A Domain-Specific Language and Simulation Architecture for the Oncilla Robot*

Arne Nordmann[1] and Alexandre Tuleu[2] and Sebastian Wrede[1]

I. INTRODUCTION

We present an experimentation environment for the exploration of rich motor skills on a quadruped robot. We introduce a simulation platform around the (open) quadruped robot 'Oncilla' that is integrated with a domain-specific language (DSL) tool-chain for modeling and execution of reproducible movement experiments. The Webots-based simulator, shown in a Screenshot in Figure 1, features an open source and cross-platform (Linux and Mac OS) interface with taxonomic structure and rich sensor feedback. By providing a common abstracted interface for both simulation and hardware, it enables faster and easier transfer of experiments between these two domains. This interface is accessible through a local C++ interface as well as remotely via an open-source middleware in with C++, Java, Python, and Common Lisp bindings for extended language and tool support. Integrated with the common interface, a DSL-based environment around the simulator allows compact formulation of experiments in domain-specific terminology and eased generation of reproducible artifacts for experimentation. We present how movement architectures and dynamical-systems-based components of different partners of the AMARSi project are incorporated in the development of the DSL and the experimentation toolchain.

II. DSL APPROACH

Research on movement generation in robotics is not only challenging for reasons of the intrinsic complexity of the underlying control problems, but also due to the technological and conceptual fragmentation of the domain. Our answer to this challenge is a model-driven approach around a domain-specific language (DSL) environment [1], integrated with the quadruped simulator. We show our work based on DSLs to establish a domain-specific hypothesis test cycle for movement generation on robots.

The two core DSLs separate the domain along the formulation of functional and software architectural aspects. The AMARSi DSL [2] is designed to allow compact structural description of motor control architectures. It expresses motor control systems as combinations of so-called *movement primitives*, that model flexible and adaptive movements in terms of dynamical systems. It focuses solely on the functional concepts of a system, hiding the software architectural aspects. A second DSL covers the software architectural aspects, organizing the system in `Components` with `States`, `Ports`, and their wiring.

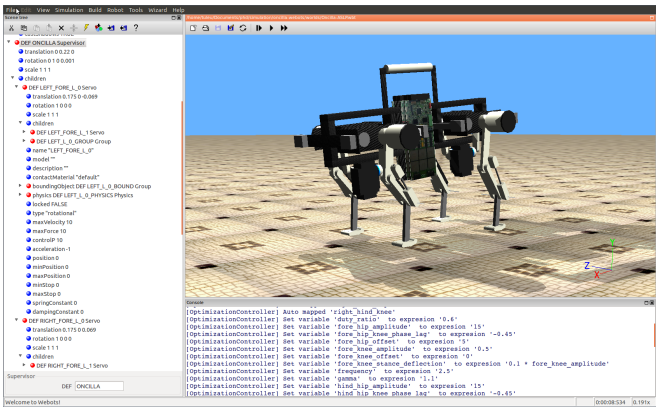

Fig. 1. Screenshot of the Oncilla simulator frontend in Webots.

As an example of how we foresee the DSL tool-chain to speed-up the hypotheses test cycle, we show how we allow formulation of dynamical system hypotheses directly in the DSL environment. Figure 2 shows an example of writing a dynamical system as formula in the projectional AMARSi DSL editor to be tested on the Oncilla platform. It supports writing the expression with references to the inputs and outputs of its context.

III. SOFTWARE ARCHITECTURE

The DSLs are prototyped and built in *JetBrains MPS* [3], a language workbench for building domain-specific languages and their projectional editors. Model to model transformation between the AMARSi DSL and the Component DSL maps the functional system model to software concepts defined in the Component DSL.

The Component DSL allows to automatically generate experimentation code, targeting the AMARSi software architecture, which we exemplify together with the integrated quadruped simulation platform. The two main software libraries of the software architecture are the Compliant Control Architecture (CCA), a robotics component framework, and the Robot Control Interface (RCI), a library providing software abstractions to model robot interfaces [4].

Figure 3 shows the common abstracted interface for both simulation and hardware. Its interface is locally accessible

*This work was partially funded by the European FP7 project AMARSi (grant no. 248311).

[1]A. Nordmann and S. Wrede are with the Research Institute for Cognition and Robotics, Bielefeld University, 33615 Bielefeld, Germany {anordman, swrede} at cor-lab.de

[2]A. Tuleu is with the BioRob Lab at EPFL, 1015 Lausanne, Switzerland alexandre.tuleu at epfl.ch

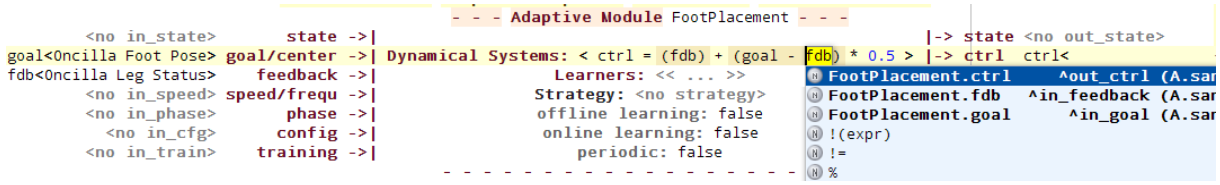

Fig. 2. `Adaptive Module` with its inputs and outputs for goal, feedback and control output. It contains a `Dynamical System` that is specified by an mathematical expression referencing inputs and outputs.

through an RCI-based C++ interface, using multiple inheritance to expose the node taxonomy. It is also remotely available through CCA and an open-source middle-ware with C++, Java, Python and Common Lisp bindings for extended language and tool support. One of the main design goals was to implement a common abstraction between hardware and simulation, with binary compatibility, to facilitate an easier transfer between hardware and simulation. The abstraction also allows to potentially exchange the simulation backend.

IV. SIMULATION ENVIRONMENT

The Oncilla robot is an open source, lightweight, bio-inspired quadruped robot developed by the AMARSi consortium. It can be seen as an improved version of the Cheetah-Cub robot [5], with one additional degree of freedom per leg (adduction / abduction), stronger brush-less motor, more sensors (passive joint position measurement, three axis load cells per leg and IMU), improved payload capacity and fastest communication loop (up to 500 Hz). From Cheetah-Cub, Oncilla retakes the three-segmented Advanced Spring Loaded Pantograph (ASLP) leg design, based on observation of leg kinematics in animal locomotion.

A Webots [6] model has been created, with an exact representation of the ASLP leg design including the two closed kinematic loop of each legs, serial and parallel compliance, and asymmetric actuation of the knee. These complex tasks have been performed with the use of our open source library *libwebots*. These library expose a series of reusable modules that provide an efficient API to access the Open Dynamic Engine primitives created by Webots. It allows to extend the Webots API by further features like closed kinematic loops, or create new complex functionality like the possibility to restraint the robot motion in its sagital plane. Although a precise description of the Oncilla quadruped robot could not be created, we motivate our choice of Webots as physic simulator in comparison to other open source or more accurate alternatives, for the simplicity of the Webots interface, targeting education [7]. The simple interface led to the application of the Oncilla simulator as teaching material for several master level courses.

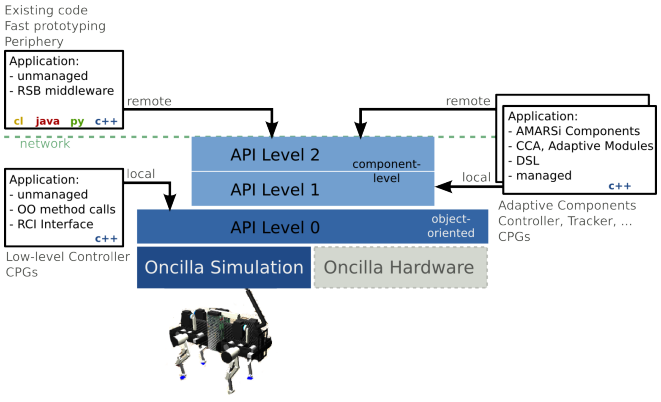

Fig. 3. Three-layered Oncilla Software Architecture.

V. CONCLUSION

Implementing the Oncilla interface to different simulator-backends and the Oncilla hardware still requires a lot of manual (implementation) work. Modeling functional and non-functional aspects of a platform would allow easier and faster integration of further backends (further simulators [8] and robot platforms) into the presented experimentation environment, by automatic generation of the required glue code.

The presented approach allows explicit modeling of motion control architectures and its quick validation in a simulation backend. Algorithmic and system parameters typically hidden in software artifacts are made explicit through the DSL approach. Thereby, reproducible experimentation is facilitated and Experimentation is sped up as long as developers stay within the modeled domain. That said, adapting the Oncilla interface to different simulator-backends and the Oncilla hardware still requires manual (implementation) work. Modeling functional and non-functional aspects of a platform would allow easier and faster integration of further backends (further simulators [8] and robot platforms) through automatic code generation. Future work at the DSL level will focus on improved representation of dynamic model aspects and a more powerful expression language to directly specify dynamical systems.

REFERENCES

[1] A. Nordmann and S. Wrede. A Domain-Specific Language for Rich Motor Skill Architectures. In *3rd International Workshop on Domain-Specific Languages and models for Robotic systems*, Tsukuba, 2012.

[2] A. Nordmann and S. Wrede. A Case-Study on a Domain-Specific Language for Robotics Motion Architectures (submitted). In *ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2013.

[3] JetBrains. Meta Programming System.

[4] A. Nordmann, M. Rolf, and S. Wrede. Software Abstractions for Simulation and Control of a Continuum Robot. In Itsuki Noda, Noriaki Ando, Davide Brugali, and James J. Kuffner, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 7628 of *Lecture Notes in Computer Science*, pages 113–124, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] A. Sproewitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. Ijspeert. Towards Dynamic Trot Gait Locomotion - Design, Control and Experiments with Cheetah-cub, a Compliant Quadruped Robot. *International Journal of Robotics Research*, submitted, 2013.

[6] Olivier Michel. Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[7] Luc Guyot, Nicolas Heiniger, Olivier Michel, and Fabien Rohrer. Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions.

[8] Michael Reckhaus, Nico Hochgeschwender, Jan Paulus, Azamat Shakhimardanov, and Gerhard K. Kraetzschmar. An Overview about Simulation and Emulation in Robotics. In *Proceedings of SIMPAR 2010 Workshops Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS*, pages 365–374, 2010.