Modelling and Verification of Robotic Platforms for Simulation using RoboStar Technology

Ana Cavalcanti[1]

Department of Computer Science, University of York,
York, YO105GH, UK,
`Ana.Cavalcanti@york.ac.uk`

The RoboStar framework[1] supports model-based engineering of robotic applications. Modelling is carried out using diagrammatic domain-specific languages: RoboChart [13] and RoboSim [3]. Verification and generation of artefacts is justified by a formal semantics given using a state-rich hybrid version of a process algebra for refinement [7]. It is inspired by CSP [19] and cast in Hoare and He's Unifying Theories of Programming (UTP)[10] formalised in Isabelle [6].

RoboChart is an event-based language for design, while RoboSim is a cycle-based language for simulation. Tool support is provided by RoboTool, which includes facilities for graphical modelling, validation, and automatic generation of CSP (for analysis with the model checker FDR [9]) and PRISM [11] scripts (for verification of probabilistic controllers), and simulations. RoboChart and RoboSim are based on the use of state machines to specify behaviour, akin to notations already in widespread use [5, 16, 2, 20], but RoboChart and RoboSim are enriched with facilities for verification and traceability of artefacts.

Recent work has focussed on enriching RoboSim for physical modelling. Current practice in robotics often uses simulation to understand the behaviour of a robotic controller for a particular robotic platform and environment. A wide variety of simulators for robotics use different tool-dependent or even proprietary programming languages and API [18, 14, 8, 12, 17]. Physical modelling of the platforms are encoded by programs in customised notations, generated from graphical tools, or in C++, Java, Python, or C#, for example.

RoboSim, on the other hand, is a tool-independent notation. For physical modelling, we have defined a notation based on SysML block diagrams [15]. Our profile is inspired by XML-based notations used by robotics simulators[2]. It defines a physical model by a diagram that captures the physical components of a platform as links (rigid bodies), joints, sensors, and actuators. Properties of these blocks capture their attributes that are relevant for simulation and for capturing behaviour: movement and use of sensors and actuators.

In contrast with XML-based notations in current use, RoboSim *block diagrams* encourage readability and support modularisation via several mechanisms. Models can be parametrised by constants that represent, for example, key measures of physical bodies. The pose of an element is defined always in reference to

[1] `www.cs.york.ac.uk/robostar/`

[2] `sdformat.org`

the element that contains it. A richer notion of connection captures flexible and fixed compositions. A library fosters reuse by the possibility of defining parts and fragments that can be instantiated or simply included to define a complete model. Finally, well-formedness rules ensure validity of models.

The most distinctive feature of RoboSim block diagrams, however, is the possibility of defining systems of differential algebraic equations that capture behaviour of the platform. For sensors, these equations define how inputs (from the environment) are reflected in sensor outputs for use with the software. For actuators, the equations define how inputs from the software affect the outputs of the actuators, and therefore, affect the platform itself (in the case of motors, for example), or the environment. For joints, the equations define how their movement induces movement on the links connected to them.

A system view is provided by connecting a RoboSim block diagram that specifies a physical model for a robotic platform, to a RoboSim module that specifies a control software. This is achieved by a *platform mapping*, which specifies how software elements that abstract services of the platform are defined. In specifying these services, we can use outputs of sensors and inputs of actuators.

Ongoing work, provides support to translate RoboSim block diagrams to XML for use in simulation (using Coppelia, formerly, v-rep). For mathematical modelling, the UTP semantics constructs a hybrid model, with constructs inspired by those of *Circus* [4], combining Z [1, 21] and CSP.

Acknowledgements The work mentioned is a collaboration with colleagues at the RoboStar group, in particular, Alvaro Miyazawa and Sharar Ahmadi. The author's work is funded by the Royal Academy of Engineering grant CiET1718/45, and UK EPSRC grants EP/M025756/1 and EP/R025479/1. No new primary data was created as part of the study reported here.

References

1. ISO/IEC 13568:2002. Information technology - Z formal specification notation - syntax, type system and semantics. International Standard.
2. S. G. Brunner, F. Steinmetz, R. Belder, and A. Domel. Rafcon: A graphical tool for engineering complex, robotic tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3283–3290, 2016.
3. A. L. C. Cavalcanti, A. C. A. Sampaio, A. Miyazawa, P. Ribeiro, M. Conserva Filho, A. Didier, W. Li, and J. Timmis. Verified simulation for robotics. *Science of Computer Programming*, 174:1–37, 2019.
4. A. L. C. Cavalcanti, A. C. A. Sampaio, and J. C. P. Woodcock. A Refinement Strategy for *Circus*. *Formal Aspects of Computing*, 15(2 - 3):146–181, 2003.
5. S. Dhouib, S. Kchir, S. Stinckwich, T. Ziadi, and M. Ziane. *Simulation, Modeling, and Programming for Autonomous Robots*, chapter RobotML, a Domain-Specific Language to Design, Simulate and Deploy Robotic Applications, pages 149–160. Springer, 2012.
6. S. Foster, J. Baxter, A. L. C. Cavalcanti, A. Miyazawa, and J. C. P. Woodcock. Automating Verification of State Machines with Reactive Designs and Isabelle/UTP.

In K. Bae and P. C. Ölveczky, editors, *Formal Aspects of Component Software*, pages 137–155, Cham, 2018. Springer.
7. S. Foster, A. L. C. Cavalcanti, S. Canham, J. C. P. Woodcock, and F. Zeyda. Unifying theories of reactive design contracts. *Theoretical Computer Science*, 802:105 – 140, 2020.
8. B. Gerkey, R. T. Vaughan, and H. Andrew. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *11th International Conference on Advanced Robotics*, pages 317–323, 2003.
9. T. Gibson-Robinson, P. Armstrong, A. Boulgakov, and A. W. Roscoe. FDR3 - A Modern Refinement Checker for CSP. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 187–201, 2014.
10. C. A. R. Hoare and He Jifeng. *Unifying Theories of Programming.* Prentice-Hall, 1998.
11. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.
12. S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.
13. A. Miyazawa, P. Ribeiro, W. Li, A. L. C. Cavalcanti, J. Timmis, and J. C. P. Woodcock. RoboChart: modelling and verification of the functional behaviour of robotic applications. *Software & Systems Modeling*, 18(5):3097–3149, 2019.
14. M. Olivier. Webots^{TM}: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.
15. OMG. OMG Systems Modeling Language (OMG SysML), Version 1.3, 2012.
16. I. Pembeci, H. Nilsson, and G. Hager. Functional reactive robotics: An exercise in principled integration of domain-specific languages. In *4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pages 168–179. ACM, 2002.
17. C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.
18. E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pages 1321–1326. IEEE, 2013.
19. A. W. Roscoe. *Understanding Concurrent Systems.* Texts in Computer Science. Springer, 2011.
20. M. Wachter, S. Ottenhaus, M. Krohnert, , N. Vahrenkamp, and T. Asfour. The ArmarX Statechart Concept: Graphical Programing of Robot Behavior. *Frontiers in Robotics and AI*, 3:33, 2016.
21. J. C. P. Woodcock and J. Davies. *Using Z - Specification, Refinement, and Proof.* Prentice-Hall, 1996.