Self-* and Predictability: are these conflicting system capabilities?

Rogério de Lemos

Computing Laboratory
University of Kent at Canterbury, UK
r.delemos@kent.ac.uk

Abstract

In principle, it would have been desirable for any system to have self-* capabilities when faced with changes that might occur either in the system or its environment. However, these capabilities are difficult to incorporate in certain classes of systems, mainly those in which uncertainties in their behaviours are not desirable. In this position paper, we discuss how the expected degree of autonomy of a system is related to the way in which a system can be described. The discussion is presented in the context of predictability and the assurances associated with the accuracy of that predictability. The subject of this paper are those systems known to be critical in the services they deliver, and which cannot incorporate high degree of uncertainty.

1. Introduction

As computer based systems become more complex, design solutions that promote their operational autonomy have become the holy grail of system architects and designers. Moreover, system autonomy should not be restricted to the actual services delivered by the system, but it should also be associated with the infrastructure that enables the system to deliver its services with a certain degree of quality. In other words, although autonomy is not something that is observed at the system interface, it should nevertheless enable the system to provide its services as specified.

Self-* capabilities are the means by which a system attains its autonomy, and these capabilities have an impact upon the system's fundamental properties, which are functionality, usability, performance, cost, and dependability. In this paper, properties and capabilities are considered to be two different concepts, the former is a system attribute that can be directly measured and quantified, while the latter are the means that enable one or more of the system properties. Although some aspects of self-* capabilities might allow the system to provide different kinds of services, the view taken in this paper is that system capabilities are essentially enablers for improving the system non-functional properties. In addition to self-* capabilities, the system might have other capabilities whose purpose is also to enhance the quality of services provided by the system, and reduce the cost for providing them.

This paper explores the relationship between self-* capabilities and predictability, which is a fundamental aspect in the design, operation and evaluation of a certain class of systems. The objective of this exercise is to identify, from the perspective of predictability, what type of systems should the self-* capabilities be associated with, and what techniques could be used to attain such capabilities. This would allow establishing the theoretical and practical limits that can be associated with the different approaches that are employed in the development of systems.

Systems in general can be described either in terms of *process* or *data* [8][11]. While the former characterises the system as acted upon, the latter characterises the system as sensed [11]. A major difference between these two forms of description is the amount of information required for modelling a system. The process description of a system usually involves less information than its counterpart. Moreover, it might be the case that due to its very nature, the data description of systems becomes unbounded, which is not the case in the process description of systems.

These two forms of system representation will be used as a basis for investigating the provision of self-* capabilities in different classes of systems. Depending on the system, expectations on the degree of autonomy of a system might be curtailed depending on the assumptions made and the properties of that system. In the next section, we present how self-* capabilities can be incorporated into systems that are essentially represented as process and data, and finally, we present some concluding remarks.

2. Process and Data Representation of Systems

The complexity of systems and the way they are integrated will require new approaches for their development, operation and maintenance. Conventional deterministic approaches may not be sufficient for enabling the provision of a wide range of services that are expected from these systems. Several new approaches have recently emerged from different areas, such as, biologically inspired computing, software engineering and agent technology, just to mention a few. In this paper, we restrict ourselves to the first two areas. Issues related to agent technology and dependability were recently discussed on two panels [4][7].

The provision of self-* capabilities by software engineering solutions essentially relies on the representation of systems as processes, in which solutions are normally based on the feedback control loop principle. Meta-parameters of system behaviour and structure, and its environment, are monitored for eventual changes so that the system can be adjusted for delivering required services in a stable way. The degree of self-* capabilities that can be achieved by employing these solutions is limited because of the need for having predictive behaviours, otherwise reaction to changes would not be deterministic. Predictability is achieved by removing operational uncertainties from the system otherwise these could disrupt the normal operation of the system. In other words, it is fundamental that during the development of these systems the complete state specification is identified, or else the occurrence of unexpected states can lead to system failures. Considering such restrictions, can a process oriented system be able to show self-* capabilities? They might be able, but the degree of autonomy is restricted, and it might be the case that these capabilities need to be established during design time.

On the other hand, the provision of self-* capabilities by biologically inspired solutions essentially relies on the representation of systems as data. Since these solutions are based on a sample of the whole data associated with a system, complete system models are difficult to obtain, which explains why uncertainties are an inherent aspect of these models. Incorporating learning capabilities into a system might eliminate this deficiency, however these are likely to introduce another degree of uncertainty. Emergent behaviours might be useful in dealing with unexpected circumstances, but the system reaction to these might become unpredictable.

It has been claimed that data oriented approaches might be appropriate for new emerging applications, but in what capacity is not yet clear. One issue however is clear, if predictability has to be an essential capability in the development and operation of a system, then a data oriented approach might not be an appropriate solution. This is particularly significant in those classes of systems in which performance and dependability constraints are critical. However, data oriented approaches could nevertheless be employed in such systems if sufficient protections are incorporated into their designs. Again such conservative solution would restrict one of the major benefits of data oriented approaches, which is that of emergent behaviours. An alternative approach, yet not fully explored, would be to build massively redundant systems, in which the failure of some the components would not affect the expected outcome of the whole system. However, for such solution to be successful, diverse data oriented approaches should be composed in order to increase their combined effectiveness, or coverage. However, a major weakness in such configuration would be the quality of the training data. If the data is not good, it does not matter how many approaches are employed if all of them suffer the same deficiencies.

Still considering the idea of exploring data oriented approaches in the context of systems containing trillions of components, issues like the identification of the source of change is important for establishing the appropriate mechanisms to deal with the change. For example, changes that occur internally to a component and that eventually affect the behaviour of that component, how these should be handle in the wider system? If the rest of the system was able to accommodate the unknown behaviours, what should be the threshold to which the system should react either for eliminating a whole group of abnormal components, or incorporating these components as normal? The reverse also raises very interesting questions. If the environment of a system changes, how these changes are reflected upon the components of that system: either the components are eliminated from the system, or the components have to be modified for coping with the changes. All these decisions affect the predictability of the overall system behaviour if clear strategies are not implemented. However, as already mentioned, it might be the case that the combined usage of diverse strategies might be the only way of bringing out the best of the system, which eventually might lead to unpredictabilities.

In the following, we present two approaches that serve as an example of process and data oriented systems, respectively.

2.1. Architectural Approaches

Architectural representations of systems have shown to be effective in assisting the understanding of broader system concerns by abstracting away from details of the system. To leverage the dependability properties of systems, solutions are

needed at the architectural level that are able to guide the structuring of undependable components into a fault tolerant architecture. Fault tolerance, one of the means to dependability, is related to the self-repair and self-healing capabilities [1].

Architectural flexibility for supporting run time change can be achieved by using specialised co-operative connectors to change the pattern of collaboration between components: components are rigid entities, and how they interact provide the basis for adaptability [3][5]. Each collaboration is identified in terms of pre- and post-conditions, and invariants. Depending on the required change, a different collaboration is selected that makes the system to change its behaviour. All the collaborations are defined during design time together with their respective trigger conditions. Uncertainty between the alternative collaborations does not exist because choice has to be deterministic, and uncertainties associated with a particular collaboration is restricted because behavioural invariants have to be maintained.

In a different work, in order to deal with undesirable, though expected circumstances, an idealised architectural component was defined with structure and behaviour equivalent to that of the idealised fault-tolerant component concept [8]. This approach was later extended to deal with commercial off-the-shelf (COTS) software components [9]. The basic mechanism to deal with the expected circumstances employed in these approaches was exception handling. The system architect must know from the outset what exceptions might occur, the causes associated with these exceptions, and there is the need to match these exceptions with their respective handlers. The predictability in these systems is obtained by clearly identifying what is expected, and avoiding the system to become brittle towards the unexpected. How a system reacts towards expected circumstances should be know beforehand and should be incorporated in the design of the system.

Alternative techniques could be employed if undesirable circumstances, i.e. faults, could be grouped in terms of classes. Instead of the need for identifying specific handlers for each type of undesirable circumstance, as mentioned above, general solutions based on replication, diversity, and consensus could be devised. However, although these systems would be robust towards certain classes of faults, they are not considered sufficiently robust towards any class. In all these approaches, there is almost no degree of autonomy for the sake of obtaining predictable behaviour, which was an essential requirement of the applications involved.

2.2. Artificial Immune Systems

Artificial immune systems are adaptive systems inspired by theoretical immunology and observed immune functions, principles and models, which are applied to complex problems [2]. A number of works have attempted to build artificial immune systems for fault tolerance, virus detection, and computer security. In particular, the creation of immunised fault tolerant embedded systems has been proposed [13], which explores negative selection, an immune inspired algorithm, for the generation of error detectors [1]. More recently, this work has been extended to incorporate the capability of generating adaptable error detectors during run time, thus providing the means for the system to adapt itself to previously unexpected and undesirable circumstances. The incorporation of this capability has come to a price: the accuracy in detecting erroneous states has decreased when compared with that of an equivalent well craft engineered system; other studies have drawn the same conclusions [11].

The application of data oriented approaches to error detection, that could be either the consequence of faults or intrusions, clearly illustrates the limitations associated with these approaches. Since faults and intrusions are considered rare events, the question to be asked is how the system is able to learn from rare events. If some correlation could be established between rare events, then the process of identifying new undesirable events could be based on the extrapolation of what is already known. However, this assumption cannot be generalised, since it is difficult to establish the correlation between undesirable events. An alternative approach could be that of learning new undesirable events from what is already known about the non-erroneous behaviours of the system. This is a daunting challenge if we consider that the state space of normal behaviours might be much larger if compared with that of abnormal behaviour. Either the normal behavioural state can be encoded in such way that facilitates the search process, or such an approach becomes prohibitive in terms of efficiency and storage.

Another problem that is inherent in data oriented approaches is the data itself. In addition for the need to the data to be representative of the actual system, there is also the need to have a deep understanding what the data represents. If either of these issues are not observed the predictability of the system is affected.

3. Conclusions

Although in this paper, the issues concerning self-* capabilities of systems were presented in terms of the dichotomy on how systems are represented, i.e.

process versus *data*, we have not overlook the possibility of systems relying on both representations for achieving different degrees of autonomy depending on the services to be delivered. The idea of developing systems that rely on both process and data representations, which explores the complementary benefits of these, is not new. Such hybrid systems have mostly been confined to stand alone closed systems, however the challenge ahead is whether the same idea can be applied to more complex systems that are open and collaborative in their nature, and which are expected to show self-* capabilities and be predictive at the same time.

References

[1] M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, R. Duncan. "Negative Selection: How to Generate Detectors". *Proceedings of the 1st International Conference on Artificial Immune Systems*. Canterbury, UK. September 2002. pp. 89-98.

[2] L. N. de Castro, J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag. 2002.

[3] R. de Lemos. "A Co-operative Object-Oriented Architecture for Adaptive Systems". *Proc. of the 7th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'00)*. Edinburgh, Scotland. April 2000. pp. 120-128.

[4] R. de Lemos. "Novel Approaches in Dependable Computing". *Proceedings of the 4th European Dependable Computing Conference (EDCC-4)*. Lecture Notes in Computer Science 2485. P. Thevenod-Fosse and A. Bondavalli (Eds.). Springer-Verlag. Toulouse, France. October 2002. pp. 79-80.

[5] R. de Lemos, J. L. Fiadeiro. "An Architectural Support for Self-adaptive Software for Treating Faults". *Proceedings of the 1st ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02)*. A. Wolf, D. Garlan, J. Kramer (Eds.). Charleston, SC, USA. November 2002. pp. 39-42.

[6] R. de Lemos, C. Gacek, and A. Romanovsky (Eds.). *Proc. of the ICSE 2003 Workshop on Software Architectures for Dependable Systems*. Portland, OR. April 2003. http://www.cs.ukc.ac.uk/events/conf/2003/wads/ (October 2003).

[7] A. Garcia, J. Sardinha, C. Lucena, J. Castro, J. Leite, R. Milidiú, A. Romanovsky, M. Griss, R. de Lemos, A. Perini. "Software Engineering for Large-Scale Multi-Agent Systems – SELMAS 2003: Workshop Report". *ACM Software Engineering Notes 28(5)*. November 2003.

[8] P. A. de C. Guerra, C. Rubira, and R. de Lemos. A Fault-Tolerant Software Architecture for Component-Based Systems. *Architecting Dependable Systems*. Lecture Notes in Computer Science 2677. Springer. Berlin, Germany. 2003. pp. 129-149.

[9] P. A. de C. Guerra, C. Rubira, A. Romanovsky, and R. de Lemos. Integrating COTS Software Components into Dependable Software Architectures. *Proc. of the 6th IEEE Int. Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)*. Hokaido, Japan. May 2003.

[10] A. G. J. MacFarlane. "Information, Knowledge and Control". *Essays on Control: Perspective in the Theory and its Applications*. Eds. H. L. Trentelman, J. C. Willians. Birkhäuser. 1993.

[11] R. A. Maxiom, K. M. C. Tan. "Anomaly Detection in Embedded Systems". *IEEE Transactions on Computers 51(2)*. February 2002. pp. 108-120.

[12] H. A. Simon. *The Sciences of the Artificial*. Second Edition. MIT Press. Cambridge, MA, USA. 1981.

[13] J. Timmis, R. de Lemos, M. Ayara, R. Duncan. "Towards Immune Inspired Fault Tolerance in Embedded Systems". *Proceedings of 9th International Conference on Neural Information Processing*. IEEE Computer Society. November 2002. pp. 1459-1463.