MuPIF open simulation platform: architecture and applications

Bořek Patzák, Vít Šmilauer, Martin Horák, Stanislav Šulc, Edita Dvořáková

Czech Technical University in Prague, Czech Republic

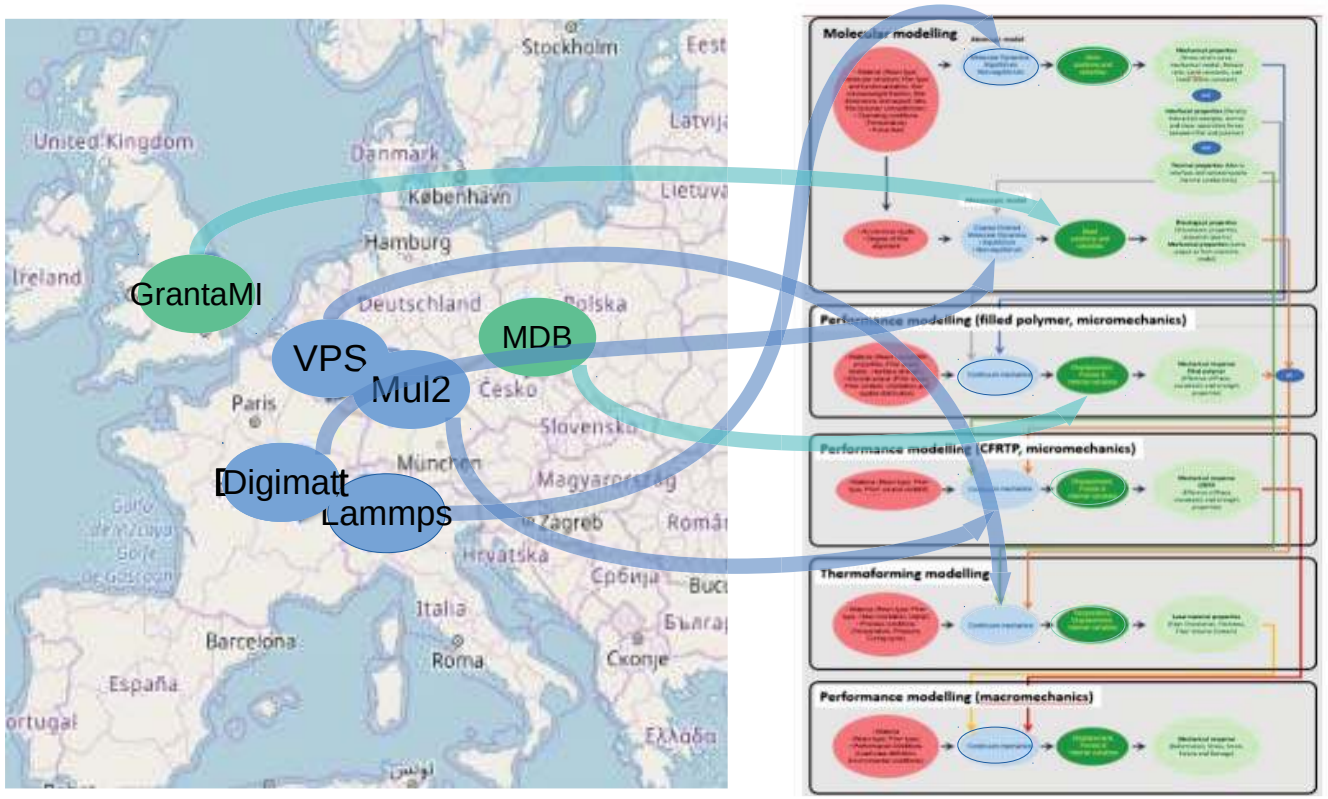
Outline

- Motivation
- Platform architecture & features
- Graphical workflow editor
- MuPIF applications
- Conclusins

Motivation

Combine existing tools to build a **customized multi-physics and multi-scale simulation workflows**

Provide inter-operable, distributed plug and play architecture enabling to define complex workflows, plug in independent simulation tools, and independent on particular data formats



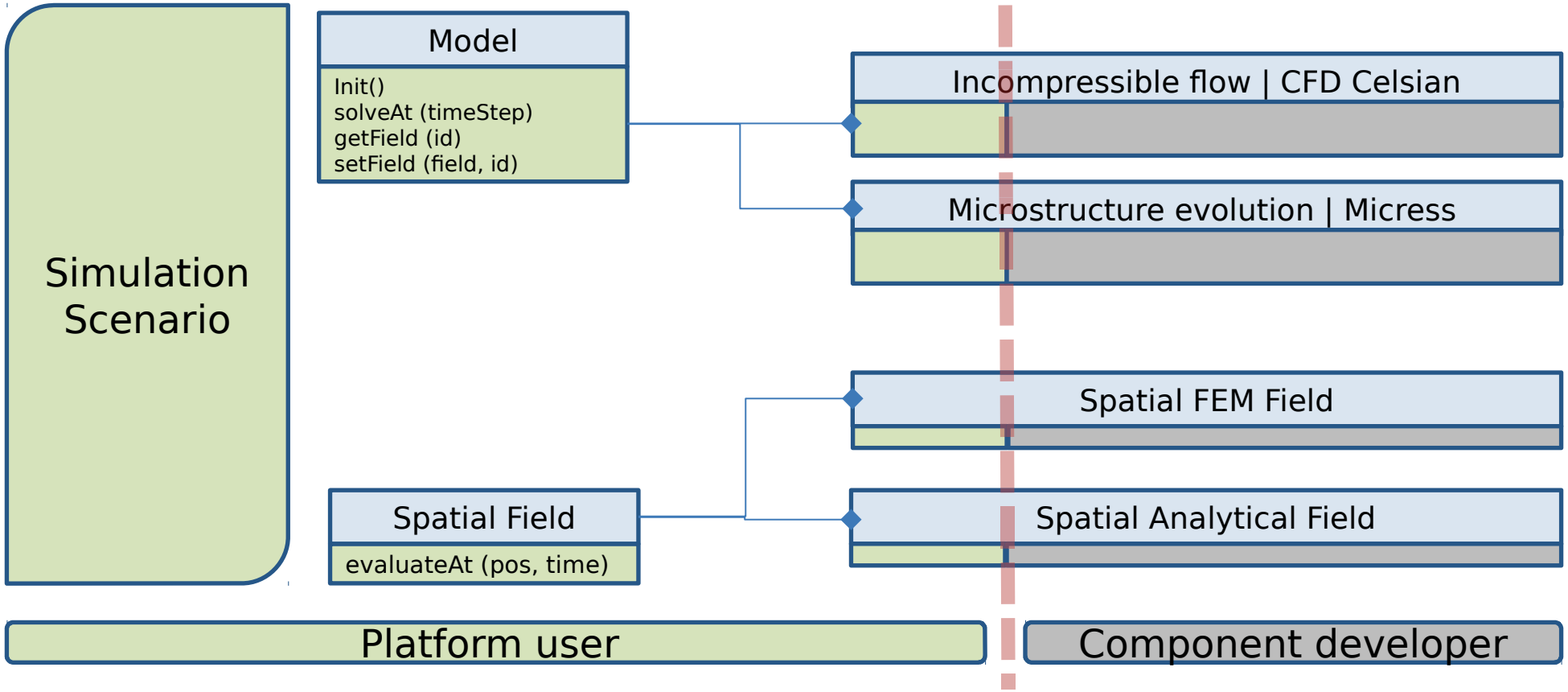
MuPIF platform architecture

Design based on interacting, distributed objects (components)

- Instead of trying to standardize data structures, MuPIF is focused on identification and standardization of services on similar objects (models, workflows, spatial fields, micro structures,...)
- **Models, workflows and complex data represented as software components** with **standardized interfaces**
  - **Each generic object (model, workflow, complex data types) represented by abstract class which defines abstract interface** (set of standardized services)
  - **Interfaces allow to communicate with any object using generic services, hiding the particular implementation details and allowing plug-and-play architecture**
- **Data and services** (algorithms) operating on data **encapsulated** in a component and **exchanged** between applications → Models will get data and operations on data in one consistent package, do not have to interpret data themselves
- **Components** in MuPIF can represent **local as well as remote** objects → **distributed workflows, distributed data, enabling business model based on software or data as service and marketplaces integration**

Component Approach

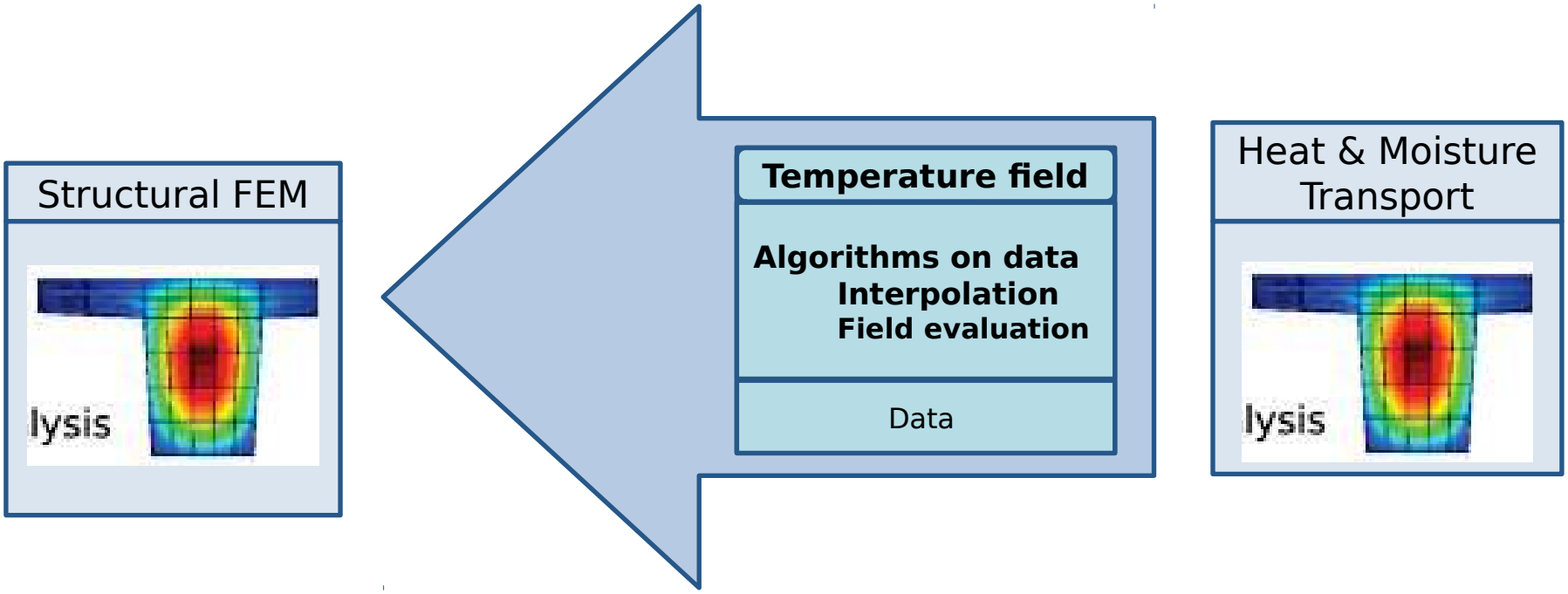
All components of the same type can be manipulated using the same interface defined by abstract class, ellowing to hide internal details of particular object (data storage, functions) hidden



Data Component Approach

Data component approach -> **data and algorithms** operating on data **encapsulated in components** and **exchanged** between applications

- less numerical errors due to data translation,
- consistency (data and algorithms together)
- efficiency (implemented only once)

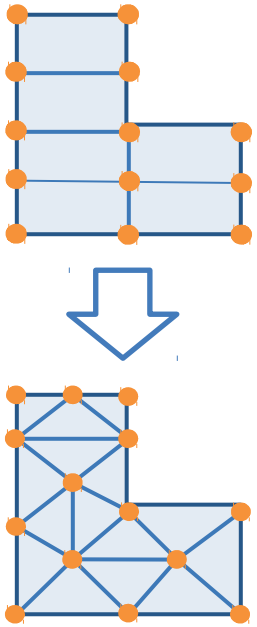


Data component Approach

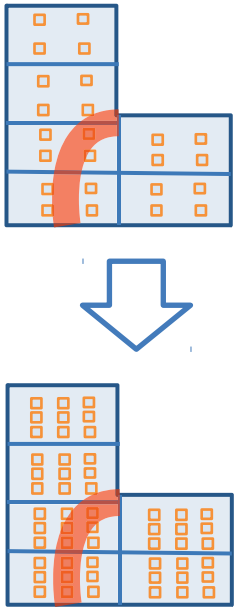
Data component approach -> **data and algorithms** operating on data **encapsulated in components** and **exchanged** between applications

- less numerical errors due to data translation,
- consistency (data and algorithms together)
- efficiency (implemented only once)

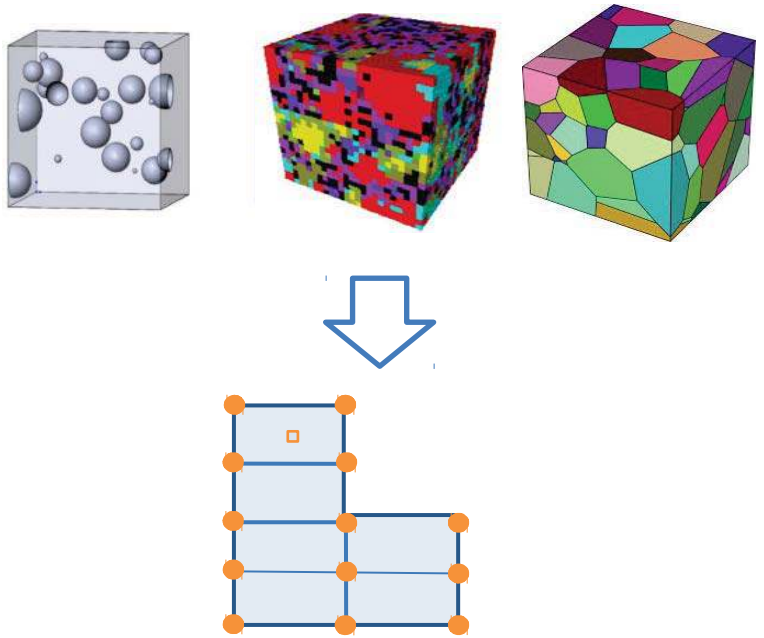
Arbitrary discretizations



Physics aware mapping



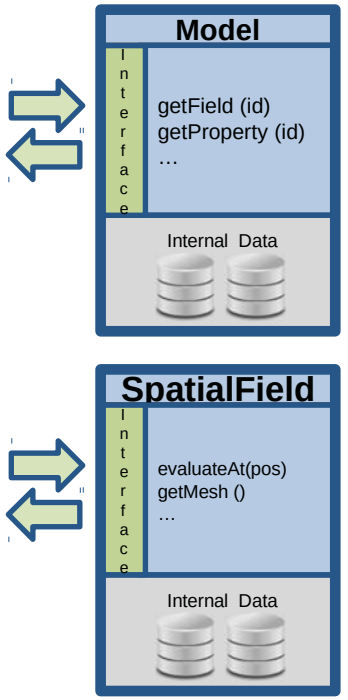
Multiple data representations



MuPIF platform architecture

Advantages of component based design

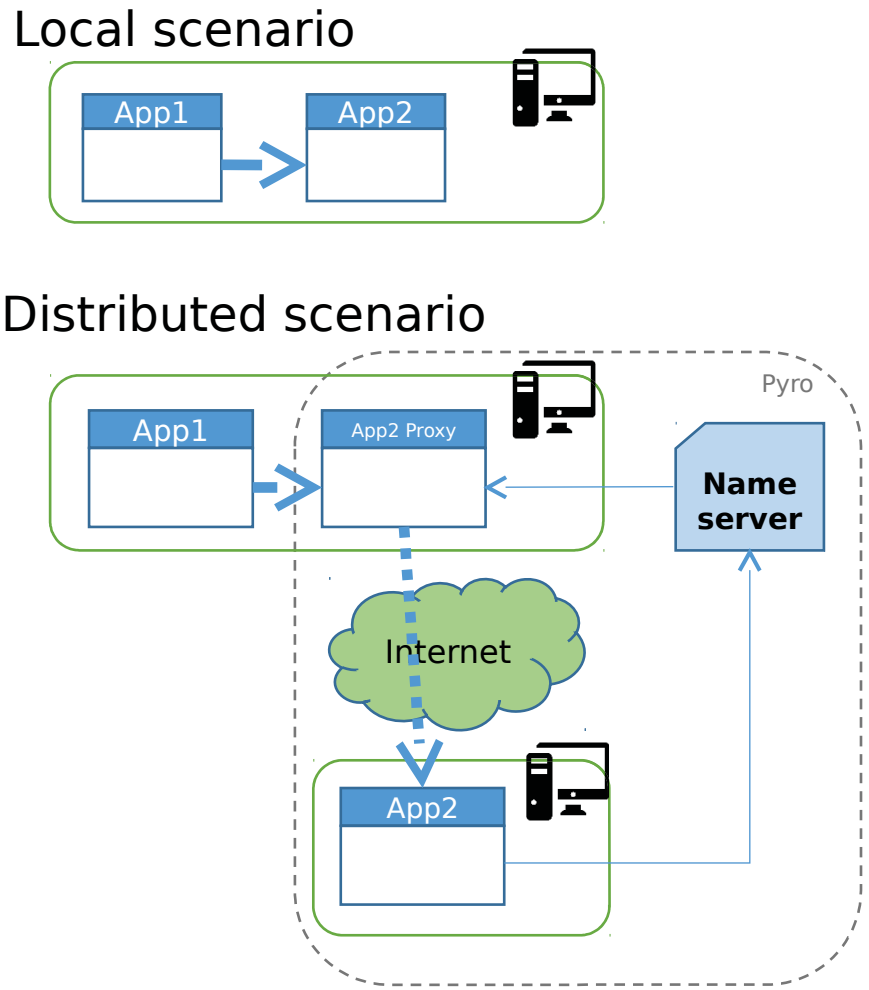
- Internal data,storage formats and data location naturally hidden behind common interface, native support for multiple data formats (objects manipulated using the same interface)
- Instead of trying to standardize data structures, MuPIF is focused on identification and standardization of methods on similar components (models, fields, microstructures,...)
- Models will get data and operations on data in one consistent package, do not have to interpret data themselves
- Meta data can be attached to any component
- Components in MuPIF can represent local as well as remote, distributed data



MuPIF distributed architecture

The MuPIF components can be distributed

- **Transparent**, **distributed** object system fully integrated using so-called proxies; **security** based on SSL or VPN
- Integrates with existing grid middleware (Condor) or provides own resource manager
- Support new business models, such as software as a service



MuPIF: interfacing models

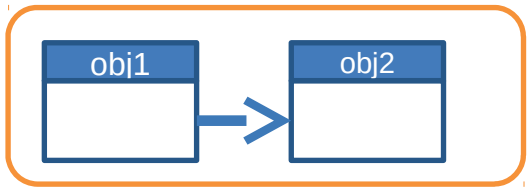
- **Model interface defined by Model API**
  - Data exchange services (get, set data)
  - Steering services (initialize, solve step, terminate)
- **Model API independent on model type** (electronic/atomistic/mesoscale/continuum) **and actual implementation** (programming language, open/close source)
  - **Native** API implementation using direct calls to application's function, usually using generated Python interface (Swig, Boost)
  - **Indirect** API implementation using wrapper around I/O streams or files, more common

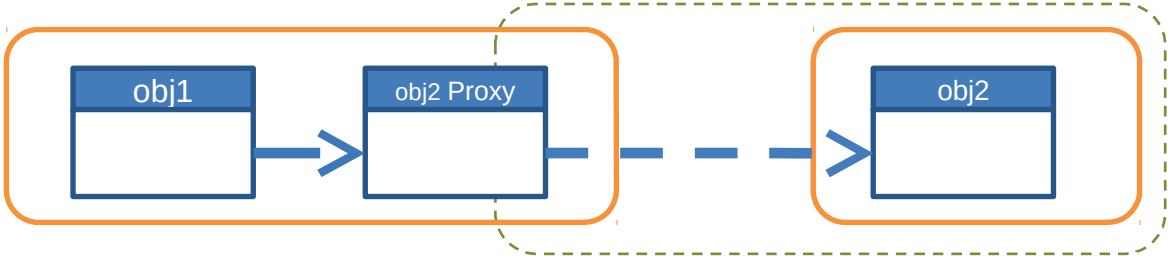
Distributed environments, remote objects

A proxy is local representation of remote object.

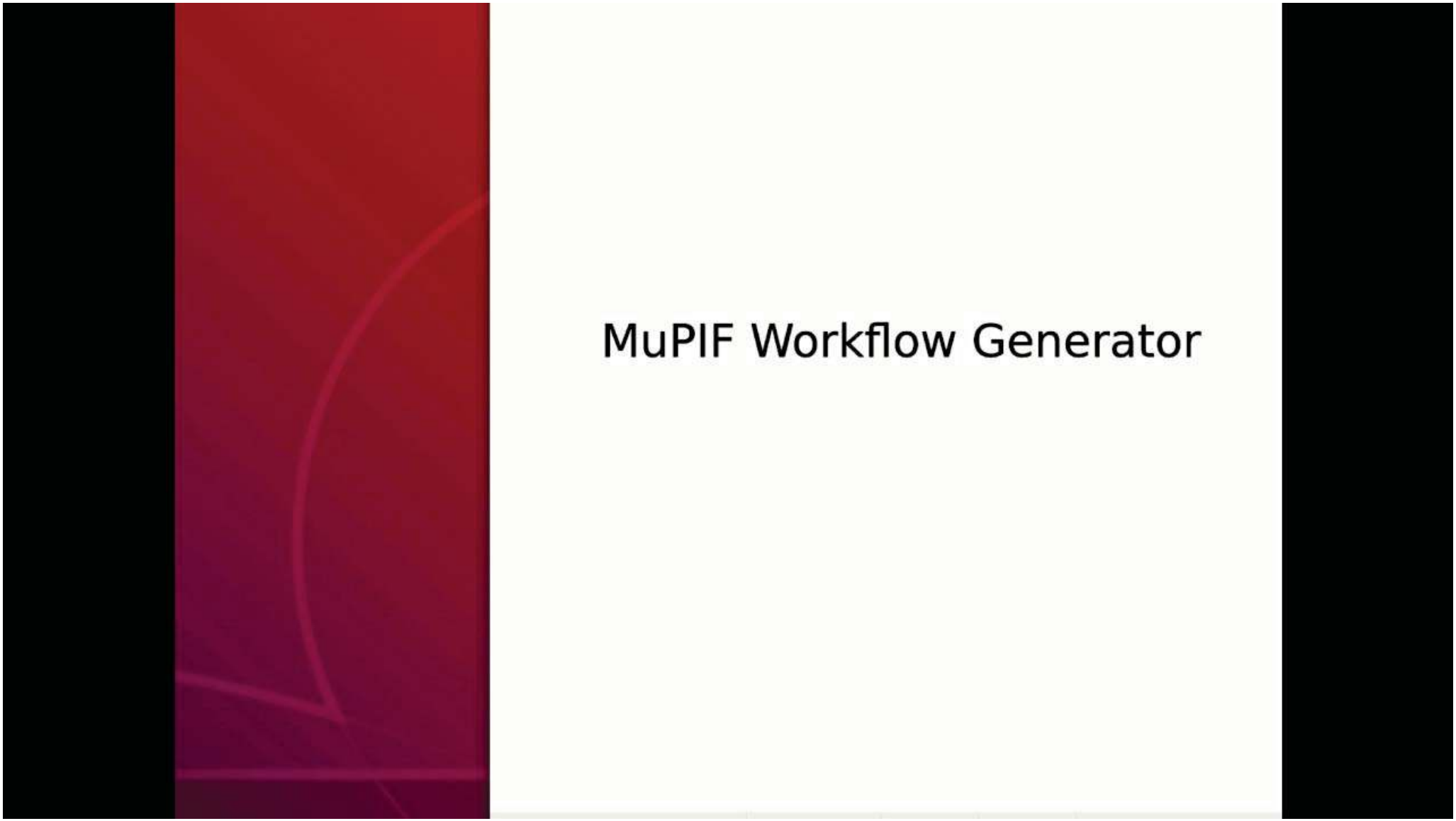
- It intercepts the method calls you would normally do on an object
- transfers the call with arguments to the computer that contains the remote object
- transfers the results back to the caller

Calling code doesn’t have to know if it’s dealing with a normal or a remote object, because the code is identical.



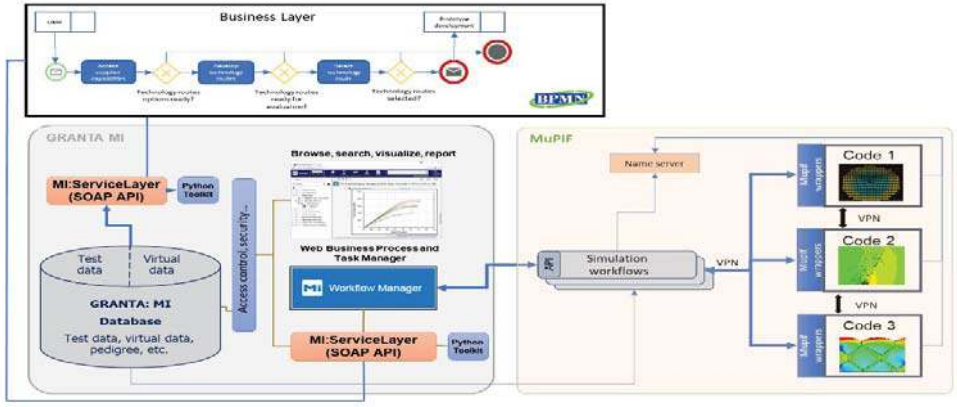


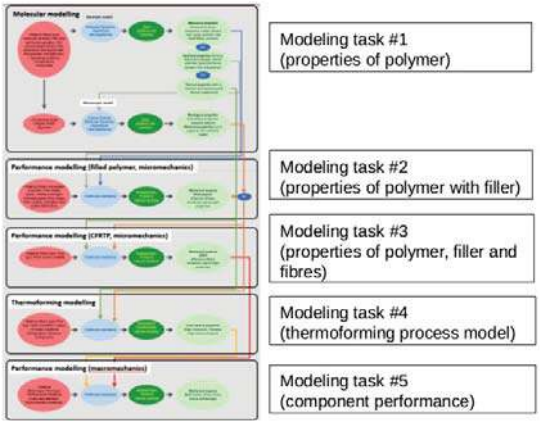
Graphical workflow editor

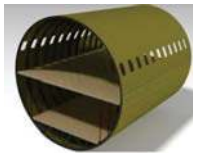


MuPIF applications

- FP7 project **MMP** “Multiscale Modelling Platform: Smart design of nano-enabled products in green technologies”, project no: 604279, 2014-2016
  - Optimization of selenization process
  - Coupled opto-thermal model for LED design
- Czech Grant Agency: Modeling Fire-Exposed Structures
- **Composelector** H202 project – integration of MuPIF into BDSS, three use case: composite airplane frame, composite automotive leaf spring and car tyre optimization







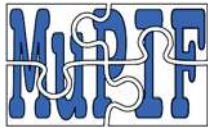
COMPOSELECTOR
Polymer Composites
Business Decision Support System

Conclusions

- MuPIF is open simulation platform supporting complex, distributed data and workflows
- Different tools and data sources/representations can be combined by using standardized APIs
- MuPIF is distributed under open source license (LGPL) and available at www.mupif.org

Acknowlegdements

- Czech Grant Agency (Project No. P105/10/1402.), 2012-2014
- EU FP7 project “Multiscale Modelling Platform: Smart design of nano-enabled products in green technologies”, project no: 604279, 2014-2016
- H2020 Composelector project, Multi-scale Composite Material Selection Platform with a Seamless Integration of Materials Models and Multidisciplinary Design Framework, project no. 721105, 2017-2021.



Thank You