



**HAL**  
open science

# Uniform and Model-Driven Engineering of Feedback Control Systems

Filip Křikava, Philippe Collet, Mireille Blay-Fornarino

► **To cite this version:**

Filip Křikava, Philippe Collet, Mireille Blay-Fornarino. Uniform and Model-Driven Engineering of Feedback Control Systems. Proceedings of the 8th IEEE/ACM International Conference on Automatic Computing, 2011, Karlsruhe, Germany. 10.1145/1998582.1998616 . hal-01117776

**HAL Id: hal-01117776**

**<https://inria.hal.science/hal-01117776v1>**

Submitted on 17 Feb 2015

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



ensure that the dataflow toward controllers is well typed and not interrupted.

The adaptation part is represented by controllers and effectors. Controllers orchestrate the bounded effectors (via *control link*) that carry out the actual system modification. There are many different kinds of controllers that can be used depending on the intended purpose [7, 5]. However, in our approach, we are not directly concerned by the actual design of the adaptation behavior itself, we focus on supporting the surrounding architecture as a whole.

The links themselves support adaptation so that their transmission properties, such as the notification or observing periods, or in case of remote links connection timeouts, can be advertised and used by other loops to make the system more robust.

### 3. CASE STUDY IN GRID MIDDLEWARE

Among others, our work is motivated by several use cases that aim at improving parts of the gLite<sup>1</sup> grid middleware operation by introducing self-adaptive capabilities. One of them is the overload control scenario of the Workload Management System (WMS), which is responsible for distributing and managing jobs across grid resources.

Fig. 2 exemplifies our proposal with a simplified version of the corresponding autonomic architecture. The main controller (1) objective is the optimization of the number of tasks ( $N$ ) in the WMS Task Queue by adjusting the job submission requests arrival rate ( $\lambda$ ), based on the resource usage observed from the system sensors ( $R$ ). The adjustment can be either by simply refusing the requests or, for aware clients, by imposing a delay before next submission. The second controller (2) coordinates the first one by setting the reference value ( $R^*$ ) for the resource usage based on the system administrator preference ( $\bar{R}$ ) and the system availability ( $S$ ). Shall these two controllers fail (incorrect model, system in an unanticipated state, etc.), the system may get overloaded and so to prevent this, the controller (3) is employed, running at lower pace, restarting the process when resource consumption is over a limit. The controller (4) is used to try to start the WMS in cases it is not running. In order not to conflict with the controller (3), it double checks the heartbeat with a delay.

The primary concern in this overload control scenario is to protect every unbounded resource usage, regardless whether it is a system resource or a service usage. This has to be done recursively since protecting one resource might in consequence jeopardize another and thus transforming one problem into a different one that could be potentially much worse. Following a “at least do no harm” principle, one can prevent the system being flooded with restart or start requests, for example by using an exponential back-off to control the minimum time between the requests. The controllers (2) and (4) are examples of controllers running at different time scales [7].

Besides the controllers, like (1) and (2) in our case, which usually have to be always designed specifically for each particular scenario, the other parts of the architecture are likely to be reused in other scenarios. This is an ongoing experimentation.

### 4. CONCLUSION & ONGOING WORK

We have presented an approach to support the architectural design of control loop elements and their connection links. Each of them being an adaptive element enables architects to uniformly explicit control patterns, from coordination between loops to adaptive monitoring. The presented model is complemented by a deployment model, not shown here, which deals with element locations and instantiation issues.

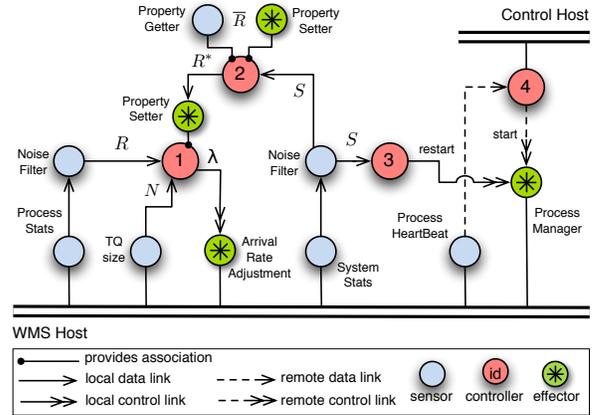


Figure 2: WMS Overload Control Architecture (excerpt)

A first runtime platform is based on an OSGi runtime with a DSL for model definition and code generation. Ongoing work deals with the distribution aspect of the system as well as with generating different compact representations in which a large number of model elements are merged at runtime. Additionally, SCA components and C++ code are targeted in two different grid middlewares, gLite and Condor<sup>2</sup> respectively. In the longer term, we expect the model to be the underlying basis for a library of loop elements and patterns.

**Acknowledgements.** The work reported in this paper is partly funded by the ANR SALTY project<sup>3</sup> under contract ANR-09-SEGI-012.

### 5. REFERENCES

- [1] O. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, and A. van Moorsel. The Self-star Vision. *Self-star Properties in Complex Information Systems*, pages 1–20, 2005.
- [2] L. Broto, D. Hagimont, E. Annoni, B. Combemale, and J.-P. Bahoun. Towards a model driven autonomic management system. In *Fifth International Conference on Information Technology: New Generations*, pages 63–69, 2008.
- [3] Y. Brun, G. Di Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. Engineering Self-Adaptive Systems Through Feedback Loops. *Software Engineering for Self-Adaptive Systems*, pages 48–70, 2009.
- [4] D. Garlan, B. Schmerl, and P. Steenkiste. Rainbow: architecture-based self-adaptation with reusable infrastructure. *International Conference on Autonomic Computing, 2004. Proceedings.*, pages 276–277, 2004.
- [5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback Control of Computing Systems*. John Wiley, 2004.
- [6] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. *Computer*, 36:41–50, 2003.
- [7] M. Litoiu, M. Woodside, and T. Zheng. Hierarchical model-based autonomic control of software systems. *Proceedings of the 2005 workshop on Design and evolution of autonomic application software - DEAS '05*, page 1, 2005.
- [8] M. Salehie and L. Tahvildari. Self-adaptive software. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):1–42, May 2009.

<sup>2</sup><http://www.cs.wisc.edu/condor>

<sup>3</sup><https://salty.unice.fr>

<sup>1</sup><http://glite.web.cern.ch/glite/>