**Milestone MS3**

Milestone MS3: Running a first simulation chain on a distributed configuration (TRL4)

Responsible Partner: Czech Technical University in Prague

Contributors: Consortium partners

Dissemination Level: Public

Distribution List: Consortium members, European Commission, PTA

Due Date: M24

Preparation Date: v1: 23.11.2015

v2: 29-1-2016

Report Status: FINAL

---

**NMP.2013.1.4-1:** Multiscale Modelling Platform: Smart design of nano-enabled products in green technologies

Project nr: 604279, collaborative project


Project website: http://www.mmp-project.eu

Coordinator: Dr. Jan-Paul Krugers, jan-paul.krugers@tno.nl

Ir. Sjoukje Wiegersma, sjoukje.wiegersma@tno.nl

Project partners: TNO, TU/e, CTU, Access, VTT, CelSian, Philips, Abengoa
Table of Content
1 Introduction ..................................................................................................................... 2
2 WP2 simulation chain ....................................................................................................... 3
  2.1 Description of simulation chain .................................................................................. 3
  2.2 Description of individual models ................................................................................. 5
  2.3 Description of test setup .............................................................................................. 7
  2.4 Results ......................................................................................................................... 11
  2.5 Internal WP2 evaluation .............................................................................................. 12
3 WP3 test chain .................................................................................................................... 13
  3.1 Description of simulation chain .................................................................................. 13
  3.2 Description of individual models ................................................................................. 13
  3.3 Description of test setup .............................................................................................. 16
  3.4 Results ......................................................................................................................... 17
  3.5 Internal WP3 evaluation .............................................................................................. 18
4 Conclusions ...................................................................................................................... 18

1 Introduction

The approach followed in the MMP project is based on a system of distributed, interacting objects designed to solve mathematical tasks of underlying chemo-physical problems. The operational environment is established by interconnecting the distributed resources (individual models, data sources, and hardware) provided by individual partners into an integrated infrastructure allowing performing complex simulation scenarios. The purpose of this milestone is to validate the developed platform technology in the lab in a distributed environment and document the capability to run preliminary simulation chains in distributed environments (TRL4). This milestone also evaluates platform stability, job allocation, model handshaking, data exchange and other aspects of the platform in prototype setup close to a final operational environment.
2 WP2 simulation chain

2.1 Description of simulation chain

This simulation chain is the prototypic scenario considered in WP2, focused on simulating phosphor converted lighting systems. This scenario involves the following models:

- Continuum model of scattering
- Continuum model of ray-tracing
- Continuum model of thermal conductivity

The prototype workflow considered is schematically shown in Figure 1.

![Figure 1 Prototype workflow considered in WP2](image)

The generic steps for a simulation chain as devised for the current test can be seen in Figure 2. Obviously it should start by connecting to the models in different computers running the models described above. This setup is necessary for future configuration of properties and exchange of data among models. After this, the initial meshes associated to the continuum model need to be obtained so they can be later used in the other models. Similarly it follows a data transfer between the continuum models of scattering and ray tracing, respectively. After all initial data is transferred and set, the simulation chain can proceed to find the solution for the different quantities, e.g. fields,
properties, etc. Finally, the user can make use of the available data by accessing the relevant methods in each model. In Error! Reference source not found. the actual pipeline can be seen with the different models and computers where they run.

![Figure 2](image)

**Figure 2** Overview of the simulation chain workflow as used in this test scenario.

![Figure 3](image)

**Figure 3**: Pipeline of the simulation chain with each model and the hardware where it runs.
2.2 Description of individual models

The following models are involved in this scenario:

- The ray-tracing continuum model, represented by MMP ray tracer developed by VTT. The application is written in C++. The API allows to receive scattering parameters (communicated as properties) and phase function.
  - The model uses a well-known technique of Monte-Carlo ray tracing where a large number of light rays are propagated through the 3D model according to probability distribution of scattering events. The probability of scattering is computed by the Mie theory from the microscale.
  - The model calculates the angle dependent spectrum (wavelength, intensity), and the thermal absorption of the LED phosphor material.
  - The LED geometry is defined using a JSON-formatted file.
  - Outputs:
    - Absorption distribution
    - Photometric properties of the LED: Correlated Colour Temperature, Colour coordinates and total radiant power.

- The continuum model of Scattering, provided by MMP Mie solver developed by VTT.
  - The developed solver uses the Mie theory of scattering to calculate the scattering probabilities and scattering angles for the ray tracer. The Mie theory is a specific solution of Maxwell’s equations for a spherical particle and a plane wave.
  - The model uses widely available python port of Mie solver code developed by Bohren and Huffman.
  - No geometry definition is needed.
  - Boundary conditions:
    - Plane wave light (True in the case of LED)
    - Spherical particle. (Not true in the case of LED, but a good approximation)
    - Particle is alone in the system. (Not true in the case of LED, but a good approximation)
  - Outputs:
    - Effective scattering cross sections
    - Probability distribution of scattering angles.

- The thermal model is a continuum model, provided by the commercial software tool COMSOL. MATLAB and the COMSOL LiveLink to MATLAB are used to prepare the data and connect the COMSOL model to MuPIF.
  - The model uses (transient) heat equations given by:

\[
\rho C_p \frac{\partial T}{\partial t} + \rho C_p \mu \cdot \nabla T + \nabla \cdot q = Q + Q_{\text{rad}}
\]

\[q = -k \nabla T\]
Special boundary conditions, such as heat convective cooling with air, are already available in COMSOL via empirical sub-models, i.e., boundary heat fluxes with flux coefficients that depend on surface and ambient temperatures, pressure, geometrical parameters. Similar boundary condition models for radiation are also available. Sub-models to describe thin layers are built into COMSOL. A large range of solvers and solver settings is available and in general the default COMSOL settings provide robust results. In the models so far, there was no need for special settings.

Model geometry. The LED package (in the center) is mounted on an Aluminum heat sink (complete square)

Model part with the silicone part, that is connected to the VTT models via the platform.

The corresponding application interfaces have been developed as well, basically implementing a bridge from the individual models into MuPIF. Each software package requires proprietary licenses.
2.3 Description of test setup

Given the versatility of the framework that allows for remote objects communication (Pyro4), there is no single way to configure a working connection for a simulation chain. The setup presented in the following aims to illustrate the one used in this particular test and such configuration is considered a standardized way in any form.

Ultimately the test configuration was run on computers that were all accessible for TNO. Although running from a single party is not strictly necessary, we adopted such a strategy because current firewall settings were too cumbersome to be bypassed rapidly. Nevertheless, the test was run in computers physically separated from each other, located in two different cities within the Netherlands, namely Eindhoven and Hengelo.

Among other things we needed to guarantee that at least one computer had an SSH server running and also that the SSH port (often port 22) is reachable for the other computers. This will allow for seamless communication, in addition to connectivity via secure port tunnelling.

The main features of the 2 computers involved in this test are summarized in Error! Reference source not found.. Computer A is a Linux machine running Red Hat Enterprise, access to external computers is provided via OpenSSH version 5.3.

Computer B is a PC that runs Windows 7 Enterprise. In both computers Pythonv2.7, Pyro4 v4.39 and MuPIF v0.11.11 were installed. Additional software might be necessary in each computer depending on the application that is running. See details for each application elsewhere.

| Table 1 Summary of hardware and preliminary software common to each computer taking part in the test |
|-----------------|-----------------|-----------------|
| Feature | Computer A | Computer B |
| Operating system | Red Hat Enterprise | Windows 7 Enterprise |
| SSH Server | OpenSSL 5.3 | N/A |
| SSH Client | OpenSSL 5.3 | Putty 0.66 |
| Python version | 2.7 | 2.7 |
| Pyro4 version | 4.39 | 4.39 |
| MuPIF version | 0.11.11 | 0.11.11 |
| Running | appMie, appTracer | appTNO |

A schematic representation of the initial configuration, before the simulation chain can be run, is displayed in Figure 4. In the computer labeled “A” a nameserver was started. Separately, in the same computer two applications namely appMie and appTracer are also started. Each of those is assigned to a daemon previously registered to the above mentioned nameserver.
In the presented prototypic workflow and its test runs, the necessary SSH tunnels have been created in advance. This includes the SSH tunnels from computer “B” to computer “A”, mapping the connection to nameserver and application daemons serving the application interfaces.

After having such secure connections made, we only needed to start individual instances of the applications in computer B and associate them with a daemon. The Figure 5 shows the listing of registered objects in the nameserver.

```
--------START LIST
mieApp  -->  PYRO:obj[:3cc1a779b624d6e42766ebad0a11f0109localhost:44384
appTNO  -->  PYRO:obj[:d455778ea990417b8a8cdd2653f2851580localhost:60000
Pyro::NameServer  -->  PYRO:Pyro::NameServer=localhost:9090
tracerApp  -->  PYRO:obj[:5306f891ec465d91c0c0e850ef1178localhost:44385
--------END LIST
```

Figure 5 Example of the objects registered at the nameserver after initial configuration.

From the above it is clear that only specific local ports were exposed with no additional details on individual participating computers. This ensures an additional degree of security.

Later with all the applications registered into the nameserver, it was possible to combine them in a simulation chain that uses data transfer and steering provided by MuPIF. For this test the simulation chain has been run from computer A, where also tracerApp and mieApp are located. Still appTNO is running at computer B. Such choice was made by convenience, being possible always to setup a third computer from which the simulation chain can be driven. This will require, of course, additional port mappings. In the following the most important steps of such a simulation are presented.
1. Location of the nameserver and connection to each application therein.

```python
ns=Pyro4.locateNS(port=9091)
apptNO = Pyro4.Proxy(ns.lookup('TNO'))
appMie = Pyro4.Proxy(ns.lookup('Mie'))
apprTracer = Pyro4.Proxy(ns.lookup('Tracer'))
```

2. Requesting fields from appTNO and transfer of data into appTracer

```python
fHeatSurf=appTNO.getField(FieldID.FID_HeatSourceSurf,0)
fHeatVol=appTNO.getField(FieldID.FID_HeatSourceVol,0)
tracerApp.setField(fHeatSurf)
tracerApp.setField(fHeatVol)
```

3. Property assignment (only one shown)

```python
pPhase = mieApp.getProperty(PropertyID.PID_InverseCumulativeDist, 0, objID.OBJ_PARTICLE_TYPE_1)
tracerApp.setProperty(pPhase, objID.OBJ_PARTICLE_TYPE_1)
```

4. Solve for Mie and Tracer models

```python
mieApp.solveStep(0)
tracerApp.solveStep(0, runInBackground=False)
```

5. Transfer of fields back to appTNO

```python
fHvolnew = tracerApp.getField(FieldID.FID_HeatSourceVol,0)
appTNO.setField(fHvolnew)
```

6. Solve for TNO models

```python
appTNO.solveStep(0)
```

7. Extract results from models (e.g., temperature field at timestep 1 from appTNO)

```python
(t,x) = appTNO.getAssemblyTime(1)
fTemp = appTNO.getField(FieldID.FID_Temperature,t)
```

Examples of the results obtained from running the previous simulation chain are displayed in Figure 6, Figure 7 and Figure 8.
Figure 6 Example of the silicone filled with phosphorus particles as a result of calling the appMie and appTracer applications.

Figure 7 Visualization of the temperature field, (left) for the first time step and (right) for an intermediate step before completion of the solveStep method of appTNO.
2.4 Results

At present a proposed setup has been tried for the first time to test a simulation chain, making use of models developed independently by VTT and TNO. The interfacing is done using MuPIF as a common platform to transfer data among them. Two different computers with distinct operating systems have been configured to work with same software versions of Python, Pyro4 and MuPIF. These computers are physically separated but accessible to each other, with one of them running an SSH server.

Within this test we have been able to find a setup that allows connecting different models running on separate computers as if they were on the same machine. For the test to work we needed to be capable of creating various SSH connections. By adopting this method we avoided exposing directly the IP of each machine in the Pyro4 objects. Additionally, all the data is transferred via the tunnels that were opened using SSH, therefore this data exchange is encrypted and secured.

The simulation chain as presented earlier ran successfully. This means that from an external script it was possible to allocate the applications, set properties, transfer information and extract results that made use of the combination among them. The overall performance was very good. An indication of this can be seen in the logs and the time spent in processes that involve any data transfer, as can be seen from Figure 9. Virtually there is little time needed for transferring large amounts of data, i.e. 4 seconds to send 2 fields with more than 20,000 nodes equivalent to a few MBs in size. The largest amount of time spent in a single instruction was during the execution of appTNO, nearly 6 minutes, where no data was actually transferred and only the model was being solved.
2.5 Internal WP2 evaluation

<table>
<thead>
<tr>
<th>Connection stability:</th>
<th>1, connection was stable during every iteration and simulation run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1(stable) - 5(weak)</td>
<td></td>
</tr>
<tr>
<td>Data throughput:</td>
<td>1, typical data transfer speeds as found in ftp, http, SSH.</td>
</tr>
<tr>
<td>1(fast)-5(slow)</td>
<td></td>
</tr>
<tr>
<td>API Implementation:</td>
<td>3, conceptual understanding of MuPIF in combination with Pyro4</td>
</tr>
<tr>
<td>1(easy, clear) - 5(difficult)</td>
<td>while working on a framework that is in constant development.</td>
</tr>
<tr>
<td>Overall platform</td>
<td>2, optimizations are still in place. However performance so far is</td>
</tr>
<tr>
<td>performance:</td>
<td>above decent.</td>
</tr>
<tr>
<td>1(excellent) - 5(bad)</td>
<td></td>
</tr>
<tr>
<td>Setting up the network:</td>
<td>4, most internal networks have restrictive firewalls. Also it has</td>
</tr>
<tr>
<td>1(simple)-5(difficult), reason</td>
<td>been difficult to find a computer that can be accessed easily for all parties involved.</td>
</tr>
<tr>
<td>Problems encountered</td>
<td>The main setback has been the lack of a centralized server where a nameserver could run and which is commonly accessible to all parties.</td>
</tr>
<tr>
<td>Additional comments</td>
<td>Job allocation has only being used partially (by VTT). TNO has not work on it yet. That is the reason why the Job Manager did not appear in the current simulation chain.</td>
</tr>
</tbody>
</table>
3 WP3 test chain

3.1 Description of simulation chain
In this work package, a simulation chain is targeted which allows the prediction of the microstructures and PV properties of a CIGS component fabricated in a selenization process. The following prototypic simulation chain is intended for a test of the platform components necessary to enable a full two-way coupling between two continuum models on different scales.

The fabrication process of the CIGS component is modelled with a computational fluid dynamics macro simulation from which certain glass surface locations are scheduled for further analysis of the microstructure. The microstructure evolution of the thin film is done by a multi-component, multi-phase simulation. The resulting selenium concentration is given back for homogenization to the macro level which will give an estimate of the emissivity as a boundary condition for the next time step of the macro simulation.

3.2 Description of individual models
The following models are involved in the scenario:

- Continuum CFD model, implemented in X-Stream software package provided by CelSian Glass & Solar B.V., The Netherlands.
In the flow domain laminar flow equation in combination with the energy and radiation equation will be solved. It is assumed that the gas that is flowing is nitrogen with constant material properties. In the glass substrate the energy equation is solved. The glass material has constant material properties.

Input data:

- The sketch of the input geometry can be found in the next figure:

![Figure 11: The component geometry of prototype scenario](image)

- The geometry (see Figure 11) consists of the glass substrate and part of the flow domain above the substrate. Dimensions are indicated in the sketch. The origin is at the center of the top plain of the glass substrate, indicated by the blue dot in the sketch. The grid uses grid cells of 1x1x1 mm. This means that the glass substrate surface in contact with the flow domain will consist of 30x30=900 grid cells.

- The boundary conditions in the flow domain are the following:
  - Plane@x = -1.5 cm; Inlet boundary: fixed rate \((2.5\cdot10^{-7} \text{ kg/s})\) and temperature \((1175 \text{ K} \approx 902 \degree \text{C})\)
  - Plane@x= 1.5 cm; Outlet boundary: fixed pressure \((1000 \text{ Pa})\), zero gradient for other variables
  - Plane@y = -1.5 cm; Wall boundary: no-slip for velocity and adiabatic wall (heatflux=0.0 W/m²)
  - Plane@y = 1.5 cm; Wall boundary no-slip for velocity and adiabatic wall (heatflux=0.0 W/m²)
  - Plane@z = 1.6 cm; Wall boundary no-slip for velocity and fixed temperature \((1175 \text{ K} \approx 902 \degree \text{C})\)
  - Plane@z= 0.0 cm; Domain boundary: temperature & heat flux exchange between the domains
• The boundary conditions in the glass domain are the following (only temperature boundary conditions):
  - Plane@x = -1.5 cm  Wall boundary: adiabatic wall (heatflux=0.0 W/m²)
  - Plane@x= 1.5 cm  Wall boundary: adiabatic wall (heatflux=0.0 W/m²)
  - Plane@y = -1.5 cm  Wall boundary: adiabatic wall (heatflux=0.0 W/m²)
  - Plane@y = 1.5 cm  Wall boundary adiabatic wall (heatflux=0.0 W/m²)
  - Plane@z= -0.4 cm  Wall boundary: fixed temperature (1175 K≈902 °C)
  - Plane@z= 0.0 cm  Domain boundary: temperature & heat flux exchange between the domains

• The inflow rate on the inlet boundary is fixed $2.5\cdot10^{-7}$ kg/s, resulting in an inlet velocity of $1.82\cdot10^{-1}$ m/s. For the transient case, the temperature of the inflowing nitrogen gas is instantly lowered from 1175 K to 300 K. The bottom wall of the glass domain is changed to an adiabatic (zero heat flux) wall. For the top wall of the flow domain a time-dependent temperature is imposed. Starting from 1175 K (=902 °C) the temperature is lowered to 723.15 K (=450 °C) and then kept at this temperature.

• Output data:
  - A typical result at a certain time step is presented below. The temperature of the top of the glass substrate is shown in Figure 12. What can be seen is the effect of the cold nitrogen inlet on the left side of the domain. Also the effect of the circular temperature distribution at the top wall of the flow domain can be seen in the contour shapes, mainly caused by the non-uniform thermal radiation heat flux towards the gradually cooled top wall.

Figure 12: Temperature distribution on top of glass substrate

• The software operates under Windows or Linux.

• A commercial license is required.
• Continuum model of microstructural evolution, solved by MICRESS (microstructure evolution simulation software) provided by Access, Germany - a commercial software for the calculation of microstructure formation in time and space during phase transformations.
  ◦ Underlying equations are phase-field and diffusion equations.
  ◦ Unknowns to be solved are the local phase distribution and concentrations of components.
  ◦ Input data
    • Initial conditions
      • 40 x 20 x 50 finite differences grid with a grid spacing of 0.05 micrometres
      • 100% liquid: 0.39% copper, 0.22 % indium, 0.39 % selenium
    • Boundary conditions:
      • Temperature profile from macro scale
  ◦ Output data:
    • Averaged phase and concentration distribution
    • Local phase, concentration and temperature distribution.
  ◦ Operating systems: Windows/Linux
  ◦ Requires a commercial license.

• Homogenization dummy method. This part is under development.

3.3 Description of test setup
The simulation software packages, X-Stream and MICRESS have been installed on two separate servers (acsrvappmic1, acpcmst1) both located on the local network (hosted by Access). This is schematically presented in Figure 13. The individual applications are allocated by MuPIF platform job managers, running permanently on these servers. The top-level scenario is started on an Access client computer. That means all network communication and data exchange is done over local area network. In this scenario one X-Stream interface and four MICRESS interfaces are allocated and steered. Each MICRESS interface handles one single microstructure analysis for a given macro location. MuPIF version 0.11.14 and Pyro4 version 4.39 have been used for this test.
3.4 Results
It took some development cycles to get a stable test scenario running. There are a lot of sources for errors on which the top-level script and the interfaces have to react in fail safe way. This is for sure a challenge in the next project period.

However, the final test scenario was completed successfully on the platform. The scenario performs 10 time steps to evaluate the overall functionality of all framework components. All platform components and the remote connections were stable during test execution.

The results indicate a relative small overhead for initializing a remote execution scenario and communication between remote interface components (about 30 seconds mainly for file upload). The most time was really spent in the steered external simulations, i.e. about 6 seconds for an X-Stream time step and about 7-10 minutes for a MICRESS time step.
3.5 Internal WP3 evaluation

<table>
<thead>
<tr>
<th></th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection stability: 1(stable)-5(weak)</td>
<td>1</td>
</tr>
<tr>
<td>Data throughput: 1(fast)-5(slow)</td>
<td>1</td>
</tr>
<tr>
<td>API Implementation: 1(easy, clear) -5(difficult)</td>
<td>2</td>
</tr>
<tr>
<td>Overall platform performance: 1(excellent) – 5(bad)</td>
<td>2</td>
</tr>
<tr>
<td>Setting up the network: 1(simple) -5(difficult)</td>
<td>2</td>
</tr>
<tr>
<td>Problems encountered</td>
<td>Some minor bugs in the framework</td>
</tr>
<tr>
<td>Additional comments</td>
<td>It will be necessary to improve the stability of interfaces and their recovering in case of errors.</td>
</tr>
</tbody>
</table>

4 Conclusions

The successfully executed distributed prototypic scenarios in WP2 and WP3 proved the capability of the platform to operate in distributed environments. Overall, the performance and stability of the platform were evaluated positively. Evaluation, however, revealed also some open problems. The setup of distributed workflows can be more difficult when multiple partners are involved. This is not due to the limitation of the platform, but rather the consequence of a strict IT policy enforced by individual IT departments, not allowing any (even secure) incoming connections. This can be resolved by setting up at least one computer outside intranets accepting incoming SSH connections, so that all the communication can be tunnelled using forward and reverse SSH tunnels via this node. The testing runs revealed some issues and provided a lot of feedback for future improvements. Based on the successful demonstration of platform ability to run prototypic scenarios we consider the milestone as reached.