



# Modelling and Verification of Robotic Platforms for Simulation Using RoboStar Technology

Ana Cavalcanti<sup>(✉)</sup>

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

The RoboStar framework<sup>1</sup> 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 [2, 5, 16, 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 [8, 12, 14, 17, 18]. 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<sup>2</sup>. 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

<sup>1</sup> [www.cs.york.ac.uk/robostar/](http://www.cs.york.ac.uk/robostar/).

<sup>2</sup> [sdformat.org](http://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. Brunner, S.G., Steinmetz, F., Belder, R., Domel, A.: Rafcon: a graphical tool for engineering complex, robotic tasks. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3283–3290 (2016)
3. Cavalcanti, A.L.C., et al.: Verified simulation for robotics. *Sci. Comput. Program.* **174**, 1–37 (2019)
4. Cavalcanti, A.L.C., Sampaio, A.C.A., Woodcock, J.C.P.: A refinement strategy for Circus. *Formal Aspects Comput.* **15**(2–3), 146–181 (2003)
5. Dhoubi, S., Kchir, S., Stinckwich, S., Ziadi, T., Ziane, M.: RobotML, a domain-specific language to design, simulate and deploy robotic applications. In: Noda, I., Ando, N., Brugali, D., Kuffner, J.J. (eds.) SIMPAR 2012. LNCS (LNAI), vol. 7628, pp. 149–160. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34327-8\\_16](https://doi.org/10.1007/978-3-642-34327-8_16)
6. Foster, S., Baxter, J., Cavalcanti, A., Miyazawa, A., Woodcock, J.: Automating verification of state machines with reactive designs and Isabelle/UTP. In: Bae, K., Ölveczky, P.C. (eds.) FACS 2018. LNCS, vol. 11222, pp. 137–155. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-02146-7\\_7](https://doi.org/10.1007/978-3-030-02146-7_7)

7. Foster, S., Cavalcanti, A.L.C., Canham, S., Woodcock, J.C.P., Zeyda, F.: Unifying theories of reactive design contracts. *Theoret. Comput. Sci.* **802**, 105–140 (2020)
8. Gerkey, B., Vaughan, R.T., Andrew, H.: The player/stage project: tools for multi-robot and distributed sensor systems. In: 11th International Conference on Advanced Robotics, pp. 317–323 (2003)
9. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.W.: FDR3—a modern refinement checker for CSP. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 187–201. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54862-8\\_13](https://doi.org/10.1007/978-3-642-54862-8_13)
10. Hoare, C.A.R., Jifeng, H.: *Unifying Theories of Programming*. Prentice-Hall, Upper Saddle River (1998)
11. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic symbolic model checking with PRISM: a hybrid approach. *Int. J. Softw. Tools Technol. Transf.* **6**(2), 128–142 (2004). <https://doi.org/10.1007/s10009-004-0140-2>
12. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: Mason: a multiagent simulation environment. *Simulation* **81**(7), 517–527 (2005)
13. Miyazawa, A., Ribeiro, P., Li, W., Cavalcanti, A., Timmis, J., Woodcock, J.: RoboChart: modelling and verification of the functional behaviour of robotic applications. *Softw. Syst. Modeling* **18**(5), 3097–3149 (2019). <https://doi.org/10.1007/s10270-018-00710-z>
14. Olivier, M.: Webots<sup>TM</sup>: professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **1**(1), 39–42 (2004)
15. OMG. *OMG Systems Modeling Language (OMG SysML), Version 1.3* (2012)
16. Pembeci, I., Nilsson, H., Hager, G.: 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, pp. 168–179. ACM (2002)
17. Pinciroli, C., et al.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intell.* **6**(4), 271–295 (2012)
18. Rohmer, E., Singh, S.P.N., Freese, M.: V-REP: a versatile and scalable robot simulation framework. In: *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, pp. 1321–1326. IEEE (2013)
19. Roscoe, A.W.: *Understanding Concurrent Systems*. Texts in Computer Science. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-1-84882-258-0>
20. Wachter, M., Ottenhaus, S., Krohnert, M., Vahrenkamp, N., Asfour, T.: The ArmarX statechart concept: graphical programming of robot behavior. *Front. Robot. AI* **3**, 33 (2016)
21. Woodcock, J.C.P., Davies, J.: *Using Z - Specification, Refinement, and Proof*. Prentice-Hall, Upper Saddle River (1996)