Demo Abstract: A CPS Toolchain for Learning-based Systems

Charles Hartsell
Institute for Software Integrated Sys.
Vanderbilt University
charles.a.hartsell@vanderbilt.edu

Nagabhushan Mahadevan
Institute for Software Integrated Sys.
Vanderbilt University
nag.mahadevan@vanderbilt.edu

Shreyas Ramakrishna
Institute for Software Integrated Sys.
Vanderbilt University
shreyas.ramakrishna@vanderbilt.edu

Abhishek Dubey
Institute for Software Integrated Sys.
Vanderbilt University
abhishek.dubey@vanderbilt.edu

Theodore Bapty
Institute for Software Integrated Sys.
Vanderbilt University
theodore.a.bapty@vanderbilt.edu

Gabor Karsai
Institute for Software Integrated Sys.
Vanderbilt University
gabor.karsai@vanderbilt.edu

ABSTRACT

Cyber-Physical Systems (CPS) are expected to perform tasks with ever-increasing levels of autonomy, often in highly uncertain environments. Traditional design techniques based on domain knowledge and analytical models are often unable to cope with epistemic uncertainties present in these systems. This challenge, combined with recent advances in machine learning, has led to the emergence of Learning-Enabled Components (LECs) in CPS. However, very little tool support is available for design automation of these systems. In this demonstration, we introduce an integrated toolchain for the development of CPS with LECs with support for architectural modeling, data collection, system software deployment, and LEC training, evaluation, and verification. Additionally, the toolchain supports the modeling and analysis of safety cases – a critical part of the engineering process for mission and safety critical systems.

CCS CONCEPTS

• **Software and its engineering → Application specific development environments**;

KEYWORDS

cyber physical systems, machine learning, model based design

ACM Reference format:
Charles Hartsell, Nagabhushan Mahadevan, Shreyas Ramakrishna, Abhishek Dubey, Theodore Bapty, and Gabor Karsai. 2019. Demo Abstract: A CPS Toolchain for Learning-based Systems. In *Proceedings of ICCPS '19: ACM/IEEE International Conference on Cyber-Physical Systems, Montreal, QC, Canada, April 16–18, 2019 (ICCPS '19),* 2 pages.
https://doi.org/10.1145/3302509.3313332

1 INTRODUCTION

Cyber-Physical Systems (CPS) are often required to operate in highly uncertain environments, with significant degree of autonomy. For such systems, it is typically not feasible to explicitly design for all possible situations within the environment. CPS designers are increasingly using data-driven methods such as machine learning to overcome this limitation. We define Learning-Enabled Components (LEC) as software components that have been created using machine learning methods. Such LECs have demonstrated good performance for a variety of traditionally difficult tasks such as object detection and tracking [3], robot path planning in urban environments [12], and attack detection in the power grid [9].

CPS are commonly used in mission-critical or safety-critical applications which demand high reliability and strong safety assurance. Assuring safety in these systems requires supporting evidence from testing data, formal verification, expert analysis, etc. Techniques for formal verification of learning-enabled systems are an active area of research [13] and will need to be incorporated into the safety assurance of such systems. Additionally, traceability and reproducibility in all phases of the development cycle is necessary for safety-critical systems and should be automatically handled by an appropriate toolsuite.

2 OVERVIEW

Our team is developing a model-driven development environment for such systems called the Assurance-based Learning-enabled CPS (ALC) Toolchain. Our approach combines multiple Domain Specific Modeling Languages (DSMLs) to support various tasks including architectural modeling, experiment configuration/data generation, LEC training, performance evaluation, and system safety assurance. The architecture of the ALC toolchain is centered around the WebGME infrastructure [6]: a web-based, collaborative, metaprogrammable modeling environment. Users may access the environment from most web-browsers without the need for any specialized hardware. Computationally intensive tasks (LEC training, simulation execution, etc.) are executed remotely on servers equipped with the required hardware, and all generated data is stored on a remote file server and managed with a version control system.

The toolchain supports a variant of the block diagram models from the SysML [8] modeling standard for system architecture design. *Components* are abstract building blocks of the system architecture, and may be hierarchically composed into complete *system models*. Components may contain one or more concrete *implementation alternatives* which fulfill the component's functional requirements. For example, a software object detection component may include implementation alternatives based on conventional (analytical) algorithms and machine-learning based methods. For software

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3313332

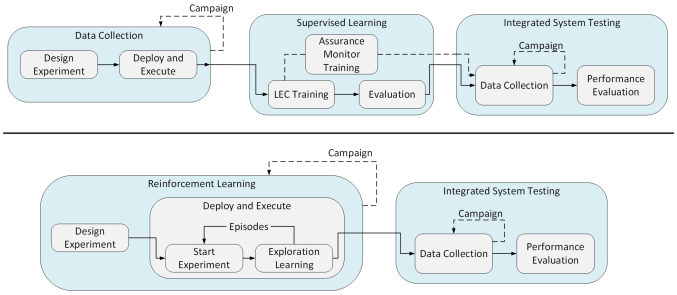

Figure 1: LEC Development workflows for supervised learning (Top) and reinforcement learning (Bottom).

components, implementation nodes contain the information necessary to load, configure and initialize the software node when the system is deployed as well as the business-logic associated with its runtime implementation. Currently, all software components are implemented using the Robot Operating System (ROS) [11].

An *assembly model* is a refinement of a *system model* where a specific implementation is selected from the available options for each component. *Experiment* models allow the user to configure and execute an assembly in the context of its simulation environment (or *gym*) (eg. UUV Simulator [1] [5], CARLA [2]) Experiments and their campaigns over multiple environmental configurations are used to collect data from a system often for the purpose of performance evaluation or LEC training.

The ALC toolchain supports workflows for training Learning Enabled Components (LECs) using either supervised or reinforcement learning techniques [7] as shown in Figure 1. In a *supervised learning* setup, an LEC model is trained against previously collected data. Then, new *experiment* or *campaign* models deploy the system with the trained LEC(s) for integrated testing and evaluation. In a *reinforcement learning* setup, the LEC is trained during execution of the system and is updated based on the environment response (state and reward) to the generated action. The training is repeated for a specified number of episodes, with each episode lasting a specified maximum time-limit or step-size. The trained LEC is evaluated by executing the reinforcement learning setup in a non-training mode.

Safety assurance is a critical component of any CPS which operates in mission-critical or safety-critical applications, and the ALC toolchain supports Goal Structuring Notation (GSN) [4] for the construction of safety case arguments (For a comprehensive introduction to safety cases and GSN, see [2]). Our toolchain also integrates new formal verification techniques [1] and dynamic assurance monitors [10] for machine learning components.

3 DEMONSTRATION EXAMPLE

As an example, we consider the design and implementation of a learning-enabled controller for an Unmanned Underwater Vehicle (UUV) tasked with following a pipeline on the seafloor. The controller components are integrated using ROS middleware with the UUV Simulator - an open source model of UUV and its environment - executed in Gazebo simulation engine. The UUV used in this example is equipped with four control fins, one propeller/thruster, a forward-looking camera, and vehicle speed and position sensors.

The controller uses the image stream from the camera and the vehicle odometry data to produce actuator commands suitable for following the pipe at a desired separation distance. The task is divided among two components: a path planner and a lower-level PID controller. The path planner uses an LEC based on a Convolutional Neural Network to determine a suitable heading for the vehicle to follow based on camera images. The PID controller translates this heading into control commands for all four fins and the thruster.

We will demonstrate the design and implementation of this controller using the ALC toolchain including: design of the various models within the WebGME environment, execution of the system in a simulated environment, and integration with Jupyter Notebooks[3] for interactive execution and performance evaluation. Wireless connectivity during the demonstration is preferable, but not required. Additionally, a poster describing the ALC toolchain and the underlying model-based methodology will be shown.

[1] uuvsimulator.github.io
[2] carla.org/

ACKNOWLEDGEMENT

This work was supported by the DARPA and US Air Force Research Laboratory (AFRL). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or AFRL.

REFERENCES

[1] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. Hyst: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.
[2] Ewen Denney, Ganesh Pai, and Iain Whiteside. Hierarchical safety cases. In *NASA Formal Methods Symposium*, pages 478–483. Springer, 2013.
[3] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.
[4] Tim Kelly and Rob Weaver. The goal structuring notation–a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*, page 6. Citeseer, 2004.
[5] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016.
[6] Miklós Maróti, Tamás Kecskés, Róbert Kereskényi, Brian Broll, Péter Völgyesi, László Jurácz, Tihamer Levendovszky, and Ákos Lédeczi. Next generation (meta) modeling: Web-and cloud-based collaborative tool infrastructure. *MPM@ MoDELS*, 1237:41–60, 2014.
[7] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
[8] OMG. OMG Systems Modeling Language (OMG SysML), Version 1.5, 2017.
[9] Mete Ozay, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. Machine learning methods for attack detection in the smart grid. *IEEE transactions on neural networks and learning systems*, 27(8):1773–1786, 2016.
[10] Harris Papadopoulos, Vladimir Vovk, and Alexander Gammerman. Regression conformal prediction with nearest neighbours. *Journal of Artificial Intelligence Research*, 40:815–840, 2011.
[11] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
[12] S. M. Sombolestan, A. Rasooli, and S. Khodaygan. Optimal path-planning for mobile robots to find a hidden target in an unknown environment based on machine learning. *Journal of Ambient Intelligence and Humanized Computing*, Mar 2018.
[13] Weiming Xiang, Patrick Musau, Ayana A. Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel A. Rosenfeld, and Taylor T. Johnson. Verification for machine learning, autonomy, and neural networks survey. *CoRR*, abs/1810.01989, 2018.

[3] jupyter.org