UNIVERSITY OF AMSTERDAM

UNIVERSITÉ DE GENÈVE

Joris Borgdorff*, Jean-Luc Falcone, Eric Lorenz, Bastien Chopard, and Alfons G. Hoekstra

A principled approach to distributed multiscale computing

from formalization to execution

Introduction

- Multiscale systems
  - are inherently complex
- Multiscale models
  - benefit from single scale decomposition both conceptually and computationally
  - can require significant computing resources
  - few general multiscale computing frameworks for e-Infrastructure

Aims

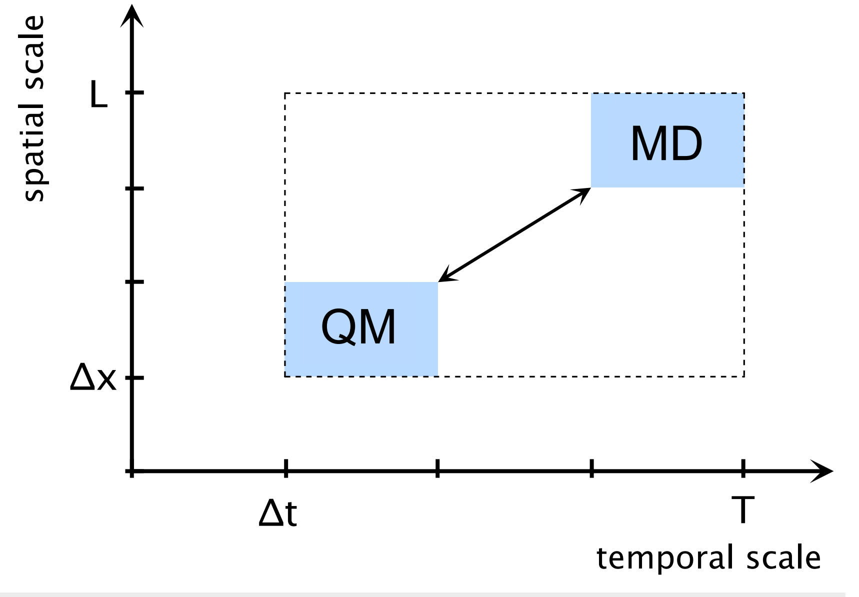
- Present a single methodology on building multiscale models
- Introduce tools that formalize running a multiscale model
- Couple the methodology with a distributed computing environment

Overview

- Modeling
  - Functional decomposition
  - Coupling topology
- Automation
  - Specification
  - Analysis

Model description

- Submodels
  - single scale, not aware of other submodels
- Couplings
  - uni-directional interaction
  - *where* in the submodel to couple?



Scale separation map (SSM)

Submodel Execution Loop

$t \leftarrow t_0$

$f \leftarrow \mathbf{f_{init}}$

while $t - t_0 < T$ *do*

  $\mathbf{O_i}$

  $t \leftarrow t + \Delta t$

  $f \leftarrow \mathbf{S}$

end

$\mathbf{O_f}$

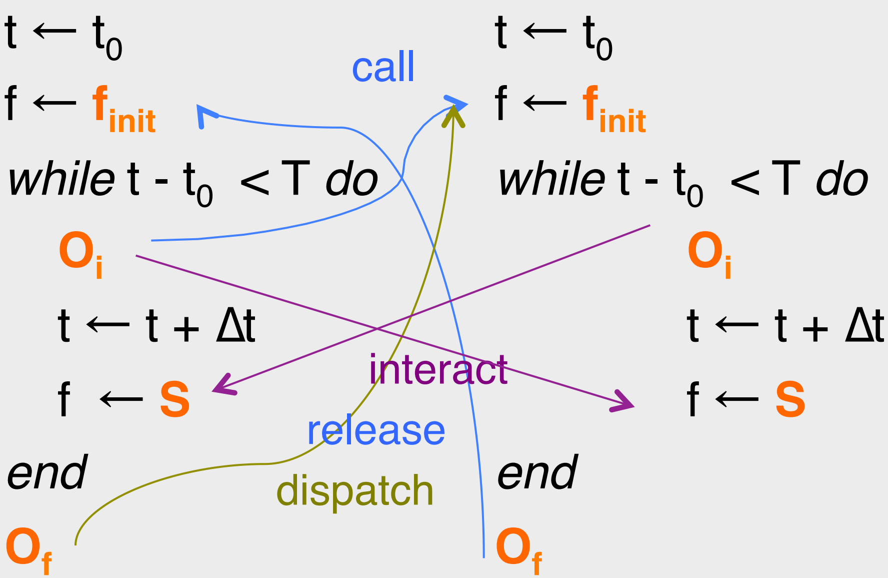
- Fully specify and limit submodel behavior:
  - iterative
  - implement 5 operators
    - initialization $\mathbf{f_{init}}$
    - intermediate observation $\mathbf{O_i}$
    - solving step $\mathbf{S}$
    - final observation $\mathbf{O_f}$
- Time step Δt constrained by temporal scale

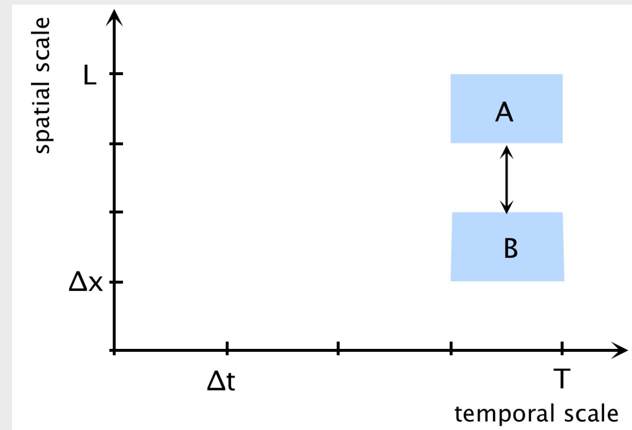
Coupling templates

- Operators O_i and O_f may send observations
- Operators f_{init} and S may receive
- Specify coupling between submodels A and B using operators

	Name	*Coupling template*
1.	interact	$O_i^A \rightarrow S^B$
2.	call	$O_i^A \rightarrow f_{init}^B$
3.	release	$O_f^B \rightarrow S^A$
4.	dispatch	$O_f^A \rightarrow f_{init}^B$

Coupling templates





	Name	*Coupling template*
1.	interact	$O_i^A \rightarrow S^B$
2.	call	$O_i^A \rightarrow f_{init}^B$
3.	release	$O_f^B \rightarrow S^A$
4.	dispatch	$O_f^A \rightarrow f_{init}^B$

Model description

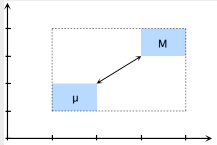
- Submodels
- Couplings
  - *where* in the submodel to couple?
    - Coupling templates
- Full network
  - *how many* submodels are instantiated?
  - *which* instances are they coupled to?

Coupling topology

- Define *coupling topology*: graph of the coupling of a multiscale model
- Exactly represents all couplings and submodel instances
  - In our example, each molecule might require its own quantum dynamics submodel instance.

Overview

SSM



↓

Coupling topology

↓

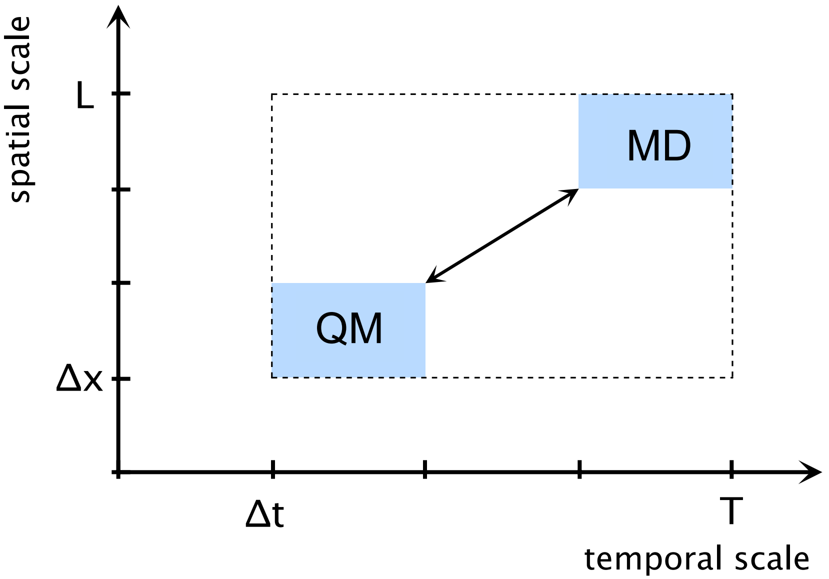
- ✓ Modeling
  - ✓ Functional decomposition
  - ✓ Coupling topology
- Automation
  - □ Specification
  - □ Analysis

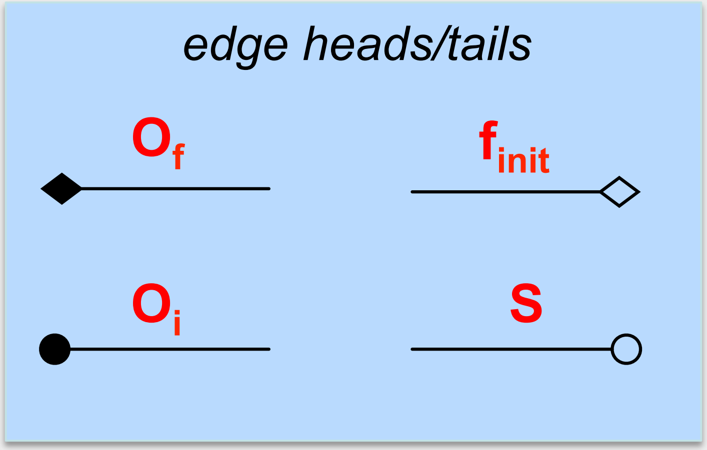
Specification

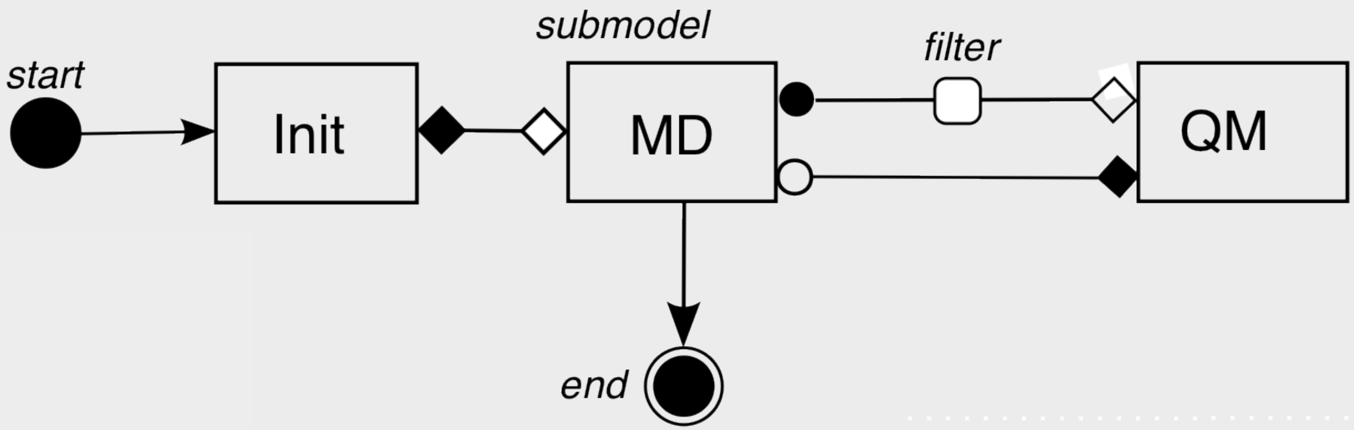
- Multiscale Modeling Language (MML)[1]
- A formal high-level specification language, constraining possible behavior of its elements, including
  - facilities for data manipulation;
  - information about implementation
- For execution, verification, and analysis
- Full XML specification: xMML

1. J.-L. Falcone *et al. Procedia Computer Science.* 1(1):819–826, 2010

Specification: macro-micro







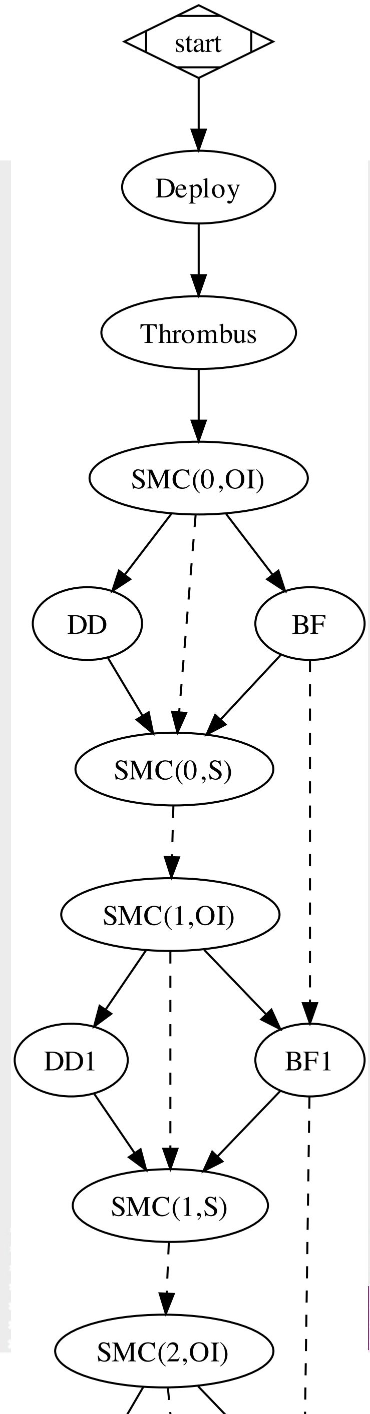
xMML: XML format for MML

```
<model id="ISR3D" name="In-stent restenosis 3D" xmml_version="0.3.3"
         xmlns="http://www.mapper-project.eu/xmml" xmlns:xi="http://www.w3.org/2001/Xinclude">
    <description>The motion of molecules and their quantum-dynamical interactions.</description>
    <definitions>
        <xi:include href="isr_meta.xml#xpointer(/metadata/*)"/>
        <submodel id="MD" name="Molecular Dynamics">
            <timescale delta="1E-7" max="1"/>
            <spacescale delta="1 nm" max="100 nm" dimensions="2"/>
            <spacescale delta="1 nm" max="5 nm"/>
            <ports>
                <in id="atomDyn" operator="S" datatype="atomDynamics"/>
                 <out id="atomPos" operator="Oi" datatype="atomPositions"/>
            </ports>
        </submodel> ... </definitions>
    <topology>
      <instance id="qm" submodel="QM"/>
      <instance id="ic" submodel="INIT"/>
      <instance id="md" submodel="MD"/>

      <coupling name="initPos" from="ic.atomPos"to="md.atomPos"/>
      <coupling name="atomPos" from="md.atomPos" to="qm.atomPos">
         <apply filter="normalizeSpace"/>
      </coupling>
      <coupling name="atomDyn" from="qm.atomDyn" to="md.atomDyn"/>
</topology> </model>
```

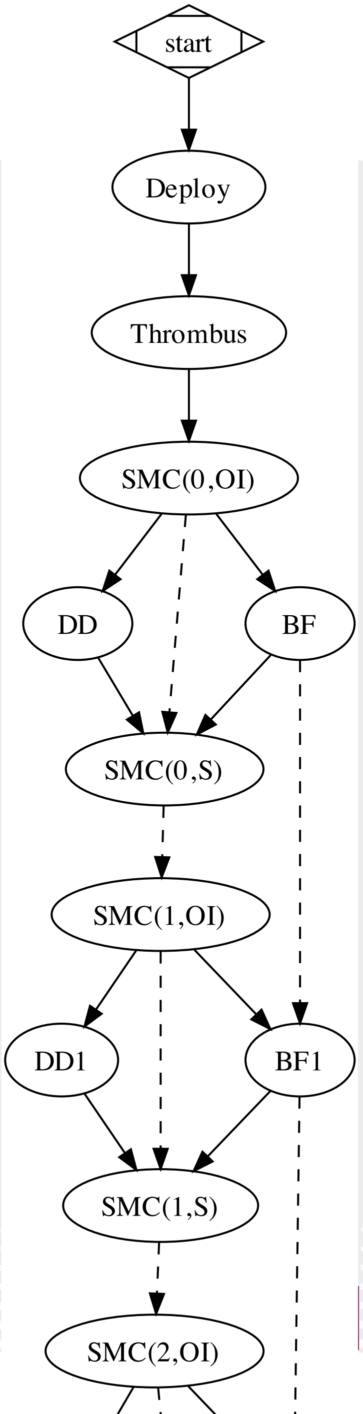
Task graph

- Determine execution order based on xMML and the SEL: task graph
  - Directed acyclic graph
  - Schedule submodels based on dependencies
  - Estimate run time and communication costs
  - Detect deadlocks
  - SEL crucial to the ordering



Task graph

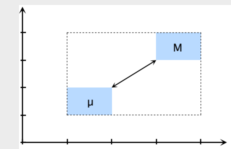
- Operators **O_i** and **S**/**B** and each iteration in a separate node
  - Transformation to reduce nodes is possible and has been performed here
- Edges based on communication
  - they indicate dependencies
  - dashed edges are stateful transitions
    - same instance could be scheduled on a different machine



Overview

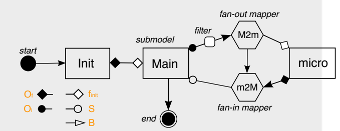
- ✓ Modeling
  - ✓ Functional decomposition
  - ✓ Coupling topology
- ✓ Automation
  - ✓ Specification
  - ✓ Analysis

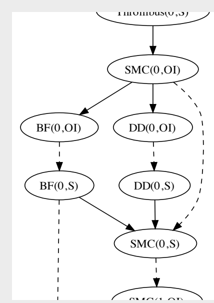
SSM



Coupling topology

MML





Task graph

Distributed computing

- Handle resources that must be scheduled
- Minimize communicational overhead
- Minimize run-time dependencies (and cross-scheduling)
- etc…
- If an MML specification (and thus a task graph) is available then educated guesses on schedules can be made

Conclusions

- By using well-defined foundations we have a general approach to take multiscale models to a runtime environment.
- MML offers:
  - coupling topology, task graph, and execution description
- MML is useful for:
  - Middleware, tools, and application developers
- Future work: real distributed computing!

Acknowledgements

- The **MAPPER** project✢
  - funded by EC's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° RI-261507.
- Carles Bona Casas from the **University of Amsterdam**
- Katarzyna Rycerz, Tomasz Gubala, Daniel Harezlak, Eryk Ciepiela from **Cyfronet**

✢ http://www.mapper-project.eu/

University of Amsterdam
Université de Genève

**Joris Borgdorff*, Jean-Luc Falcone, Eric Lorenz,
Bastien Chopard, and Alfons G. Hoekstra**

A principled approach to distributed multiscale computing

Questions?