Demo: A Design Studio for Verification Tools

Tamas Kecskes
Vanderbilt University
tamas.kecsek@vanderbilt.edu

Taylor T. Johnson
Vanderbilt University
taylor.johnson@vanderbilt.edu

Patrik Meijer
Vanderbilt University
patrik.meijer@vanderbilt.edu

Marcus Lucas
University of California, Los Angeles
maluc@g.ucla.edu

ABSTRACT

It is often the case that analysis, simulation and verification tools from different sources share a lot of commonalities. Even though the syntax might differ, the semantics of the input models are many times the same. Theoretically this means that different tools could either be compared or combined to analyze different aspects of the same model. In practise however, this often leads to time-consuming manual tweaks and modifications on a per model basis. Additionally, the necessary environments, in form of run-times and dependencies required for the tools to be executed, are not always readily available. Images for virtual machines and containers surely ease the process for setting up the context, but this still requires manual plumbing when setting up a work-flow.

In this demo a Design Studio for combining different verification tools is show-cased. Building on top of WebGME (Generic Modeling Environment) it provides end-users with an online, browser-based editor where models are stored in a version-controlled, centralized database. At any point of the design process, a model can at the click of a button be requested to act as input for a range of different verification tools such as SpaceEx, Flow* and dReach. The Design Studio is currently hosted at cps-vo.org, but also available as open source and the entire system can, thanks to containerization, be hosted locally with a single command. At the core of this Design Studio lies the HYbrid Source Transformer, HYST, which is hosted as an independent service for translating a common model format, SpaceEx, into each tool-specific format.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

DESTION '19, April 15, 2019, Montreal, QC, Canada

© 2019 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-6699-1/19/04.

<https://doi.org/10.1145/3313151.3314057>

Figure 1: A typically Design Flow within a Design Studio

1 WHAT IS A DESIGN STUDIO?

Put in single sentence a Design Studio is a framework for modeling and performing analysis within an engineering domain. Design Studio components can be organized in the following three categories:

- **Semantic integration:** components comprise the domain of the modeling language, i.e., its metamodel that explicitly specifies the building blocks of the language and their relations.
- **Service integration:** components such as a version controlled model storage, collaborative graphical user interfaces for model- and code-editing, consistency and type checking mechanisms and model interpretation/translation services.
- **Tool integration:** interfaces and integration services towards third party tools such as run-times and verification tools.

In Figure 1, the main steps of the design flow within a Design Studio are outlined. Initially, models are designed using dedicated model editors constrained by the metamodel. Next, the checking loop starts (**step 1**), where the models are checked for additional conformance. If the required conformance conditions are not satisfied by the model, the checking mechanism refers back to the inconsistent model elements. Finally, when the conformance conditions are satisfied (**step 2**), the refined models can be analyzed (e.g. for safety properties) and/or executed (**step 3**) by using integrated, into the Design Studio, third party tools. The output of the tools is

then collected and sent back to the model editors (**step 4**) for visualization of analysis or execution results.

2 DESIGN STUDIOS IN WEBGME

WebGME (Generic Modeling Environment) [5] is a web-based, online, collaborative metaprogrammable environment. WebGME supports the design of DSMLs (Domain Specific Modeling Languages) and the creation of corresponding models with added features like version control and reusable graphical widgets.

Model interpretation is supported through model APIs in either JavaScript or Python that can run directly inside the browser and/or on the server. WebGME provides a variety of extension points to customize the application and build Design Studios [6]. These extensions include: plugins/interpreters, visualizers, decorators, REST routers, webhooks, etc. and provide a range of alternatives for integrating third party tools. The next section will explain how some of these are used to build a complete Design Studio that integrates a range of verification tools from independent sources.

3 VERIFICATION TOOLS

3.1 HYST

Hybrid system verification is a very active research area that still lacks a commonly accepted model format which would grant a better comparison of such tools and would boost the integration of such algorithms into bigger tool-chains. Also, most of the model creation is done by textual model editing, which can become cumbersome, and even small adjustments can take special attention to the semantic details. HYST [1] was developed to ease some of these hardships by providing transformation and translation capabilities for hybrid systems. Its input format is the SpaceEx [3], that is largely accepted in the field and can provide translations to several formats - like Flow* [2] and dReach [4].

3.2 HYST DS

We would like to demonstrate a Design Studio that is aimed to leverage the capabilities of HYST and on top of that provide an integrated Design Studio where users can evaluate different verification algorithm on the same benchmark models with ease and tool creators can easily integrate their tools making use of such integrated framework.

3.2.1 Editor. The metamodel, is rooted in the SpaceEx specification as that is the input format of the HyST itself. It allows the creation of timed automata that can be instantiated and composed into hybrid systems. The Design Studio provides a template project with examples and in-model documentation which should serve as a starting point for any

developer who has a system in mind. It also provides import and export capabilities to help the transition of existing models. Moreover, the editor ensures that the compositional constraints of the metamodel are always followed.

3.2.2 Automated verification. Once the modeling task is finished or reached a mature enough stage, the user can initiate any pre-configured verification by a single click. The available tools are listed in the toolbar and if they require additional configuration, that will also be asked from the user as a first step of initiation. The results can be downloaded for further data processing or can be visualized with the provided basic plotting capabilities. To make the integration easier, the Design Studio operates with docker containers, that not only provide security but allows reproducibility for the tools themselves. This also minimizes the additional steps that are required from verification tool owners to integrate their software.

3.3 Future Work

The current version of the Design Studio already covers some verification tools, but they are mostly meant to serve as an example for tool developers and to be used by the community in a more accessible manner. The work on the Design Studio will continue to provide better coverage of available tools and to engage the community so they can catch up and start providing new and new options for verification.

4 ACKNOWLEDGMENT

This research was supported by the National Science Foundation under award CNS-1521617.

REFERENCES

- [1] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. 2015. HYST: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 128–133.
- [2] Xin Chen, Sriram Sankaranarayanan, and Erika Abraham. 2015. Flow* 1.2: More Effective to Play with Hybrid Systems.. In *ARCH@ CPSWeek*. 152–159.
- [3] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*. Springer, 379–395.
- [4] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. dReach: δ -reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*. Springer, 200–205.
- [5] Miklós Maróti, Tamás Kecskés, Róbert Kereskényi, Brian Broll, Péter Völgyesi, László Jurácz, Tihamer Levendovszky, and Ákos Lédeczi. 2014. Next Generation (Meta) Modeling: Web-and Cloud-based Collaborative Tool Infrastructure. *MPM@ MoDELS* 1237 (2014), 41–60.
- [6] Patrik Meijer and Anastasia Mavridou. 2018. How to Build a Design Studio with WebGME. (05/09/2018 2018).