SPIE Newsroom

10.1117/2.1xxxxxx.xxxx

Designing and controlling trustworthy self-organising systems

Giovanna Di Marzo Serugendo and John Fitzgerald

A software architecture and a development method for engineering self-organising systems that can be justifiably trusted and controlled.

A self-organising system can arrange itself and modify its behaviour without receiving specific instructions to do so. Such systems are common in nature: flocks of birds responding to wind changes or a colony of ants structuring itself in response to a threat. But self-organisation is not only seen in nature – increasingly, artificial systems such as robots, mobile networks and software services are able to self-organise, enabled by modern computing and network technologies. Such artificial self-organising systems show some of the adaptability of their natural counterparts, but their behaviour is hard to control (to stop, reset or guide) and even harder to predict in advance. So - can such systems be trusted? The challenge in our work is to provide means of designing and controlling artificial self-organising systems so that there is enough evidence to justify placing reliance on them to perform safely, correctly and efficiently, even in the presence of erratic behaviour by the environment or faulty components.

The achievement and demonstration of system dependability is a well established field of study.[1] The main techniques avoid the introduction of defects during design, use over-engineering to tolerate faults should they arise anyway, and detect remaining faults through system verification. These techniques produce evidence that can form the basis of a system's "dependability argument". Most of these techniques assume a static system structure fixed during design, while real self-organising systems are dynamic, with components and agents joining and leaving, changing goals and reacting to events. Specific approaches targeting self-* systems vary from multi-layer reference architectures for self-adaptive systems[2], to analysis guidelines and specific agent-based solutions.[3] They do not address trustworthiness and controllability. To bridge this gap, we have been working on a software architecture and development method that permits the definition and analysis at design-time of mechanisms that both ensure and constrain the run-time behaviour of a self-organising system, thereby providing some assurance of its self-* capabilities.

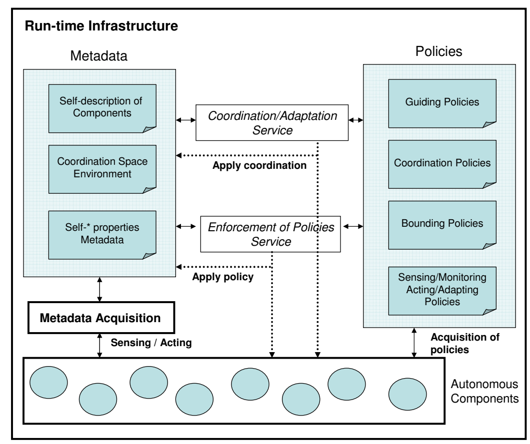

Figure 1. *Architecture involving loosely coupled components, metadata and policies.*

We view the self-organising system as a collection of loosely coupled autonomous components. We gather and maintain metadata that describes characteristics such as components' functional specifications or non-functional characteristics such as availability levels and environment-related metadata such as artificial pheromone. The behaviour of the system, for example reconfiguration to compensate for a component failure, is governed by policies that describe the response of system components to detected conditions and changes in the metadata. When the system is "live", both the components and the run-time infrastructure exploit metadata to support decision-making and adaptation in accordance with the policies.[4]

In order to realise this approach, we have proposed a *system architecture* (Figure 1) that involves autonomous components, repositories of metadata and executable policies, and reasoning services that dynamically enforce the policies on the basis of metadata values.[5] Metadata may be stored, published and

Continued on next page

updated at run-time both by the run-time infrastructure and by the components themselves. Policies are available at run-time to both the run-time infrastructure and the components themselves. Guiding policies are high-level goals (e.g. starting or stopping a swarm of robots); bounding policies define environmental limitations; sensing/monitoring policies define reflex behaviour for the components (e.g. if metadata value reaches a threshold then an action must be taken).

Policies may be generic, e.g. replacing a current (slow) component with an equivalent component having a higher performance. By accessing metadata about current performance of the components, the reasoning engine can actually determine which of the available components has to replace the failing one. In principle, policies can change dynamically, although allowing unconstrained change can affect dependability!

We have defined a *development method* in which the requirement and analysis phase identifies the functionality of the system along with self-* requirements – where and when self-organisation is needed/desired. A design phase determines the autonomous components (services, agents, etc) design and the mechanisms governing the autonomous components' interactions and behaviour (e.g. stigmergy, trust or gossip), addressing the self-* requirements. The implementation phase produces the run-time infrastructure.[5]

We have applied our approach in two case studies. First, we have developed dynamically resilient Web services where a client, requesting a specific Web service, specifies its choice of dependability at run-time, e.g. the Web Service with the best dependability metadata is selected as the primary service and the others used as alternatives if that one fails.[6] Second, we have designed self-organising robotic assembly systems. In response to an incoming product order, robotic modules self-organise - select each other and re-program themselves - to form an ad hoc assembly system able to produce the specified product. During production, the modules self-adapt to ensure production also in degraded modes - adapt each other speeds or take over from a faulty module.[7]

Our approach is designed to promote predictability and control in artificial self-organisisng systems. *Predictability* is primarily obtained by the dynamic enforcement of policies instantiating the self-organising and resilience mechanisms identified at design-time. Policies turn out to be a useful tool for analysing the emergent properties of the design. The construction of compositional proofs of emergent properties depends on the level of rigour used in the policy and metadata definitions. *Low-level control* results from the activity of the components. Components sense and retrieve metadata and policies. Their behaviour causes metadata changes which in turn cause components to adapt

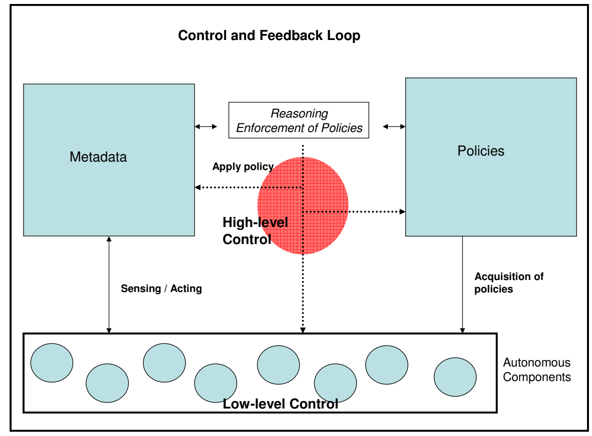

Figure 2. *Control occurs through active modification of metadata, policies and components*

to the new situation. The run-time infrastructure itself is active and through reasoning services enforces active (possibly human) *high-level control* by direct reconfiguration of components; modification of the metadata; and modification of the policies used by the components for driving (changing) their behaviour on-the-fly (Figure 2). Loose coupling is crucial: changing a policy or metadata occurs without modifying/stopping the components, their new value immediately affecting the components' behaviour.

Self-organising mechanisms are an attractive paradigm for engineering robust artificial systems from simple individual components, but their very flexibility challenges our ability to predict and control their behaviour, and hence their trustworthiness. We have defined a software architecture and established a development method that addresses predictability by exploiting metadata to support decision-making and adaptation based on the dynamic enforcement of explicitly defined policies. Control is obtained by actively modifying metadata, policies or components. Future work will concentrate on enhancing predictability by formal analysis of policies and on the spontaneous production of new policies at run-time through reasoning over an internalised model of the self-organising system.

Acknowledgements
Our work has been conducted in collaboration with Alexander Romanovsky (Newcastle University) and Nicolas Guelfi (University of Luxembourg). The EU-funded coordination action Perada [8] supported travel exchanges between Uninova and Birk-

Continued on next page

beck College for developing self-organising assembly systems.

Author Information

Giovanna Di Marzo Serugendo
School of Computer Science and Information Systems
Birkbeck College, University of London
London, UK

Dr Giovanna Di Marzo Serugendo is lecturer at Birkbeck. She has a PhD in Software Engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL). Her research interests are related to the engineering of self-* systems. She co-founded the IEEE International Conference on Self-Adaptive and Self-Organising Systems (SASO) and is the editor-in-chief of the ACM Transactions on Autonomous and Adaptive Systems.

John Fitzgerald
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK

Dr John Fitzgerald is Reader in Computing Science at Newcastle University. His research concerns formal modelling and its potential to help master the complexity of building provably resilient and fault-tolerant systems, particularly those that are reconfigurable. He currently leads work on resilience in the EU FP7 Integrated Project "Deploy". He has a PhD in Computer Science from Manchester University, UK, and is Chairman of Formal Methods Europe.

References

1. A. Avizienis, J.-C. Laprie, B. Randell, and L. C., *Basic Concepts and Taxonomy of Dependable and Secure Computing*, **IEEE Transactions on Dependable and Secure Computing 1** (1), pp. 11–33, 2004.
2. J. Kramer and J. Magee, *Self-Managed Systems: an Architectural Challenge*, In Future of Software Engineering (FOSE'07), pp. 259–268, IEEE Computer Society, 2007.
3. G. Picard and M.-P. Gleizes, *The ADELFE Methodology - Designing Adaptive Cooperative Multi-Agent Systems*, In Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook, pp. 157–176, Kluwer Publishing, 2004.
4. G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi, *A Metadata-Based Architectural Model for Dynamically Resilient Systems*, In ACM Symposium on Applied Computing (SAC'07), pp. 566–572, 2007.
5. G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi, *MetaSelf - A Framework for Designing and Controlling Self-Adaptive and Self-Organising Systems* Tech. Rep. BBKCS-08-08, School of Computer Science and Information Systems, Birkbeck College, London, UK, 2008.
6. Y. Chen and A. Romanovsky, *Improving the Dependability of Web Services Integration*, **IT Professional 10** (3), pp. 29–35, 2008.
7. R. Frei, G. Di Marzo Serugendo, and J. Barata, *Designing Self-Organization for Evolvable Assembly System*, In IEEE International Conference on Self-Adaptive and Self-Organising Systems (SASO'08), pp. 97–105, IEEE Computer Society, 2008.
8. http://www.perada.eu