




Ecos: An accessible and intuitive co-simulation framework

Lars Ivar Hatledal ¹

¹ Norwegian University of Science and Technology (NTNU), Department of ICT and Natural Sciences, Norway

DOI: [10.21105/joss.08182](https://doi.org/10.21105/joss.08182)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Evan Spotte-Smith](#) 

Reviewers:

- [@ElektrikAkar](#)
- [@elac-safran](#)

Submitted: 14 April 2025

Published: 06 June 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Ecos is a cross-platform framework for running co-simulations adhering to the Functional Mock-up Interface (FMI) standard ([Blochwitz et al., 2012](#)); an open standard for model exchange and co-simulation of dynamic systems. The FMI Standard provides three interface types for difference aspects of models, namely: FMI for Model Exchange (ME), FMI for Co-simulation (CS), and FMI for Scheduled execution (Since FMI 3.0). Ecos supports the co-simulation mode, which is the most widely used interface type. An FMU (Functional Mock-up Unit) is a self-contained component that implements the FMI standard. It is packaged as a zip archive containing:

- A shared library for each supported platform, which implements a standardized C interface.
- A *modelDescription.xml* file, describing the FMU's capabilities and available variables.
- Optionally, component-specific resources embedded within the archive.

The intention of Ecos is to provide a streamlined way of working with such FMUs, and supports version 1.0, 2.0 and 3.0 of the standard with respect to co-simulation. In particular, support for FMI 3.0 ([Junghanns et al., 2021](#)) is still missing in many tools, and Ecos aims to help bridge this gap by providing partial support for this version. In collaboration with interested users, Ecos aims to gradually expand its support for FMI 3.0, working toward a more complete and practical implementation of the standard over time. Ecos also supports the System Structure & Parameterization (SSP) standard ([Köhler et al., 2016](#)), which can be used to import systems of FMUs in a structured and tool-agnostic way. Ecos consists of a Command Line Interface (CLI), as well as a C++ library, *libecos*, with interfaces provided in C and Python. The Python package is available through the PyPI package index as *ecospy*. The project is structured as a mono-repo with a major goal of being straightforward to build. This also implies few and light-weight dependencies.

Some features available with Ecos:

- Support for SSP 1.0.
- Support for FMI 1.0, 2.0 & 3.0 for Co-simulation.
- Built-in plotting capabilities with inline and XML configuration options.
- CSV writer with inline and XML configuration options.
- Scenarios - actions to run at specific events.
- Remoting - allowing models to interact across processes/computers.

In particular, remoting is a key feature of Ecos, allowing model instances to be automatically distributed across processes on a local machine. Instances may also be distributed across computers by manually booting a server application and passing their IP address to Ecos.

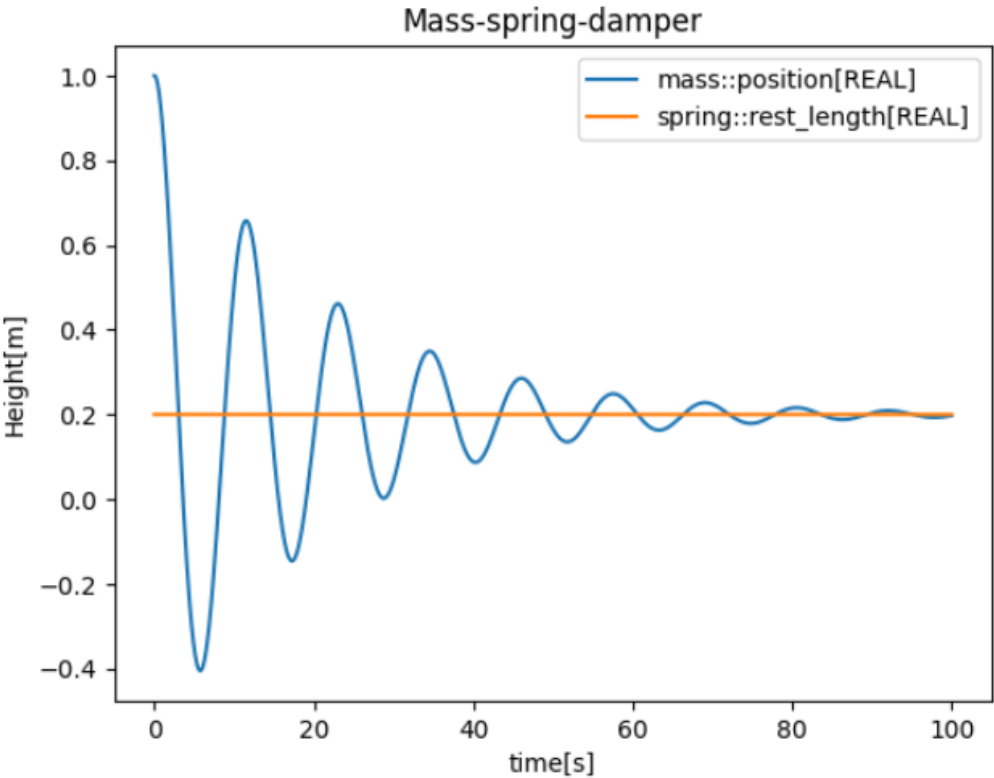


Figure 1: Ecos provides post-simulation plotting facilities.

Figure 1 demonstrates a simulation of a simple *mass-spring-damper* system with Ecos. The models are packaged following the SSP standard and subsequently simulated and plotted with *libecos*.

Statement of need

While similar tooling exists like FMPy (CATIA Systems, 2017), Vico (Hatledal et al., 2021), Open Simulation Platform (OSP) (Smogeli et al., 2020) and OMSimulator (Ochel et al., 2019), Ecos aims to deliver a higher level of flexibility, extensibility and accessibility through an easy-to-build and consume package. In particular, Ecos acts as a successor to the JVM based Vico framework, improving on accessibility, usability and performance.

Table 1 compares Ecos with some of the other available tools. While seemingly similar, Ecos niche is to provide an intuitive high-level C++ API, simple yet powerful Python, C and CLI interfaces, as well as support for all three versions of FMI for Co-simulation and built-in means of distributing simulation components across processes, while also keeping build dependencies to a minimum.

Table 1: Comparison of tools.

Feature	FMPy	OMSimulator	OSP	Ecos
Language	Python	C, Lua, Python	C++, C, Python	C++, C, Python
FMI Support	FMI 1.0, 2.0, 3.0 (ME, CS)	FMI 2.0, 3.0 (ME, CS)	FMI 1.0, 2.0 CS	FMI 1.0, 2.0, 3.0 CS

Feature	FMPy	OMSimulator	OSP	Ecos
Co-simulation	Yes	Yes	Yes	Yes
Model Exchange	Yes	Yes	No	No
GUI	Basic (Individual models)	Yes (OMEdit)	No	No
CLI	Yes	No	Yes	Yes
Remoting	Yes	No	Yes	Yes
License	BSD	OSMC-PL	MPL	MIT

The software is currently being used to support the EU project TWINVEST, where NTNU is a partner.

Future of Ecos

Ecos currently ships with a capable, but simple Jacobi-type *fixed_step* orchestration algorithm. The algorithm can run models in parallel, and individual model may run at different rates. The API is designed to be extensible, and the goal is to include more advanced orchestration algorithms. However, pursuing this should be driven by a clear user need. In this respect, users are encouraged to provide use-cases and sample simulations systems where more advanced orchestration algorithms are needed.

References

- Blochwitz, T., Otter, M., Åkesson, J., Arnold, M., Clauss, C., Elmqvist, H., Friedrich, M., Junghanns, A., Mauss, J., Neumerkel, D., & others. (2012). Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. *9th International Modelica Conference*, 173–184. <https://doi.org/10.3384/ecp12076173>
- CATIA Systems. (2017). FMPy - a Python library to simulate functional mock-up units (FMUs). In *GitHub repository*. <https://github.com/CATIA-Systems/FMPy>; GitHub.
- Hatledal, L. I., Chu, Y., Styve, A., & Zhang, H. (2021). Vico: An entity-component-system based co-simulation framework. *Simulation Modelling Practice and Theory*, 108, 102243. <https://doi.org/10.1016/j.simpat.2020.102243>
- Junghanns, A., Gomes, C., Schulze, C., Schuch, K., Pierre, R., Blaesken, M., Zacharias, I., Pillekeit, A., Wernersson, K., Sommer, T., & others. (2021). The functional mock-up interface 3.0-new features enabling new applications. *Modelica Conferences*, 17–26. <https://doi.org/10.3384/ecp2118117>
- Köhler, J., Heinkel, H.-M., Mai, P., Krasser, J., Deppe, M., & Nagasawa, M. (2016). *Modelica-association-project “system structure and parameterization”-early insights*. <https://doi.org/10.3384/ecp1612435>
- Ochel, L. A., Braun, R., Thiele, B., Asghar, A., Buffoni, L., Eek, M., Fritzson, P., Fritzson, D., Horkeby, S., Hällquist, R., & others. (2019). OMSimulator-integrated FMI and TLM-based co-simulation with composite model editing and SSP. *Modelica*, 157–007. <https://doi.org/10.3384/ecp1915769>
- Smogeli, Ø. R., Ludvigsen, K. B., Jamt, L., Vik, B., Nordahl, H., Kyllingstad, L. T., Yum, K. K., & Zhang, H. (2020). Open simulation platform—an open-source project for maritime system co-simulation. *19th International Conference on Computer and IT Applications in the Maritime Industries*.