Special Contribution

Towards robust self-managed systems*1

Jeff KRAMER[1] and Jeff MAGEE[2]
[1,2]Department of Computing Imperial College

Software is pervasive. It plays a part in every aspect of our lives, forming a significant part of every automated system or tool that we use, from washing machines to healthcare systems. Like the washing machine, some of these applications are simple, self-contained devices; however the majority are far more complex. They are generally distributed systems which rely on interacting, distributed subsystems of software components to perform their function. They are expected to interact with users with changing needs and with other systems with unreliable service provision. The challenge is to provide these software systems in such a way that they are robust in the presence of major issues such as change and complexity.

Change is inherent, both in the changing needs of users and in the changes which take place in the operational environment of the system. Hence it is essential that our systems can adapt as necessary to continue to achieve their goals. Change is also induced by failures or the unavailability of parts of the system. It is therefore necessary to envisage dynamically changing configurations of software components so as to adapt to the current situation. Dynamic change, which occurs while the system is operational, requires that the system evolves dynamically, and that the adaptation occurs at run-time.

Complexity requires that we use rigorous techniques to design, build and analyse our software and thereby avoid unnecessary design flaws. This implies the need for analytical techniques which cope with changing goals and the changing compositions of adaptive software.

Since the complexity and response times required by the changes may not permit human intervention, we must plan for automated management of change. The systems themselves must be capable of determining what system change is required, and in initiating and managing the change process wherever possible. This is the aim of self-managed systems.

What is Self-Management?

Self-managed systems are those that capable of self-configuration, self-adaptation and self-healing, self-monitoring and self-tuning, and so on, also referred to as self-∗ or autonomic systems.

The aim of self-configuration is that the components should either configure themselves such that they satisfy the specification of the goals, properties and constraints that you expect your system to achieve or be capable of reporting that they cannot. If the system suffers from changes in its requirements specification or operational environment such as changes in use, changes in resource availability or faults in the environment or in parts of the system itself, then the aim of self-adaptation and self-healing is that the system should reconfigure itself so as to again either satisfy the changed specification and/or environment, or possibly degrade gracefully or report an exception. Note that the specifications should include not only functional behaviour, but also those non-functional properties such as response time, performance, reliability, efficiency and security, and that satisfaction of a specification may well include optimisation.

What research is currently being conducted?

Different research communities are engaged in relevant research, investigating and proposing approaches to various aspects of self-management for particular domains. For instance, in the networking, distributed systems and services community, there has been the Autonomic Computing conferences [3] and more recently, the SelfMan Workshop 2006 [2] to discuss and analyse the potential of self-∗ systems for managing and controlling networked systems and services. Dobson et al. [11] provide a recent survey on autonomic com-

Received December 18, 2007.
1) *j.kramer@imperial.ac.uk*, 2) *j.magee@imperial.ac.uk*
*1 This paper is based on [18]
DOI: 10.2201/NiiPi.2008.5.1

munications, and propose an autonomic control loop (a phased approach) of actions *collect* (monitoring), *analyse*, *decide* and *act*, a cycle which naturally appears in many proposed approaches.

In the software engineering community, there has been a series of workshops which started in the distributed systems community with the CDS (Configurable Distributed Systems) conferences [4], [5], [8] and more recently with WOSS (Workshop on Self-Healing and Self-Managed Systems) [6], [7] and SEAMS (Software Engineering for Adaptive and Self-Managing Systems) [1]. Other interested research communities include the intelligent agent, machine learning and planning communities. Huebscher and McCann [15] provide an excellent and comprehensive survey on autonomic computing.

However, although research has provided much that is useful in contributing towards self-management, the general and fundamental issues of providing a comprehensive and integrated approach remains.

What approach do we advocate?

We believe that an architecture-based approach offers potential benefits such as generality, where the underlying concepts and principles should be applicable to a wide range of application domains, each with its own peculiar architecture. Architectures are also designed to handle issues of scalability and complexity — by supporting abstraction and separation of concerns.

Many others also advocate the use of a component-based architectural approach. For instance, Oreizy et al. [21] provide a general outline of an architectural approach which includes adaptation and evolution management; Garlan and Schmerl [12] describe the use of architecture models to support self-healing; Dashofy, van der Hoek and Taylor propose the use of an architecture evolution manager to provide the infrastructure for run-time adaptation and self-healing in ArchStudio; [10] and Castaldi et al. [9] extend the concepts of network management to component-based, distributed software systems to propose an infrastructure for both component- and application-level reconfiguration using a hierarchy of managers.

Our own work has concentrated on the use of ADLs (architecture description languages) for software design and implementation from components [19], including limited language support for dynamic change [20], a general model for dynamic change and evolution [17], associated analysis techniques [16] and initial steps towards self-management [14].

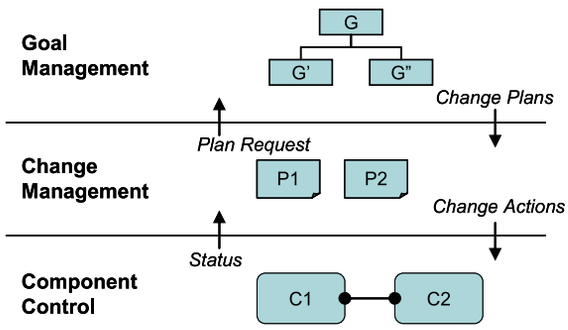

Fig. 1 Three layer architecture model for self-management.

What is the essence of our proposed architectural model for self-management?

A self-managed software architecture is one in which components automatically configure their interaction in a way that is compatible with an overall architectural specification and achieves the goals of the system. The objective is to minimise the degree of explicit management necessary for construction and subsequent evolution whilst preserving the architectural properties implied by its specification. More details of our approach can be found in [18], [22]; we provide a summary below.

Based on work by Gat [13] in robotics, we propose the use of a three layer reference architecture (see Fig. 1). This provides the necessary separation of concerns for a rigorous engineering approach in which low-level actions can be clearly and formally related to high-level goals that are precisely specified.

The bottom layer is *Component Control*, which consists of the set of interconnected components that accomplish the application function of the system. It includes facilities to report the current status of components to higher layers, to adjust the operating parameters of components and for modification by component creation, deletion and interconnection. An important characteristic of this layer, is that when a situation arises that the current configuration of components is not designed to deal with, this layer detects this failure and reports it to higher layers.

The middle layer is *Change Management* which is responsible for effecting changes to the underlying component architecture in response to new states reported by that layer or in response to new objectives required of the system introduced from the layer above. It consists of a set of reactive plans, each giving an action or sequence of actions to handle the new situation. This layer can introduce new components, recreate failed components, change component interconnec-

tions and change component operating parameters. The layer can respond quickly to new situations by executing what are in essence pre-computed plans. If a situation is reported for which a plan does not exist then this layer must invoke the services of the higher planning layer. In addition, new goals for a system will involve new plans being introduced into this layer.

The uppermost layer is *Goal Management* which is responsible for change planning. This takes the current state and a specification of a high-level goal and attempts to produce a plan to achieve that goal. This layer produces change management plans in response to requests from the layer below and in response to the introduction of new goals.

In addition to the separation of concerns, one of the criteria for placing functionality in different layers in our self managed systems architecture is that of time. Immediate feedback actions are at the lowest level and the longest actions requiring deliberation are at the uppermost level.

We would emphasize that we do not consider this an implementation architecture but rather a conceptual or reference architecture which identifies the necessary functionality for self management. It also provides a context for discussing some of the main research challenges which self-management poses. At the component layer, the main challenge is to provide change management which reconfigures the software components, ensures application consistency and avoids undesirable transient behaviour. At the change management layer, decentralized configuration management is required which can tolerate inconsistent views of the system state, but still converge to a satisfactory stable state. Finally, some form of on-line (perhaps constraint based) planning is required at the goal management layer.

In Conclusion . . .

There has been much progress towards providing self-managed systems, but much remains to be done to provide an integrated and comprehensive solution, supported by an appropriate infrastructure. In addition, the approach must be amenable to a rigorous software development approach and analysis, so as to ensure preservation of desirable properties and avoid undesirable emergent behaviour. We believe that an architectural approach offers a suitable and promising framework for working on each of the relevant research issues.

References

[1] *2nd IEEE Int. Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2007)*, ICSE, Minneapolis, 2007.

[2] *2nd IEEE Int. Workshop on Self-Managed Networks, Systems and Services (SelfMan 2006)*, IEEE, Dublin, 2006.

[3] *The 3rd IEEE International Conference on Autonomic Computing* IEEE, Dublin, 2006.

[4] *Proceedings of IEE/IFIP 1st Int. Workshop on Configurable Distributed Systems (CDS 92)*, in J. Kramer, ed., London, May 1992.

[5] *Proceedings of IEEE 3rd International Conference on Configurable Distributed Systems (CDS 96)*, in J. Magee and K. Schwan, eds., May 1996.

[6] *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, in D. Garlan, J. Kramer and A. Wolf, eds., ACM Press, Newport Beach, California, 2004, pp. 119.

[7] *Proceedings of the first workshop on Self-healing systems*, in D. Garlan, J. Kramer and A. Wolf, eds., ACM Press, Charleston, South Carolina, 2002, pp. 120.

[8] *Proceedings. of IEEE 2nd International. Conference on Configurable Distributed Systems, Pittsburgh, (CDS 94).* in J. Kramer and J. Purtilo, eds., Pittsburgh, May 1994

[9] M. Castaldi, A. Carzaniga, P. Inverardi and A. L. Wolf, *A light-weight infrastructure for reconfiguring applications*, *Proceedings of 11th Software Configuration Management Workshop (SCM03)*, LNCS, Portland, Oregon, 2003.

[10] E. M. Dashofy, A. van der Hoek and R. N. Taylor, *Towards architecture-based self-healing systems*, *Proceedings of the first workshop on Self-healing systems*, ACM Press, Charleston, South Carolina, 2002.

[11] S. Dobson, S. Denazis, Fernndez, Antonio, D. Gati, E. Gelenbe, Massacci, P. Nixon, F. Saffre, N. Schmidt and F. Zambonelli, “A survey of autonomic communications”, *ACM Trans. Auton. Adapt. Syst.*, vol. 1, pp. 223–259, 2006.

[12] D. Garlan and B. Schmerl, *Model-based adaptation for self-healing systems*, *Proceedings of the first workshop on Self-healing systems*, ACM Press, Charleston, South Carolina, 2002.

[13] E. Gat, *Three-layer Architectures*, *Artificial Intelligence and Mobile Robots*, MIT/AAAI Press, 1997.

[14] I. Georgiadis, J. Magee and J. Kramer, *Self-organising software architectures for distributed systems*, *Proceedings of the first workshop on Self-healing systems*, ACM Press, Charleston, South Carolina, 2002.

[15] M. C. Huebscher and J. McCann, *A survey of Autonomic Computing—degrees, models and applications*, ACM Computing Surveys, to appear. (2008).

[16] J. Kramer and J. Magee, *Analysing dynamic change in distributed software architectures*, Software, IEE Proceedings–, vol. 145, pp. 146–154, 1998.

[17] J. Kramer and J. Magee, *The evolving philosophers problem: dynamic change management*, Software Engi-

neering, IEEE Transactions on, vol. 16, pp. 1293–1306, 1990.

[18] J. Kramer and J. Magee, *Self-Managed Systems: an Architectural Challenge*, in L. Briand and A. L. Wolf, eds., *Future of Software Engineering 2007*, IEEE-CS Press, 2007.

[19] J. Magee, N. Dulay, S. Eisenbach and J. Kramer, *Specifying Distributed Software Architectures*, *5th European Software Engineering Conference (ESEC)*, Sitges, Spain, 1995.

[20] J. Magee and J. Kramer, *Dynamic structure in software architectures*, *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering*, ACM Press, San Francisco, California, United States, 1996.

[21] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum and A. L. Wolf, *An architecture-based approach to self-adaptive software*, Intelligent Systems and Their Applications, IEEE [see also IEEE Intelligent Systems], vol. 14, pp. 54–62, 1999.

[22] D. Sykes, W. Heaven, J. Magee and J. Kramer, *Plan-Directed Architectural Change For Autonomous Systems*, *Sixth International ACM Workshop on Specification and Verification of Component-Based Systems, SAVCBS'07*, Dubrovnik, Croatia, 2007.

Jeff KRAMER

Jeff KRAMER is Dean of the Faculty of Engineering at Imperial College London, and was Head of the Department of Computing from 1999–2004. His research interests include rigorous techniques for requirements engineering; software specification, design and analysis; and software architectures, particularly as applied to distributed and adaptive software systems. Jeff is the Editor-in-Chief of the IEEE Transactions on Software Engineering, and the co-recipient of the 2005 ACM SIGSOFT Outstanding Research Award for his work in Distributed Software Engineering. He is co-author of a recent book on Concurrency, co-author of a previous book on Distributed Systems and Computer Networks, and the author of over 200 journal and conference publications. He is a Chartered Engineer, Fellow of the IET, Fellow of the BCS and Fellow of the ACM and Fellow of the City and Guilds of London Institute.

Jeff MAGEE

Jeff MAGEE is Head of the Department of Computing at Imperial College London. His research is primarily concerned with the software engineering of distributed systems, including requirements, design methods, analysis techniques, operating systems, languages and program support environments for these systems. He is co-author of a recent book on concurrent programming entitled "Concurrency—State models and Java programs" and the author of too many journal and conference publications. He was co-editor of the IEE Proceedings on Software Engineering and is currently a TOSEM Associate Editor. He is the co-recipient of the 2005 ACM SIGSOFT Outstanding Research Award for his work in Distributed Software Engineering. He is a Chartered Engineer, Member of the IET and Fellow of the BCS.