Engineering Self-Coordinating Software Intensive Systems

Wilhelm Schäfer[1,2], Mauro Birattari[4], Johannes Blömer[2], Marco Dorigo[2,4], Gregor Engels[2], Rehan O'Grady[4], Marco Platzner[2], Franz Rammig[1,2], Wolfgang Reif[5], and Ansgar Trächtler[1,3]

wilhelm@upb.de

[1] Heinz Nixdorf Institute
University of Paderborn
Fürstenallee 11, Pohlweg 98,
and Warburger Str. 100
Paderborn, Germany

[2] Department of Computer Science
University of Paderborn
Warburger Str. 100
Paderborn, Germany

[3] Department of Control Engineering and Mechatronics
University of Paderborn
Pohlweg 98
Paderborn, Germany

[4] IRIDIA, CoDE
Université Libre de Bruxelles
Ave. F. Roosevelt 50
CP 194/6, 1050 Brussels,
Belgium

[5] Department of Software Engineering and Programming Languages
University of Augsburg
Universitätsstr. 6a
Augsburg, Germany

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: General

General Terms

Algorithms, Design, Languages, Security, Theory, Verification

Keywords

New Engineering Methodology, Systems Engineering, Mechatronic Systems, RailCab

In a globalized world, technical systems are becoming ubiquitous, and are increasingly interacting directly with the physical world. The design space of such systems also, therefore, includes the basic laws of physics. Examples of such systems are worldwide production systems, public transport systems and the Internet as a commercial platform (including shipping the real products of course).

In the future, these systems will become so complex and distributed that current development techniques will not suffice to produce safe and secure systems. This is due to a number of characteristics which are listed in the following: [18]:

- Volatility: complex volatile networks in which components cooperate as well as possibly compete,
- Decentralization: components act autonomously because no central control is feasible and a single point of failure is also not tolerable anymore,
- Scoped knowledge: an unobservable global system state and thus components with only local knowledge, optimization of own benefits being the driving force of a component's cooperation,
- Difficulty of guaranteeing behaviour: functional correctness, security and safety become difficult to guarantee when resources are limited and parts can fail.

In order to address these characteristics and to build systems which still operate safely an securely, components must be able to learn from environmental changes as a universal ability. Further, a tight integration of continuous and discrete control is needed, i.e. (software) control is becoming a combination of classical feedback controllers defined by differential equations to address physical constraints and state based transitions defined e.g. by statecharts to address the need to compute strategic decisions and to take into account environmental changes. These systems have to exhibit self-X properties like self healing, self-configuring, self-adapting, and self-optimizing leading to robustness against failures, disturbances and changing requirements. (Sometimes, a subset of such systems is also called emergent system which in our view only means that a large number of components follow relatively simple local rules and interaction patterns. As an ensemble, however, they exhibit complex system behaviour with ensemble capabilities that go beyond the single componentsŠ capabilities.) In any case it means deferring decisions from design-time to run-time. We refer to such systems as *self-coordinating* systems, by which we mean technical systems that self-organise to achieve specific goals.

As examples for such software intensive systems consider that in the future, logistics will be responsible for supporting the assembling of goods tailored to customers' individual specifications. Driver assistance systems will need to enable cars to cooperate closely with each other to ensure optimal safety and comfort for their occupants. And software will need to be constructed at runtime from millions of pre-fabricated components in a global software market and deployed dynamically on demand. Logistics must cope with volatile networks of autonomously acting carriers and suppliers, competing as well as cooperating with each other. Driver

FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA.

assistance systems form heterogeneous networks of cars, where each car aquires only local knowledge. A global software market forms a huge network of existing components whose assembly into new software requires a high degree of adaptation. Software component requests will need to be bound to corresponding components providing the required functionality and then allocated to executing devices. In each of these settings, the global system state is neither observable nor would a knowledge of it (due to its complexity) be of any help. New properties emerge while the network's components adapt to and learn from other components. Still, the requirements on safety and security are very strong and need to be certified in the end.

Currently, the engineering of self-coordinating systems follows mainly classical approaches for centrally-coordinated systems, adapting and extending them to meet the new requirements of self-coordination. A radically new approach would span a number of disciplines, all of which need to align their methods to the paradigm of self-coordination, and only in their combination can we manage the challenges that self-coordination brings. The integration of mechanical engineering, particularly control engineering, with electrical engineering and software engineering is needed to combine the physical environmental constraints of a system with its state-based control and the (software) system architecture. Integration with knowledge about business-oriented applications is needed, because in a world wide global market strategic decisions must be taken, and the corresponding logic has to be embedded in the controlling software as well (c.f. Figure 1 [9]).

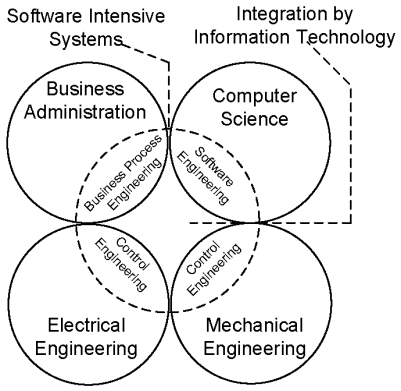

Figure 1: Software-intensive systems w.r.t classical disciplines

The challenges which the **engineering of self coordinating systems** faces, are characterized by a number of research questions like:

- How can optimal strategies be determined in the presence of partial or even unreliable information?
- How can heterogeneous components decide which information is relevant and which need not be considered?
- What algorithms might help us in reaching stable, robust, and desirable behavior in a distributed network?
- How can components find out about their coordination possibilities with cooperating or even competing components?
- What design and operation principles do we need for the underlying technical infrastructure and how can we maintain these principles when resources are restricted and parts can fail?
- How can we manage the secure exchange of information without a central public-key infrastructure and only limited resources?
- When do we need cross-border migrations between hardware and software?
- What modeling formalisms have to be supplied in order to enable the construction of adaptable systems?
- What analysis techniques are required to make software reflect on both its own and its environment's behaviour and consequently change itself?
- What verification techniques can ensure the correctness of self-coordinating systems despite their inherent potential volatility?
- How can a tight integration between classical feedback controllers and the state-based discrete control be achieved?

Today's software engineering research by and large is not really focussing on answering these questions but is rather centered around much smaller systems than the ones mentioned above which, in addition, often consist of software "only" (e.g. information systems). It is also usually done by small, non-interdisciplinary research teams. Of course, that does not mean that we no longer need research on testing and analysis, on program understanding, or on formal modelling and verification approaches, to list only a few examples. However, this research must be combined with e.g. the development and layout of complex networks and their underlying infrastructure as well as research on corresponding data structures and algorithms and control theory (e.g. [6]). Even in that paper, integration with control theory which means integration with the laws of physics, is not really addressed.

In contrast, to address some of the issues, a systems engineering community[1] has been formed. However, looking at their publications and major conference(s), software engineering in turn as well as control theory only play a very minor role in this effort[2].

There are approaches in the service-oriented architecture (SOA) community and self-adaptive community which take highly dynamic and adaptive applications into account (e.g. [16, 5, 7]). Interdisciplinary engineering with for example control engineers is not relevant for this kind of applications or is considered only on a high abstract level [13].

As indicated, the integration of a number of research strands from various fields is needed to master the complexity of self-coordinating systems. As examples consider the combination of swarm intelligence and game theory to address the contradiction between selfishness of agents [1, 2] on the one hand and their necessary cooperation on the other hand with work on safety and security to address the usual critical mission of these systems for the whole society [20]. The latter part becomes particularly complicated, because current usable techniques usually assume a finite state space and if not, approaches usually do not scale [19]. Furthermore, control theory and its current sophisticated simulation environments have to be included and extended to get an understanding of a system's behaviour, even if the inherent complexity of such systems mean that a full understanding can never be achieved before the system goes into operation [4]. Current simulation techniques are not sufficient, because they basically assume total knowledge of the

[1]INCOSE: The International Council on Systems Engineering (http://www.incose.org/)

[2]The main focus is on informal engineering and management aspects (e.g. http://www.incose.org/symp2009 or http://www.incose.org/practice/techactivities/wg/transport/).

system environment and the system behaviour itself, i.e. all possible states. Smart integration of simulation and formal system verification taking into account uncertainty about the environment and the system itself as well as adaptivity of the behaviour at runtime is another major research issue.

A highly interdisciplinary team of researchers at the University of Paderborn works already on some of the issues above. The team includes members from the departments of informatics, electrical engineering, mechanical engineering and economy[3].

The underlying paradigm of a number of common research projects is to view modern machines as built by mechanical engineers as agents and consider an (advanced) mechanical product as a system of cooperating agents. This paradigm works for as diverse applications as modern public and private transport systems, production systems but also the so-called smart grids which are supposed to distribute and use the energy produced by a lot various resources (coal, nuclear, solar, wind etc.), much more effectively than today.

Most notably and serving as a reference example, a team in Paderborn works on a modern public transport system called RailCab which exhibits many of the features of a self coordinating mechanical system. This system has reached a prototype status which includes a real physically built test track on the campus[4].

The vision of the RailCab project is to provide the comfort of individual traffic concerning scheduling and on-demand availability of transportation as well as individually equipped cars on the one hand and the cost and resource effectiveness of public transport on the other hand (e.g. [14]). The modular railway system combines sophisticated undercarriages with the advantages of new actuation techniques to increase passenger comfort while still enabling high speed transportation and (re)using the existing railway tracks. One important feature is the reduction of energy consumption due to air resistance by coordinating the autonomously operating shuttles in such a way that they build convoys whenever possible. Such convoys are built on-demand and shuttles travel close to each other (less than 1m) such that a high reduction of energy consumption is achieved. Consequently, shuttles as part of a network of possibly more than 100 000 shuttles need to decide when to build or not to build convoys. In addition, the construction of a convoy is a very critical situation which requires tight interaction between control engineering (developing the speed control units) and software engineering (developing the communication protocols) as well as the underlying communication infrastructure (reliable transmission of messages on a wireless network) [10, 8, 11, 12].

Tool support for modeling and checking such system specifications has been implemented as part of the FUJABA real-time tool suite[5] [17]. In order to support the specification of controllers by block diagrams and differential equations a commercially available tool, namely CAMeL-View[6], has been integrated into the FUJABA tool suite. Both tools offer code generators which target various platforms [3].

Fig. 2 shows a screenshot of the simulation environment of the tool. It is an extension of the CAMeL-View simulation. The three dimensional view in the lower part simulates a convoy of two shuttles. State information can now be displayed in addition to the controller status information of all shuttles (lower right part). The graphs in the upper part of the screen display "traditional" controller information. In detail, the upper right part displays the position of the shuttles, the upper left hand the velocity of the first and second shuttle, and in the lower left the reference values for speed are shown.

The authors are engaged in ongoing efforts to advance this rapidly growing field. Paderborn's existing expertise will form the backbone of the new research effort. The provincial government and the town of Paderborn have decided to place a new dedicated Science and Technology park around the "Fürstenallee Campus" of the university called "Zukunftsmeile Fürstenallee"[7]. With support from the European Union and some global players like Wincor Nixdorf, Siemens and Miele joint research labs between industry and the university are being funded and will be located here. Research areas include software engineering, mechatronics and visualisation and simulation. The proximity should establish a fruitful exchange of ideas and approaches across the different disciplines.

[3]E.g. the Collaborative Research Centre 614 "Self-optimizing concepts and structures in mechanical engineering" (CRC 614 - http://www.sfb614.de/en/sfb614/) or [15]

[4]RailCab project (http://nbp-www.upb.de/index.php?id=2&L=1)

[5]http://www.fujaba.de/projects/real-time.html

[6]http://www.ixtronics.com

[7]http://www.zukunftsmeile-fuerstenallee.de

1. REFERENCES

[1] *Advances in Artificial Life, Proceedings of the 9th European Conference on Artificial Live, ECAL 2007, Lisbon, Portugal*, volume 4648 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[3] S. Burmester, H. Giese, S. Henkler, M. Hirsch, M. Tichy, A. Gambuzza, E. Much, and H. Vöcking. Tool support for developing advanced mechatronic systems: Integrating the fujaba real-time tool suite with camel-view. In *Proc. of the 29th International Conference on Software Engineering (ICSE), Minneapolis, Minnesota, USA*, pages 801–804. IEEE Computer Society Press, May 2007.

[4] S. Burmester, H. Giese, S. Henkler, M. Hirsch, M. Tichy, A. Gambuzza, E. Münch, and H. Vöcking. Tool support for developing advanced mechatronic systems: Integrating the fujaba real-time tool suite with camel-view. In *Proc. of the 29th International Conference on Software Engineering (ICSE), Minneapolis, Minnesota, USA*, pages 801–804. IEEE Computer Society Press, May 2007.

[5] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, editors. *Software Engineering for Self-Adaptive Systems [outcome of a Dagstuhl Seminar]*, volume 5525 of *Lecture Notes in Computer Science*. Springer, 2009.

[6] P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan, and K. Wallnau. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, Carnegie Mellon, June 2006.

[7] I. Georgiadis, J. Magee, and J. Kramer. Self-organising software architectures for distributed systems. In *WOSS '02: Proceedings of the first workshop on Self-healing systems*, pages 33–38, New York, NY, USA, 2002. ACM.

[8] H. Giese, S. Burmester, W. Schäfer, and O. Oberschelp. Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration. In *Proceedings of 12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004), Newport Beach, USA*, pages 179–188. ACM Press, November 2004.

[9] H. Giese, S. Henkler, M. Hirsch, V. Roubin, and M. Tichy. Modeling techniques for software-intensive systems. In

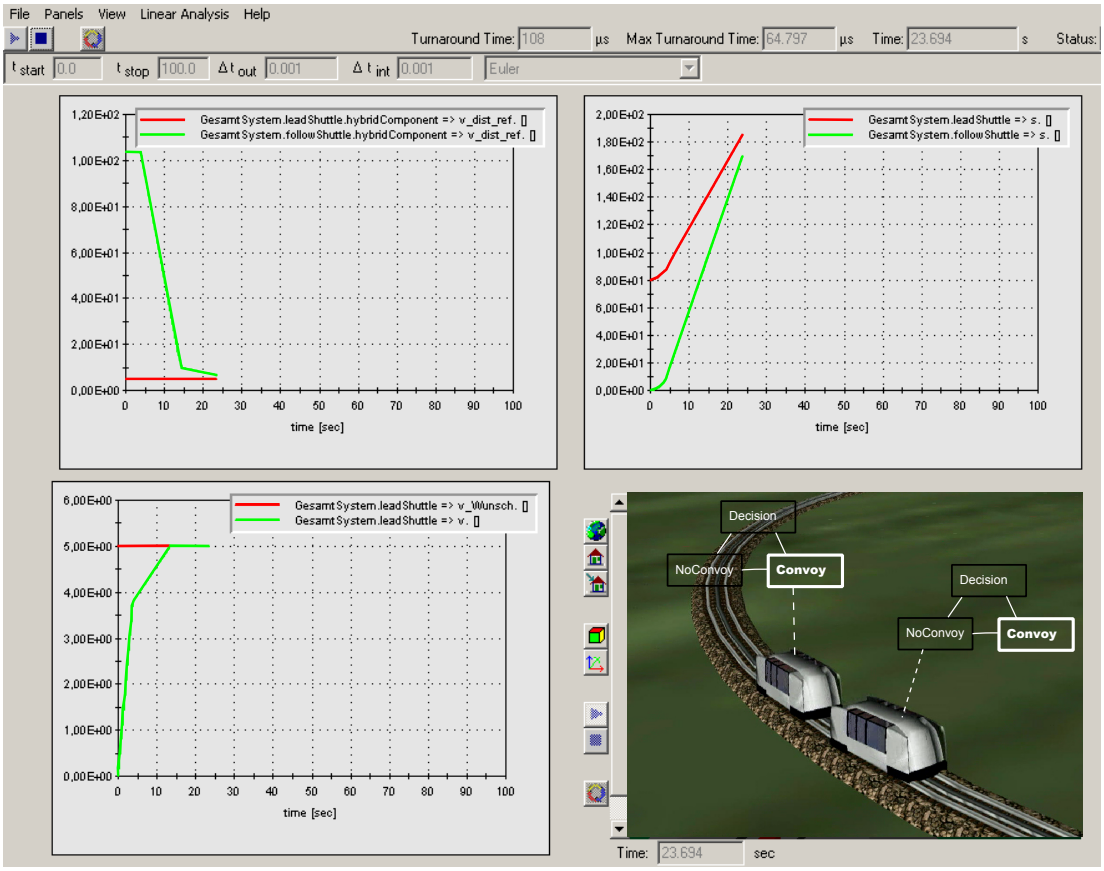

Figure 2: Simulation Environment

Designing Software-Intensive Systems: Methods and Principles. Langston University, OK, 2008.

[10] H. Giese, M. Tichy, S. Burmester, W. Schäfer, and S. Flake. Towards the Compositional Verification of Real-Time UML Designs. In *Proceedings of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-11)*, pages 38–47. ACM Press, September 2003.

[11] C. Henke, M. Tichy, T. Schneider, J. Böcker, and W. Schäfer. Organization and control of autonomous railway convoys. In *Proceedings of the 9th International Symposium on Advanced Vehicle Control, Kobe, Japan*, 2008.

[12] S. Henkler, M. Hirsch, C. Priesterjahn, and W. Schäfer. Modeling and verifying dynamic communication structures based on graph transformations. In G. Engels, M. Luckey, and W. Schäfer, editors, *Software Engineering 2010 - Fachtagung des GI-Fachbereichs Softwaretechnik, 22.-26.2.2010 in Paderborn*, volume 159 of *LNI*, pages 153–164. GI, 2010.

[13] J. Kramer and J. Magee. Self-managed systems: an architectural challenge. In L. C. Briand and A. L. Wolf, editors, *Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, pages 259–268, 2007.

[14] E. Münch, P. Adelt, M. Krüger, B. Kleinjohann, and A. Trächtler. Hybrid planning and hierarchical optimization of mechatronic systems. In *Proceedings of the International Conference on Control, Automation and Systems (ICCAS)*, Seoul, Korea, 14 - 17 Oct. 2008.

[15] F. Nafz, F. Ortmeier, H. Seebach, J.-P. Steghöfer, and W. Reif. A universal self-organization mechanism for role-based organic computing systems. In *ATC '09: Proceedings of the 6th International Conference on Autonomic and Trusted Computing*, pages 17–31, Berlin, Heidelberg, 2009. Springer-Verlag.

[16] E. D. Nitto, C. Ghezzi, A. Metzger, M. P. Papazoglou, and K. Pohl. A journey to highly dynamic, self-adaptive service-based applications. *Autom. Softw. Eng.*, 15(3-4):313–341, 2008.

[17] C. Priesterjahn, M. Tichy, S. Henkler, M. Hirsch, and W. Schäfer. Fujaba4eclipse Real-Time Tool Suite. In *Model-Based Engineering of Embedded Real-Time Systems (MBEERTS)*, LNCS, pages 1–7. Springer, 2009. to appear.

[18] F. J. Rammig. Towards self-coordinating ubiquitous computing environments. In *Embedded and Ubiquitous Computing*, volume 4096 of *Lecture Notes in Computer Science*, pages 2–13. Springer Berlin / Heidelberg, 2006.

[19] W. Schäfer and H. Wehrheim. The challenges of building advanced mechatronic systems. In *Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, pages 72–84. IEEE Computer Society, 2007.

[20] J. Sifakis. Safety, security and quality. *ACM Comput. Surv.*, page 124, 1996.