The MoDeNa multi-scale simulation software framework

Henrik Rusche†
Wikki Ltd., London, United Kingdom
Heinz Preisig
Norwegian University of Science and Technology, Trondheim, Norway

The MoDeNa project [1] aims at developing, demonstrating and assessing an easy-to-use multi-scale software framework application under an open-source licensing scheme that delivers models with feasible computational loads for process and product design of complex materials. The concept of MoDeNa is an interconnected multi-scale software framework. Four scales will be linked together by this framework namely the nano-, micro-, meso-, and macroscale (see Figure 1). As application cases we consider polyurethane foams (PU), which are excellent examples of a large turnover product produced in a variety of qualities and of which the properties are the result of designing and controlling the material structure on all levels of scale, from the molecule to the final product.

Multi-scale coupling requires the exchange of information between software instances developed for specific scales in a consistent way. In order to achieve this, generating consistent representations for models and data is necessary. The information exchange is governed by protocols and may occur in two ways, namely:

- "forward mapping" (passing information from the microscopic to the macroscopic scale in upward direction)
- "backward mapping" (passing information from the macroscopic to the microscopic scale in downward direction)

"Forward mapping" is relatively straightforward, while "backward mapping" inevitably requires iteration since changing the operating conditions at the fine level changes the feedback to the coarse level. "Backward mapping" can be realised by "two-way coupling" or by "fitting surrogate models". The first approach usually requires exchange of large amounts of data during runtime that may be expensive either due to the complexity of the data exchange or the computational cost associated with executing the microscopic-scale simulation. In such cases, replacing the microscopic-scale simulation with a surrogate model presents the only viable alternative. This operation inherently constitutes a transfer of data across scales and MoDeNa is unique in that it focuses on this approach.

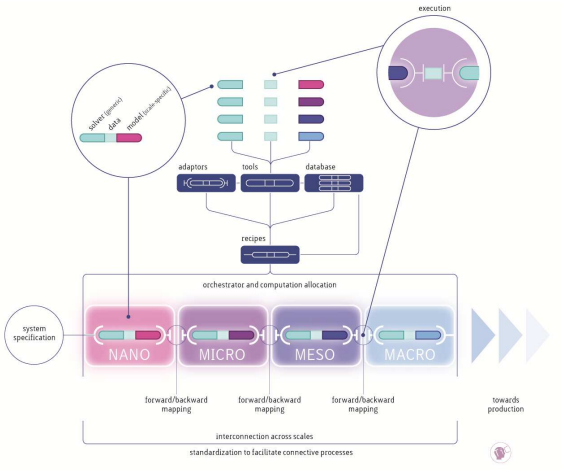

Figure 1: Conceptual structure of MoDeNa, coupling applications and surrogate models into tools, which form sequences through recipes and the orchestrator.

A typical operation sequence starts a macroscopic-scale simulation which instantiates one or more surrogate models. When the validity of a model is violated, a design of experiment operation is triggered. It creates inputs for a set of microscopic-scale simulations. When all experiments are finished, the parameter estimation component is invoked which updates the model parameters. Next, the macroscopic-scale simulation is restarted. It should be noted, that the MoDeNa software framework supports application and model dependencies across multiple scales.

The MoDeNa framework handles the communication across scales through recipes and adapters as shown in Figure 1. Recipes perform simulations by executing applications (in-house codes or external software packages such as FOAM, Materials Studio, Predici) for a given set of inputs. Adapters handle the communication

†Correspondence: h.rusche@wikki.co.uk

with the MoDeNa software framework. Both, recipes and adapters are application specific. Adapters exist as outgoing and incoming adapters. Outgoing adapters are relatively straight forward in that they perform a mapping operation (such as averaging) and communicate the results. The averaging process may have to be started and performed within the application (e.g. for time averaging). However, the results can usually be submitted in a separate process after the simulation is finished. Incoming adapters are more complicated since they usually require to embed surrogate models within the applications.

The software framework consists of an orchestrator, a database and a interface library. The orchestrator is based on FireWorks [2] and constitutes the backbone of the software framework in that it schedules simulations as well as design of experiments & parameter estimation operations which make up the work-flow of the overall simulation. It is very much like a dynamic work-flow engine, in which the different applications are "orchestrated" to obtain information, analyse and pass it to the other operations. The NoSQL database MongoDB [3] is used to store the state of the work-flow as well as the surrogate models together with associated data such as model parameters, data used for parameter estimation, and meta-data.

The interface library consists of two parts: A high-level python module providing access to the database as well as design of experiments and regression analysis capabilities by building on MongoEngine [4] and R [5], respectively. The second part is a low-level library providing unified access to the surrogate models. This component is written in C to ensure interoperability across platforms and target applications while providing the computationally efficient model execution required by the applications. The library is loaded as a shared library by the macroscopic-scale applications or as a native python extension by the high-level python module ensuring that all components instantiate identical model implementations. Complex operations such as database access are referred back to the high-level python module using call-back mechanisms.

References

[1] MoDeNa project web-site `http://www.modenaproject.eu/`

[2] Fireworks project web-site `http://pythonhosted.org/FireWorks/`

[3] MongoDB project web-site `http://www.mongodb.org/`

[4] MongoEngine project web-site `http://www.mongoengine.org/`

[5] R project web-site `http://www.r-project.org/`

†Correspondence: `h.rusche@wikki.co.uk`