reconnect according to different strategies (ET or SRC) that lead

to a radically new node degree distribution, with all nodes having $k < 10$, a condition that preserves them from the targeted attacks. Moreover, in this case the number of successful attacks is 6900, *i.e.* 54% less than the case without adaptation.

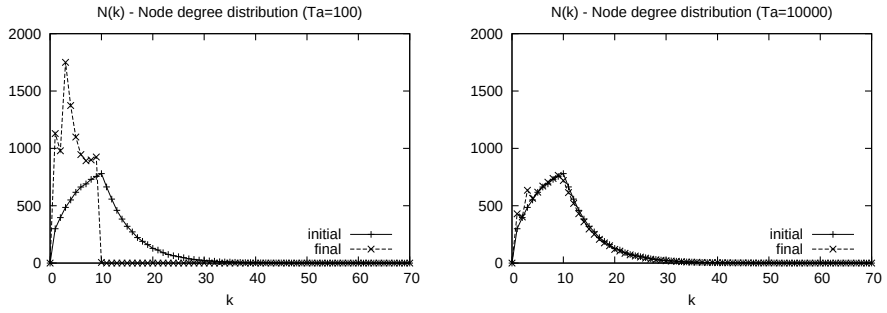

Fig. 3. Evolution of the node degree distribution for different attack rates (with adaptation).

Finally, figure 4 shows the evolution of the connection strategy distribution over time, in the case of high-frequency attacks. The adaptive strategy quickly leads to a stable configuration that preserves the network from targeted attacks.

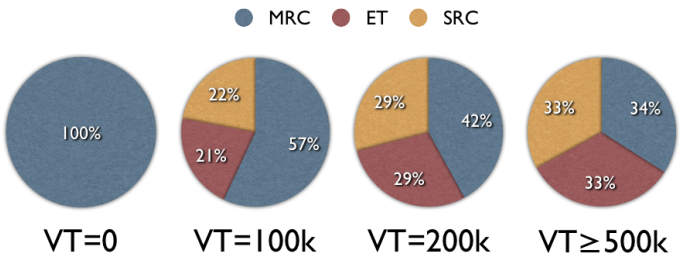

Fig. 4. Evolution of the connection strategy distribution over the virtual time of the simulation (with adaptation).

6 Conclusion

In this paper we have proposed novel approaches for the engineering of highly adaptive ULS systems, with the focus on computer-supported evolution, adaptable structure, emergent behaviors, as well as advanced monitoring and control techniques. We have illustrated the Networked Autonomic Machine (NAM),

a framework for the characterization of the elements of self-*, highly dynamic ULS systems. Moreover, we have recalled the Adaptive Evolutionary Framework (AEF), for the implementation of distributed evolutionary strategies. Then, we have described and analyzed, by means of simulations, an example scenario of large peer-to-peer network under targeted attacks. Obtained results show that the autonomic approach implemented by means of distributed evolutionary algorithms may be highly valuable.

As future work, we are going to improve the NAM formalism and to implement it in the *nam4j* middleware [13]. Moreover, we will present AEF-based solutions to other challenging problems that arise in the context of ULS systems.

References

1. R. Albert, H. Jeong, A.-L. Barabasi, *Error and attack tolerance of complex networks*, Nature 406, 378-482 (2000).
2. M. Amoretti, M. Agosti, F. Zanichelli, *DEUS: a Discrete Event Universal Simulator*, Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.
3. M. Amoretti, *A Framework for Evolutionary Peer-to-Peer Overlay Schemes*, European Workshops on the Applications of Evolutionary Computation (EvoWorkshops 2009), pp. 61-70, ISBN 978-3-642-01128-3, Tubingen, Germany, April 2009.
4. M. Amoretti, Fulfilling the Vision of Fully Autonomic Peer-to-Peer Systems, IEEE Int.l Conf. on High Performance Computing & Simulation, Caen, France (2010).
5. J.-P. Banatre, Y. Radenac, P. Fradet, *Chemical Specification of Autonomic Systems*, Proc. 13th Int.l Conference on Intelligent and Adaptive Systems and Software Engineering, Nice, France (2004).
6. S. Dobson, R. Sterritt, P. Nixon, M. Hinchey, *Fulfilling the Vision of Autonomic Computing*, IEEE Computer Magazine (2010).
7. D.Hales, *From Selfish Nodes to Cooperative networks - Emergent Link-based incentives in Peer-to- Peer Networks*, Proc. 4th IEEE Intl Conf. on Peer-to-Peer Computing, Zurich, Switzerland (2004).
8. M. C. Huebscher, J. A. McCann, *A survey of autonomic computing - degrees, models, and applications*, ACM Computing Surveys, Vol. 40, No. 3 (2008).
9. J. Holland, *Adaptation in Natural and Artificial Systems*, The MIT Press (1992).
10. IBM, *An architectural blueprint for autonomic computing.* Tech. rep. (2003).
11. R. Milner, *Bigraphical reactive systems: basic theory.* Tech. Report 503, University of Cambridge Computer Laboratory (2001).
12. M. Muro, M. Amoretti, F. Zanichelli, G. Conte, *Towards a Flexible Middleware for Context-aware Pervasive and Wearable Systems*, 3rd Int'l Symposium on Applied Sciences in Biomedical and Communication Technologies, Rome, Italy (2010).
13. Distributed Systems Group, *nam4j*, http://code.google.com/p/nam4j/
14. L. Northrop et al., *Ultra-Large-Scale Systems: The Software Challenge of the Future*, Carnegie Mellon Software Engineering Institute, Ultra-Large-Scale Systems Study Report (2006).
15. G. Tyson, P. Grace, A. Mauthe, S. Kaune, *The Survival of the Fittest: An Evolutionary Approach to Deploying Adaptive Functionality in Peer-to-Peer Systems*, Proc. of the 7th workshop on Reflective and adaptive middleware, Leuven, Belgium (2008).