Advances in model driven software engineering

The CNS is obviously very good at controlling biological mechanical systems. If von Neumann was right, and the brain essentially is a kind of automaton, we might be able to extract useful knowledge of the brain's motor control algorithms, including the properties of synergies. This is a central theme of the THE project. SICS' role is to try to understand and model enough of the function of the mammalian CNS that it can be applied to adaptive control of robot limbs.

Starting with ion channels

In contrast with other attempts - which typically proceed top-down from a cognitive behavioural level - our approach is to start on the lowest, ion channel level in the structural hierarchy of the CNS. From there, we derive the function of neuronal microcircuits involved in motor control. A microcircuit is a combination of a small number of neurons operating together; it can be seen as the next level above neurons in the hierarchy. These circuits engage motor centra, the spinal cord, the cerebellum, and the pre-cerebellar structures in a complex pattern of feedback loops (Figure 1).

A serious challenge is that the macroscopic operation of microcircuits depends critically on neuronal membrane molecular dynamics, which is directly affected by thermodynamic noise. Although noise in an electronic circuit is normally undesirable, we can see that evolution has taken advantage of this in order to design robustness into circuit operation. However, the crucial function of noise in microcircuits requires non-trivial mathematical treatment, and stochastic models of neuronal activity are among the most advanced applications of the theory of stochastic processes in biology. Nevertheless, our preliminary results offer some surprises: for instance, it seems that much of the fundamental processing in the CNS is in fact — linear!

Project partners

SICS cooperates closely with a group of neurophysiologists led by Dr Henrik Jörntell at the Department of Experimental Medical Science at Lund University, Sweden. This group performs sophisticated electrophysiological experiments, where physical connectivity and signal transmission between neurons are recorded in vivo.

Other partners in the project are Centro "E. Piaggio", University of Pisa, Italy; Deutsches Zentrum für Luft und Raumfahrt (DLR), Munich; National Technical University Athens, Greece; University of Siena, Italy; Utrecht University, The Netherlands; Université Pierre et Marie Curie, Paris, France; Universität Bielefeld, Germany; and Arizona State University, USA.

Link:
http://www.thehandembodied.eu

Reference:
[1] M. Santello et al.: "Patterns of Hand Motion during Grasping and the Influence of Sensory Guidance", J Neurosci 22, 1426-1435, 2002

Please contact:
Martin Nilsson, SICS, Sweden
E-mail: mn@sics.se

Advances in Model Driven Software Engineering

by Mark G.J. van den Brand and Jan Friso Groote

Empirical evidence shows that the use of model driven software engineering can result in an up to 10-fold quality improvement and decreased development time. Researchers from Eindhoven University of Technology tested this on a detector control system at CERN. This brings Turing's vision of software another step closer.

Well before the advent of modern computers, Turing anticipated the complexities of computing software. Dijkstra spent his latter life developing methods to simplify software. by mathematically deriving correct algorithms. Hoare and Milner worked on formalisms to model and understand the essence of behaviour long before concrete programs existed. Over time, the thinking on software became more abstract.

Our work is reversing this approach towards abstraction by using models as the primary step in the construction of software. First, it is shown mathematically that the models perform their intended functionality and never perform undesired and dangerous behaviour. Subsequently, these models generate software.

Careful comparison of projects that use a more classic approach compared to those that use models show an up to 10-fold reduction in bug-reports during development and an up to three-fold reduction in development times. These figures stem from the medical domain [1]. More telling are the responses by test engineers:"What have you done? Normally we find bugs in minutes. Now we find none." In reality, not all bugs are removed by model driven software engineering. But typically, the deep and intricate errors are removed and the shallow problems persist (e.g. reformulating a message for the user).

The biggest challenge of model driven software engineering is the state space explosion problem. The size of software can be described in terms of the number of states it can reach. This number is so forbiddingly large that a new name is invented to indicate the class of numbers: computer engineering numbers. Typically, the smallest such numbers are 10^{1000} for a small controller through to numbers not concisely expressible with a single exponential. For comparison the largest astronomical number is 10 to the power 100.

Although models have fewer states than actual software, faster algorithms, huge computers and in particular "symbolic methods" are becoming increasingly effective in establishing the correctness of huge models. For instance, at CERN in Genève the control system of one of the detectors is modelled by approximately 20,000 interacting finite state machines from which the actual control software is generated. The model and the software suffered from a persistent liveness problem, where only part of the detector would be initialized properly. By employing symbolic methods we managed to detect and remove all such liveness problems [2].

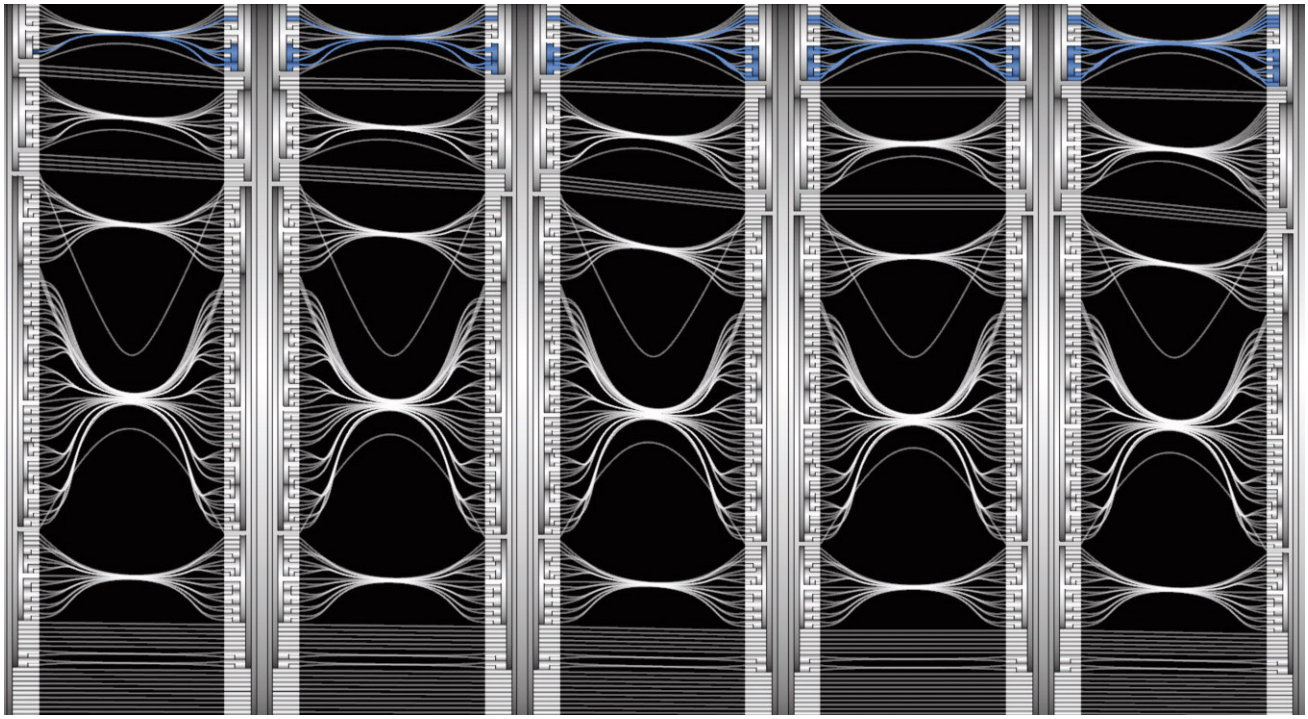

Figure 1: Model driven software improves both quality and development time of new software. During the process, visualizations of model transformation dependencies are used for debugging and other purposes.

Due to the success of the applicability of model driven software engineering, a fascinating new scientific question is emerging: which kind of software modelling avoids the state space explosion problem? We provide initial answers, among which the most interesting is a preference for information polling instead of information pushing. We already see signs that these modelling guidelines are being transformed into the way actual products operate.

Model driven software engineering not only increases the quality but also the effectiveness in software development. Models are used as artifacts from which the executable code is being generated. The executable code is obtained via model transformations. The target code can be considered as a model with a lot of low-level details.

This way of working means that the overall quality of the resulting software is based on the model and the model transformations. The quality – both internal and external - of the model transformations becomes more important.The internal quality, related to understandability, maintainability and reusability, can be established via analysis of the model transformations. This can be done via metrics or visualization of dependencies between models, meta-models, and model transformations (see Figure 1). The external quality, related to correctness, is hard to determine. In order to ensure correctness of model transformations it is necessary to establish the semantics of both the input and output language and proof obligations have to be derived [3].

The research effort in the design of domain specific languages, one of the important artifacts of model driven software engineering, shifts from a syntax towards static and dynamic semantics. The research on proving model transformations correct is relatively new and unexplored but will be crucial in order to ensure the overall quality of software.

In summary,model driven software engineering is becoming increasingly effective to the point that it will soon be generally adopted.

Link:
http://www.mcrl2.org

References:

[1] J.F. Groote, A.A.H. Osaiweran, J.H. Wesselius: “Analyzing the effects of formal methods on the development of industrial control software”, in Proc. of the IEEE ICSM 2011, Williamsburg, VA, USA, September 25-30, pp. 467-472, 2011

[2] Yi Ling Hwong, et al.: “An Analysis of the Control Hierarchy Modelling of the CMS Detector Control System”, in Journal of Physics: Conference Series, 331(2), 2011

[3] S. Andova et al.: “Reusable and correct endogenous model transformations”, in Proc. “Theory and Practice of Model Transformations”, Z. Hu & J. de Lara (Eds.), ICMT 2012. Springer LNCS, vol. 7307, pp. 72-88, 2012

Please contact:

Jan Friso Groote, Eindhoven University of Technology
E-mail: J.F.Groote@tue.nl
http://www.win.tue.nl/~jfg

Mark van den Brand, Eindhoven University of Technology
E-mail: M.G.J.v.d.Brand@tue.nl
http://www.win.tue.nl/~mvdbrand