Panel 1 Report: Autonomicity Versus Complexity*

Ioannis Stavrakakis and Antonis Panagakis

National & Kapodistrian University of Athens,
Dept. of Informatics & Telecommunications,
Panepistimiopolis, Ilissia, 15784 Athens, Greece
{ioannis, apan}@di.uoa.gr

Abstract. The first panel in WAC2005 focused on the relation between autonomicity and complexity. It is widely believed that autonomicity is a principle that can reduce complexity, but there is also concern that autonomicity itself is complexity-producing. Autonomicity promotes all "self-*" attributes of a system and naturally distributes responsibilities and costs, but it can also bring the system close to a state of "anarchy" (modern Greek interpretation of "autonomous") if not properly handled. It appears that the overall system complexity may increase, but it is distributed and shared (hence, it is potentially easier to manage), in a similar way in which Integrated Circuits encapsulate the increased complexity and hide it from the bigger system. In addition to reducing complexity in the above sense, autonomicity can also help design truly adaptable, self-tuning and "all-weather" near-optimal systems, something not possible under traditional system design that are difficult to cope with the combined fine-tuning of a very large number of parameters.

The panel was composed by the following researchers from Academia, Research Organizations and the Industry: Paul Spirakis of University of Patras - Research Academic Computer Technology Institute in Greece (coordinator), Radu Popescu-Zeletin and Mikhail Smirnov of Fraunhofer FOKUS in Germany, David Lewis of Trinity College Dublin in Ireland, Tom Pfeifer of Waterford IT in Ireland, Stefan Schmid of NEC Europe in Germany and Cesar Santivanez of BBN Technologies in USA.

According to [1], complexity may be understood in a number of different ways (e.g., computational complexity in computer science, emerging complexity in a physical system or process); the term is used to characterize a system that is hard to control (complicated nets, myriads of interactions) and might have a dynamic character (fast changes in huge structures, failures, or even updates that may "move" slower than the rate of changes).

Autonomicity is a word of Greek origin. It literally translates to "self-lawed" and in Modern Greek almost to "anarchy". For people in the networking field it means all the "self-*" properties, e.g. self-managed, self-configured, self-healing,

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-32993-0_29

* This report was supported in part by the IST FET Coordination Action ACCA (IST-6475).

I. Stavrakakis and M. Smirnov (Eds.): WAC 2005, LNCS 3854, pp. 286–292, 2006.

self-organised, self-improving; this also includes "selfishness" and thus antagonism. Autonomicity is perceived to presume a local "intelligence" of some degree and can be studied from a very low components level, up to the highest system level.

Comparing the terms, complexity is both a problem and a property. It is easy to "see" and hard to understand. Autonomicity, on the other side, is both a method and a property. It might provide an answer to complexity or it may create worse problems (chaos, anarchy ...).

Autonomic systems design is motivated by the fact that large systems disallow global control and therefore, central management becomes impossible. Examples of large systems that work nice are the market, the society and animal groups. These examples indicate that an autonomic system may start from simple principles and that evolution helps it.

Nevertheless, convincing arguments are needed to provide answers to questions such as: Do we attempt to hide some problems via autonomicity "magic"? For example, the following issues should be addressed in the near future:

- How far does "self-*" become implementable?
- How can we verify the correctness of a "self-*" implementation of a property?
- Can we convince that autonomic protocols are "stable" (think, e.g, about BGP)?
- What are the measures of quality of service in autonomic systems?
- Can we really design/derive self-improving code (and get rid of software designers)?

An additional issue is the exact difference between "autonomicity" and other research areas. For example, the foundations of distributed computing have many resemblances with autonomicity goals (e.g. Dijkstra's self-stabilizing code); thus, the question whether modern distributed computing is the same as autonomicity in communications, but just renamed, arises. In distributed computing local protocols and communication are utilized in order to achieve global goals (e.g., leader election protocols, byzantine agreement). Also, many impossibility results (a la FLP) indicate that not everything is possible.

Modern approaches to study complex systems include mathematics of local interactions from Physics, emerging nets/structures/behaviour theories and evolutionary processes. Evolutionary game theory is mathematically very precise and can be used to study evolution under antagonism. Under the framework of evolutionary game theory, "dynamics" and structure are connected in a beautiful way and individuals "learn" or even better "copy" behaviours from others. Motivating locals for "better" global behaviour is a new way to control complex systems; at the same time, it is also old, if one thinks of traffic lights, taxes, or advertisement. Modeling and controling the time-varying aspects of complex systems is a challenging task (dynamic control theory is obsolete).

The presentation in [2] focuses on the tradeoff of complexity vs. autonomicity. Complexity and autonomicity can not be viewed as two separate notions, but as the two sides of the same coin. On one hand, complexity calls for autonomicity especially in large dynamic systems. Since such systems are very complex by

nature, they require autonomic support in order for them to be managed in an economic manner; otherwise, it would seem impossible to manage them. On the other side of the coin, achieving full autonomicity in large systems (like the Internet) is very complex in terms of network engineering. Since the problem is too complex to tackle as a whole, a possible approach would be to divide the problem into many sub-problems, solve the sub-problems individually, and then merge the sub-solutions into a solution to the entire problem. However, while trying to "divide" a large, complex system in order to "conquer" it, there is always the danger to "divide" the problem in the wrong way.

One possible way to proceed would be to start with a "bottom-up" approach. That is, to build simple autonomic components that solve certain aspects of the overall problem space and then try to put them together to facilitate a good overall solution. Then, one should try to address issues like the possible interference between autonomic components that are put together, the extent to which the combination of two autonomic components form an autonomic component, the functionality that might be still missing and the extent to which the composite might be optimal. Finally, the above approach would require the application of an iterative/evolutionary methodology.

In order to manage the interoperability issues among different autonomic entities, the point was made that standards would be needed. The interaction among these autonomic entities should happen at various levels of the system hierarchy, depending on what is made autonomic each time and hopefully at not that many different levels. In the initial phase of the development of an autonomic system the required additional complexity might lead to an increase in the required capital expenses, but it is expected to lead to reduced operating expenses since it reduces management cost in the long-run.

The presentation in [3] addressed the panel question by contributing a comparison with the development of complexity in a well-known mass-market consumer device - the TV set. A short overview of the television history was presented as an example for how electronics industry dealt with the increasing complexity of electronic circuits.

Here we find early devices from the 30's with a very limited number of active elements (vacuum tubes), where functional overload was commonplace, and the signal to be received was designed in a shape that could be decomposed with such simple circuits. Over time, with the active elements getting cheaper (transistors), they increase in number, providing supportive and stabilising functions to the core functional blocks. However, without the technological need anymore, traditional functional overload remains.

With the appearance of ICs in the 80's, functional blocks get treated as black boxes, only specified by their interface parameters. While the internal complexity increases to hundreds of active elements, the task for the system designer becomes simpler, due to encapsulation. Reliability improves.

Regarding the functional overload, it is eventually the appearance of a new requirement, e.g. multi-sync computer screens, that leads to the separation of these functions into separate building blocks. Nowadays, with one-chip VLSI

TV-sets the complexity in this area has skyrocketed but the system design is easier and the reliability better than ever.

These landmarks in the history of television – representing iterative transitions that are characterized by an increase in complexity which is, however, encapsulated making the task of system design easier while at the same time increasing reliability – indicate that adding complexity to a complex system might be the answer to many of the problems that the complexity of the system creates.

The key question here is the following: can this successful, from the hardware perspective, approach be transferred to software? In order to answer this question several issues have to be addressed, such as the difference between hardware and software reliability and the definition of software quality and appropriate standards. Examples of self-healing hardware range from simple (e.g., capacitors where foil vaporizes at shortcut), to complex (hard disks reassign defective sectors), to redundant logic arrays (FPGAs) that are in an experimental phase.

As far as biological complexity and the potential impact of nature on autonomic research is concerned, it is argued that nature is complex (from quantum to the universe) and that there are several natural phenomena that could inspire autonomic research. Replication of information in every living cell poses the question as to whether abundance of stored information is needed. Ants are often quoted as examples for simple components forming a complex system; their behavior could be used as a paradigm for building pervasive systems providing redundancy and abundance. A plastic foil is simple but vulnerable, while the human skin is self-healing but far more complex. But are there any significant differences between human-made and natural complexity? If yes, one should keep that in mind when trying to apply the natural processes of autonomicity to the human-made world of engineering.

Finally, it is concluded that adding complexity to complexity (in the design) to achieve simplicity (for the end-user) might not be wrong, if it is well treated, with structure and encapsulation, with well-defined interfaces (APIs) hiding complexity.

As pointed in [4], network oriented R&D these days is largely driven by commercial interests, by expectations of new services that would natively support various types of mobility, user-centred ubiquity, personalisation and context awareness on top of increasing network heterogeneity. The IST FET proactive initiative on Situated and Autonomic Communication (S&AC) is a rare exception. The Commission provides this funding for a long-term basic research and the research community should take the opportunity for making the right strategic choices in research framework and road mapping. It is argued that a radical increase in complexity of network infrastructure is unavoidable, and that network autonomicity is the right solution within the new, service-oriented architecture.

In service-oriented computing, autonomous platform-independent computational entities are dynamically assembled into massively distributed evolvable systems. The enabler, the Service Oriented Architecture (SOA) is recognized as a mainstream trend in the design of software intensive systems. Are we prepared

for research and development towards network SOA, in which network-level services (features) will operate not only media and media signaling objects but business objects properly defined at each architecture level with proper cross-layer business relations?

As of now, application services are traded by ISPs to end-users on the retail interface with almost all needed trade sophistication in place; however, the required trade sophistication is missing at the wholesale interface between ISPs. Obviously, the end-to-end services are broken without network SOA that promises to turn complexity into in-network, self-organized trade sophistication. Within S&AC, the "end-to-end argument" should postulate that no functionality and/or intelligence that cannot self-recover should be placed inside the network; self-organization and self-recovery being the advantages of autonomicity.

To fulfil the promise, network SOA must support a variety of multi-tier dependencies between in-network state data and policies spanning generic Internet infrastructure, service- and application- specific infrastructures and propagating down the protocol stack to network functions and function surrounds that are translated at datagram level to media and media signalling processing workflows ultimately controlled by workflow access controls.

Finally, network SOA requires research and development towards two new abstraction layers – requirements abstraction and language abstraction – that must enable true end-to-end seamless inter-working of yet unknown advanced Internet services.

The presentation in [5] notes that ultimately Autonomic Systems aim to provide benefits by dramatically reducing operating costs of complicated systems. This is achieved primarily by off-loading the monitor-analyse-plan-execute operational cycle from human to system intelligence. Human operators then deal with managing high level policies, thus reducing the cognitive load on operators, allowing them to be more productive and inducing fewer mis-configuration errors.

However, autonomic systems themselves involve additional capital cost and are a source of additional complexity in system operation; thus, they are a potential source of further operational cost. In developing autonomic systems we, therefore, need a means to perform the cost-benefit evaluation on a given autonomic system in order to assess whether the additional capital and operational cost its deployment incurs is justified. In this respect, the autonomic communications research has yielded, to date, little in terms of guidance into suitable metrics and benchmarking techniques.

However, some evaluation criteria for autonomic computing systems [6] have been introduced, involving the metrics such as:

- the quality of service in achieving the primary goal, with emphasis on achieving user satisfaction;
- both the capital cost of acquisition and deployment, as well as the cost of operation over time;
- granularity and flexibility;
- the ability to avoid the negative cost and QoS impacts of operational failures

- the degree of autonomy in terms of the level of decision making that can be undertaken by the system rather than its human operators;
- adaptivity to changes in operational context and the latency in reacting to such changes;
- sensitivity to changes in operational context and the ability to attain operational stability after such operational perturbations.

The autonomic communications therefore needs to develop a comprehensive and holistic set of benchmarks that address the total cost of ownership. This may build on existing frameworks, such as TL9000, but must focus on the critical assessment of the introduction of any autonomic feature on the overall cost of ownership. This must consider interaction between all the "self-*" attributes; e.g., does a cost saving through a self-configuration feature render the system more vulnerable to attacks, thus making self-protection more problematic?

This is a challenging proposition as such operational cost - benefits analyses need to be performed over ever-changing network technologies, service portfolios and multi-provider value chains. More fundamentally, with many of the technologies currently being addressed, the solution seems to move from handing complicated systems to exploiting complexity, such that emerging behaviour in multi-agent systems is exploited. This presents a major cultural shift for network operators, which currently strive for full understanding of complicated systems, to one where they rely on the statistical behaviour of complex ones and thus a level of constrained non-determinism. Effective benchmarking also requires that we greatly improve our understanding of the lifecycle engineering costs for self-organising, adaptive systems, in terms of re-use, re-tasking and ameanability to innovation.

Finally, the presentation in [7] argues that complexity is not introduced by autonomicity but it is inherent in complex structures associated with today's networks. As an example, the case of mobile ad hoc networks is presented. In such networks there are numerous sources of complexity, such as setting the numerous parameters at the several layers. It is typically very difficult to determine the optimal setting for a given environment and this task becomes almost impossible when dealing with changing environments. Here is where autonomicity can have an important role by adding the "control stability" complexity in exchange for simplifying the parameter tuning.

"Control stability" complexity manifests itself in several ways. For instance, for the mobile ad hoc networking environment the feedback loop has to deal with many "conflicting" concerns such as forwarding, reliable delivery, resource sharing, channel access and utilization, security and trust management, etc. The various control knobs interact with each other at possibly fairly diverse time scales. The question that naturally arises is as to why one should go through all this "control stability" complexity. There is a very good reason for this: adapting to the environment can result in a great performance improvement!

Since complexity is unavoidable the key question is how to handle it. The general principle should be to keep it as simple as possible (KISS: Keep It Simple Stupid). A good approach would be to try to decouple the system's "intelligence"

from "interaction monitoring". The boundaries introduced by such decoupling would also prevent the appearance of control loops and instabilities. Such a decoupling can be observed in the human nervous system: one part reasons and a different one monitors sensory information and reacts.

To reduce complexity, different levels of "intelligence" in the nodes could also be considered, allowing for simple instantiations first, that will be open to extensions, as well as be enhanced with more sophistication as nodes (and designers) evolve and learn over time. It should be noted that dumb individuals (or low "intelligence" nodes) may result in smart group behavior, as is the case, e.g., with ants. Simple users can still adapt/mutate and be excellent for a particular goal. There is an analogy here with FPGAs – as fast as specialized DSP but with the versatility of "multipurpose" microprocessors.

References

1. P. Spirakis, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, availabe at `http://www.di.uoa.gr/~istavrak/PDF_presentations_WAC/Panel1_wac_Spirakis.pdf`.
2. S. Schmidt, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, available at `http://www.di.uoa.gr/~istavrak/PDF_presentations_WAC/Panel1_WAC2005_StefanSchmid.pdf`.
3. T. Pfeifer, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, available at `http://www.di.uoa.gr/~istavrak/PDF_presentations_WAC/Panel1_Pfeifer_WAC2005.pdf`.
4. M. Smirnov, R. Popescu-Zeletin, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, available at `http:// www.di.uoa.gr/~istavrak/PDF_presentations_WAC/Panel1_WAC2005_Smirnov.pdf`.
5. D. Lewis, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, available at `http://www.di.uoa.gr/~istavrak/ PDF_presentations_WAC/Panel1_wac05-lewis.pdf`.
6. McCann, J, Huebscher, Evaluation Issues in Autonomic Computing, GCC 2004, LNCS 3252.
7. C. Santivanez, presentation at the panel "Autonomicity vs. Complexity", 2^{nd} IFIP Workshop on Autonomic Communication (WAC2005), Oct. 2005, Athens, Greece, available at `http://www.di.uoa.gr/~istavrak/PDF_presentations_WAC/Panel1_WAC2005_Santivanez.pdf`.