OpenMOLE: a grid enabled workflow platform

Romain REUILLON, Mathieu LECLAIRE
romain.reuillon@openmole.org
mathieu.leclaire@openmole.org
www.openmole.org

Uppsala - 15th of April, 2010

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Hopefully, categories of problems exhibit naturally **parallel aspects**:

- design of experiments,
- genetic algorithms,
- stochastic simulations with many replications,
- ...

Introduction

Even though computational grids is a solution to compute problems unsolvable otherwise, grid usage is widely established a tricky domain.

Hopefully, categories of problems exhibit naturally **parallel aspects**:

- design of experiments,
- genetic algorithms,
- stochastic simulations with many replications,
- ...

=> It is possible to develop a **generic software framework** for distributing naturally parallel application.

Outline

1. Genesis of OpenMOLE
2. Central concepts
3. Task delegation
4. Detailed example
5. Scientific projects

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

Central concepts;

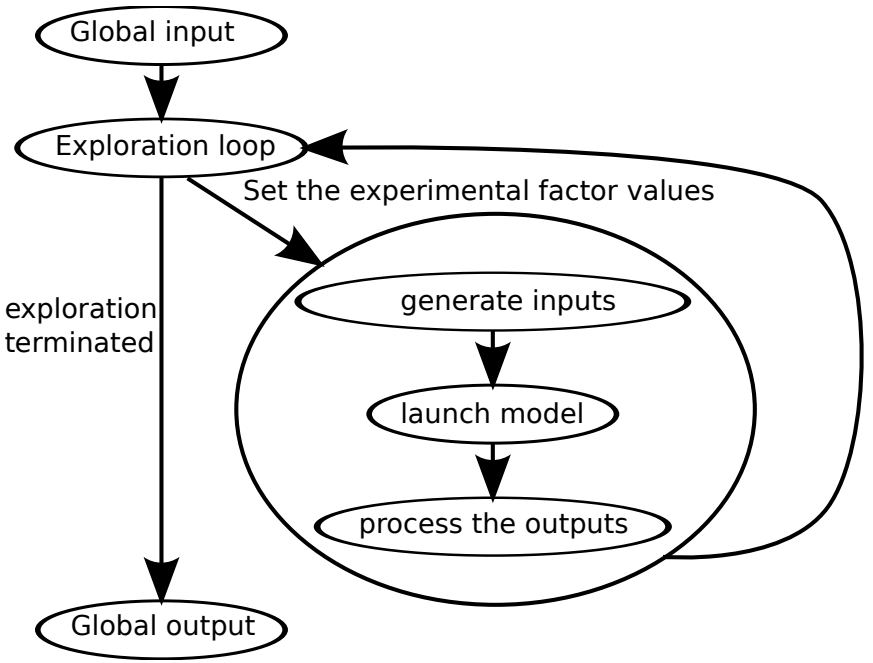
- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.

From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

Central concepts;

- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.

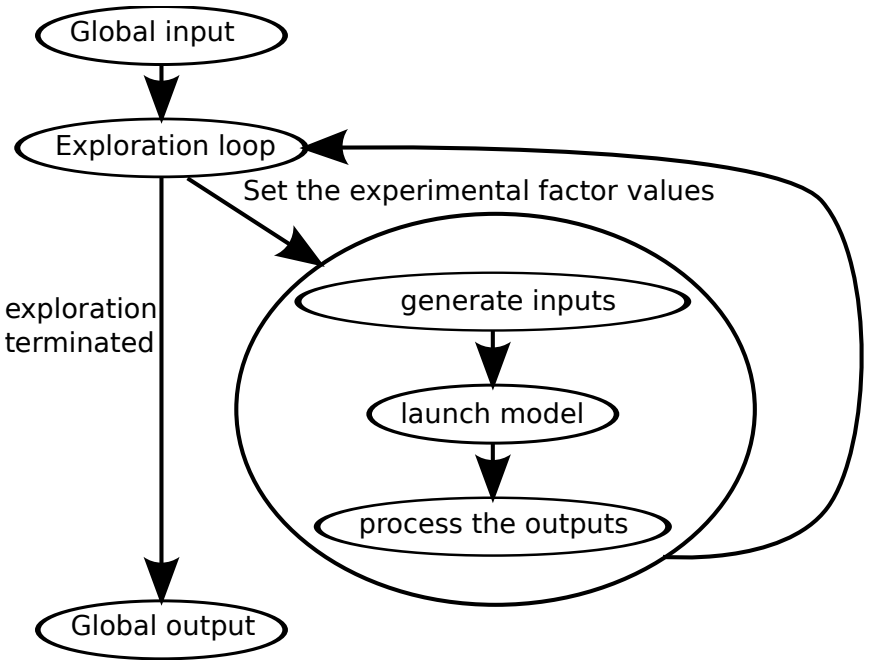


From a sequential approach...

OpenMOLE was originally a platform for experimenting on models on a single computer (SimExplorer).

Central concepts;

- tasks (software components),
- sequence of tasks,
- loops on a sequence of tasks.



These concepts were inherently **sequential !**

... to a distributed approach.

The project was redesigned from scratch in October 2008.

... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

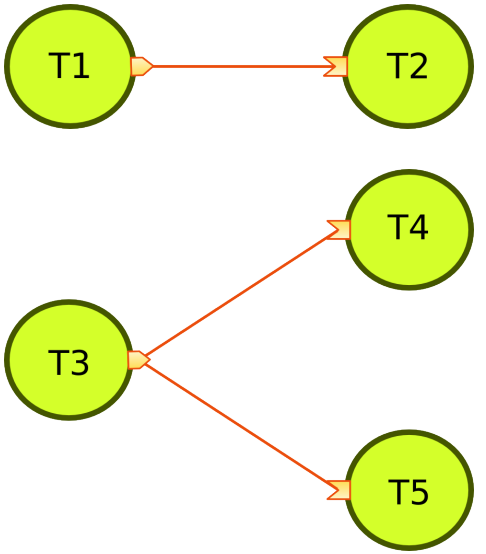
- sequential relationships between tasks where needed,

... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

- sequential relationships between tasks where needed,
- naturally parallel representations of algorithms.

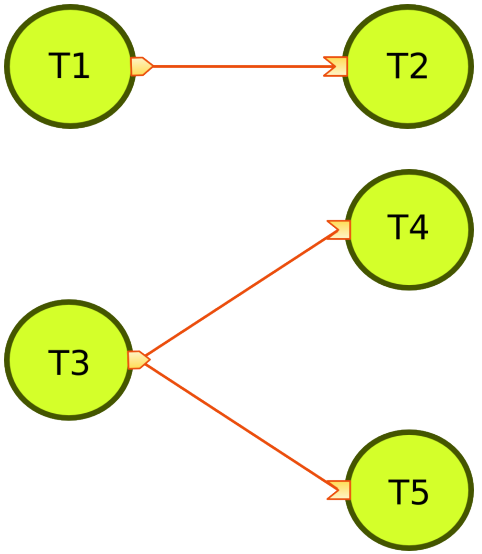


... to a distributed approach.

The project was redesigned from scratch in October 2008.

A workflow approach was chosen because it provides both:

- sequential relationships between tasks where needed,
- naturally parallel representations of algorithms.



Untie distributed algorithms and execution environments = environment agnostic distributed algorithms.

Outline

1 Genesis of OpenMOLE

2 Central concepts

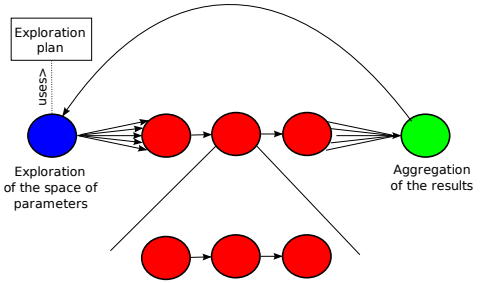
3 Task delegation

4 Detailed example

5 Scientific projects

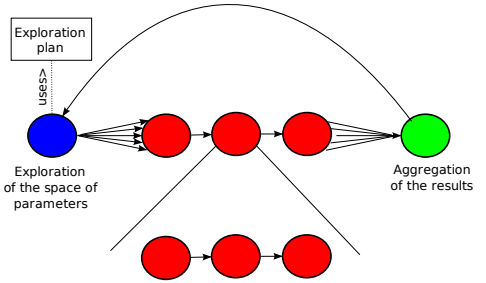
OpenMOLE:

- allows **the definition of complex workflows** for model exploration,



OpenMOLE:

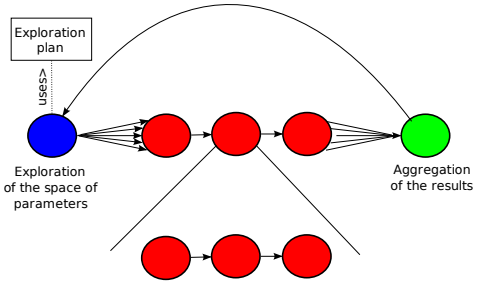
- allows **the definition of complex workflows** for model exploration,



- **manages the execution** of these workflows on the user desktop PC,

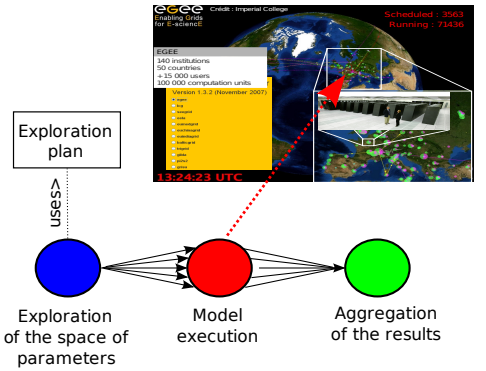
OpenMOLE:

- allows **the definition of complex workflows** for model exploration,



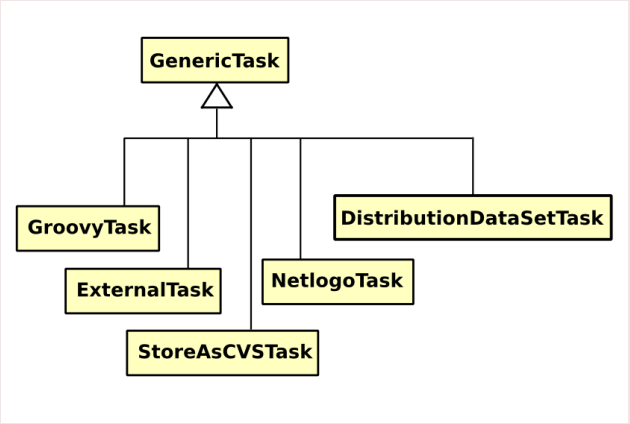
- **manages the execution** of these workflows on the user desktop PC,

- provides **transparent delegation mechanisms** of part of these workflow's execution **to distributed environments** (grids, clusters, ssh servers).



Tasks

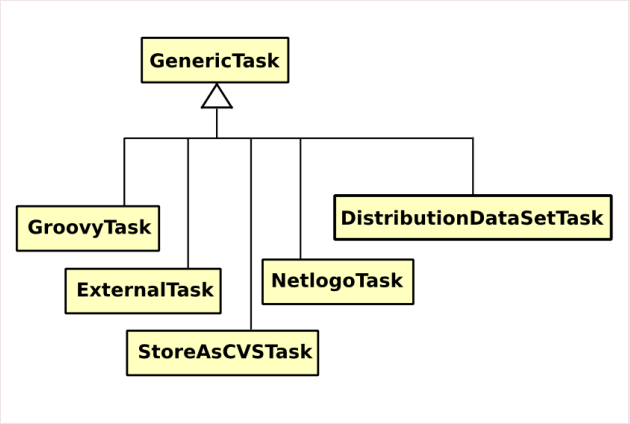
Definition



A Task is a polymorphic class that embed a work to be achieved. Tasks are portable and can be executed on any execution environment.

Tasks

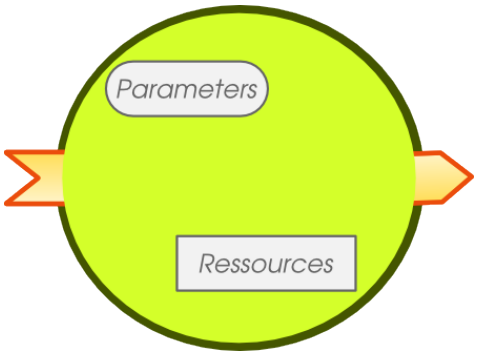
Definition



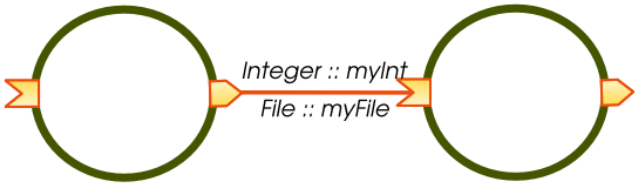
A Task is a polymorphic class that embed a work to be achieved. Tasks are portable and can be executed on any execution environment.

Integrating a Task in workflow requires:

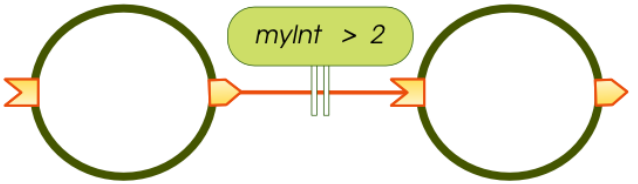
- parameterizing it (providing a Groovy script, a Netlogo script, a command line for launching a model...)
- optionally associating resources with it (libraries, data files, executable files...)



Transitions

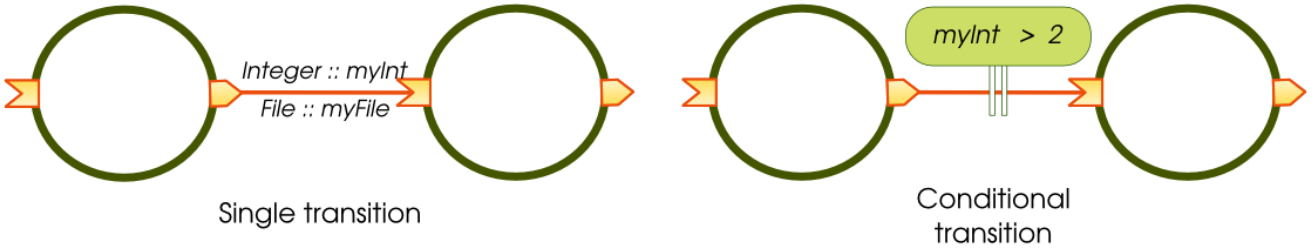


Single transition

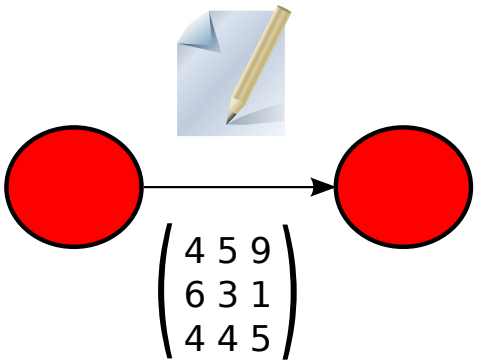


Conditional transition

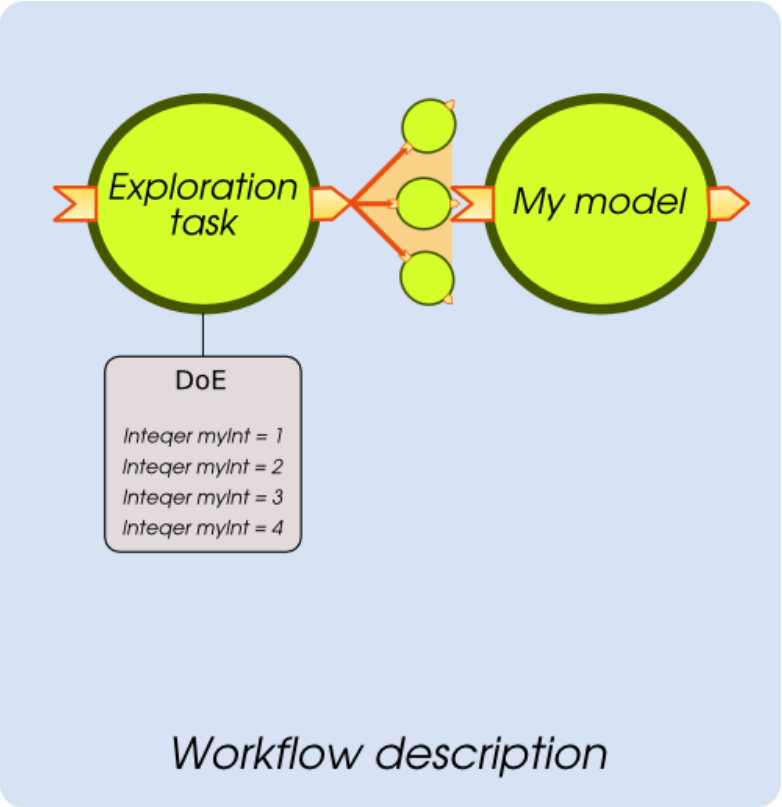
Transitions



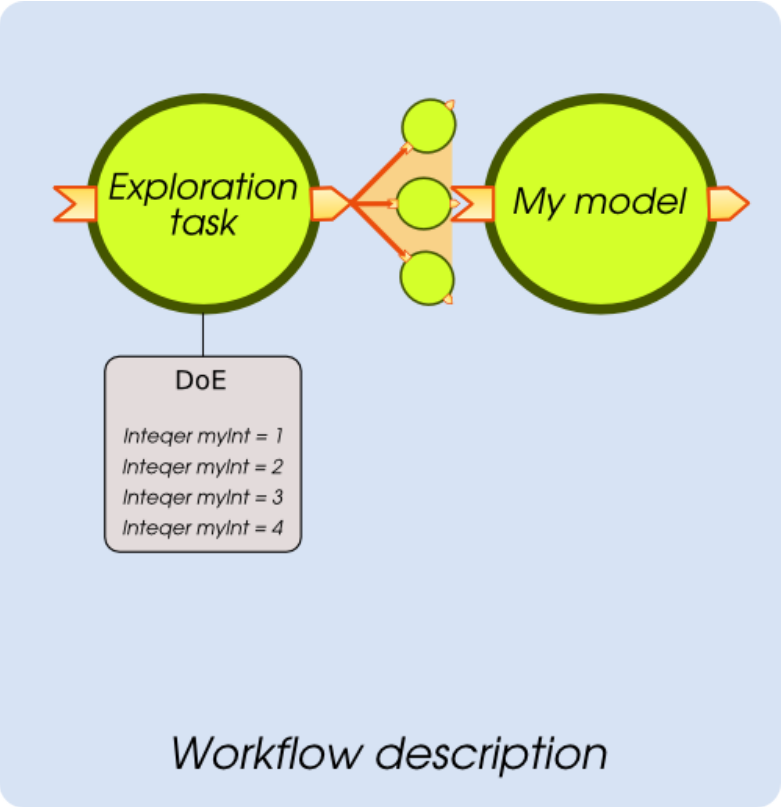
OpenMOLE **manages data and file transfers** between the workflow tasks (executed locally, or remotely on an execution node of a cluster or a grid)

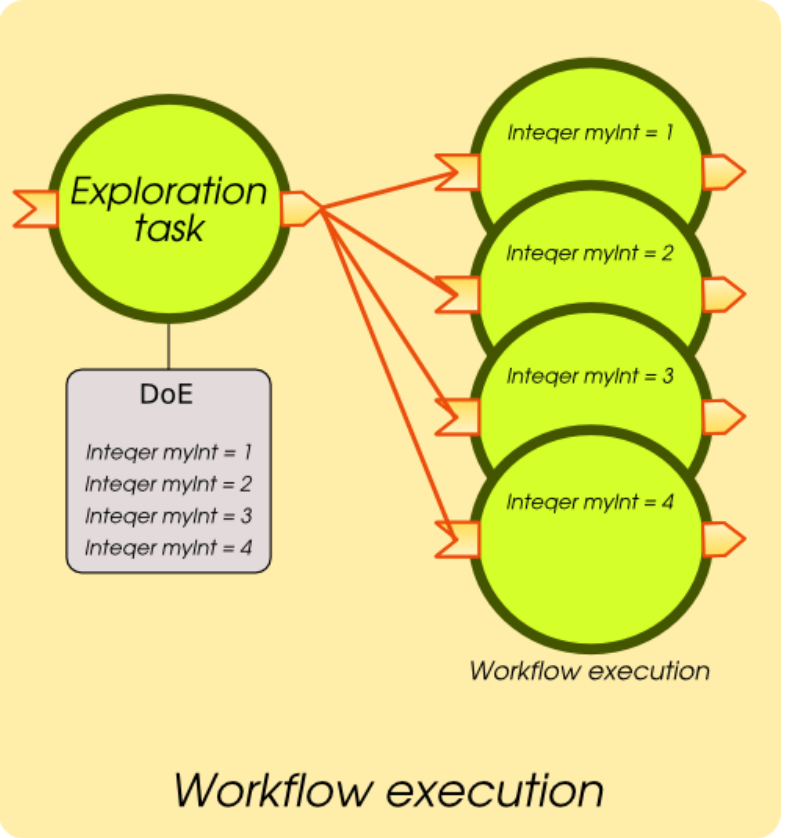


Exploration transition

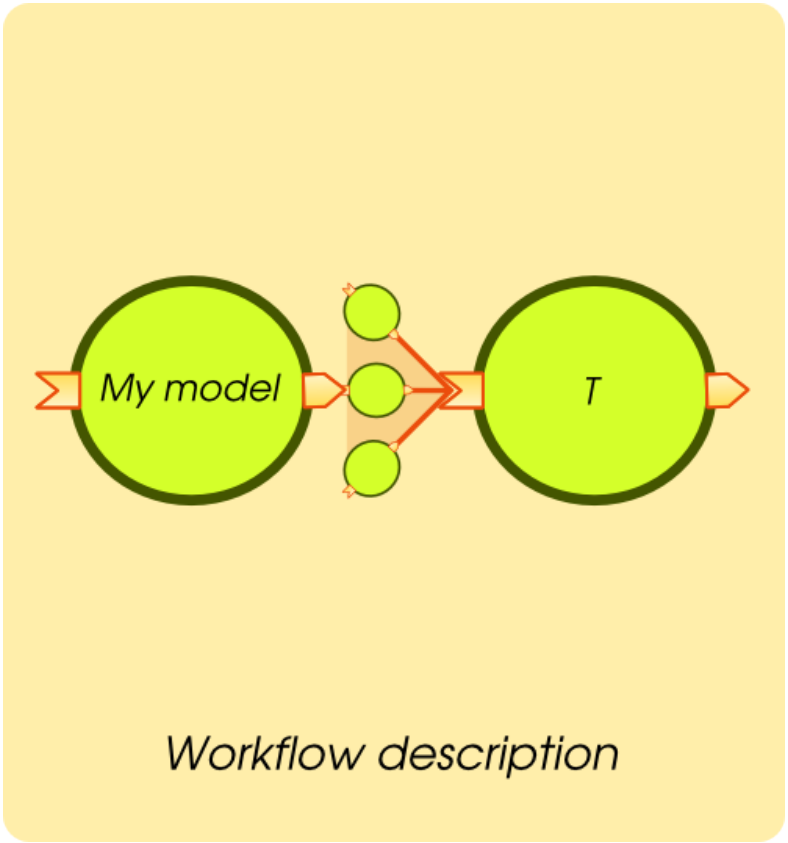


Exploration transition

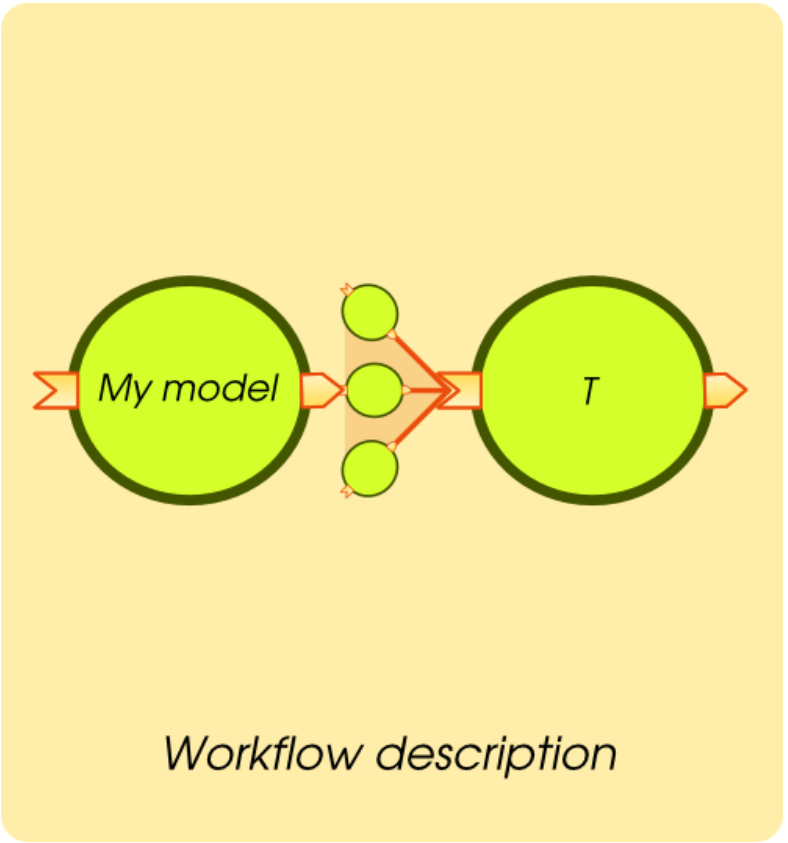


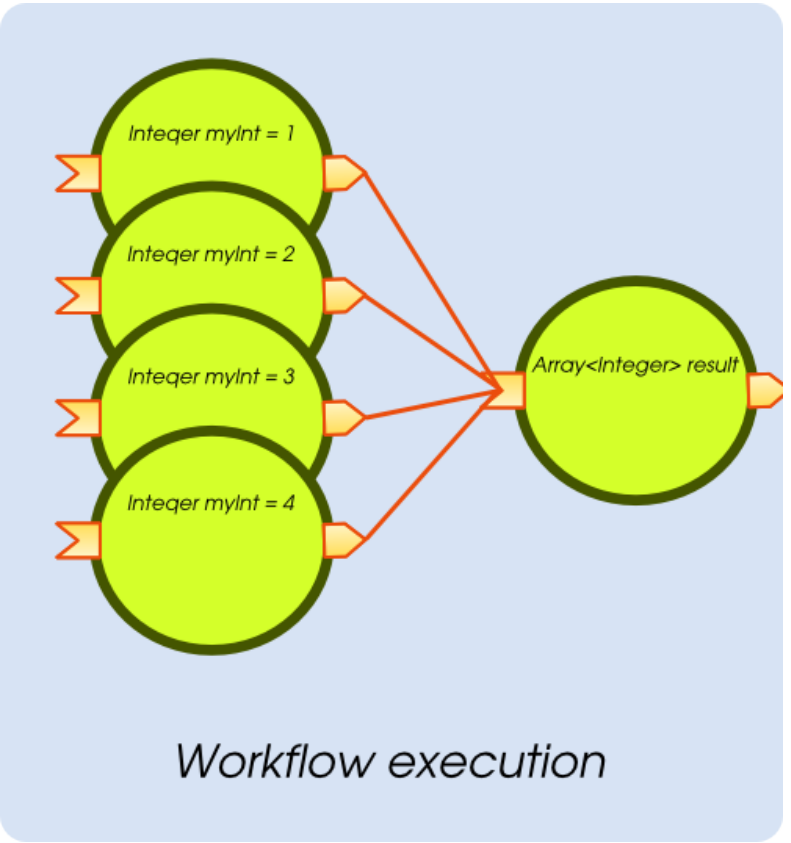


Aggregation transition



Aggregation transition





Outline

1 Genesis of OpenMOLE

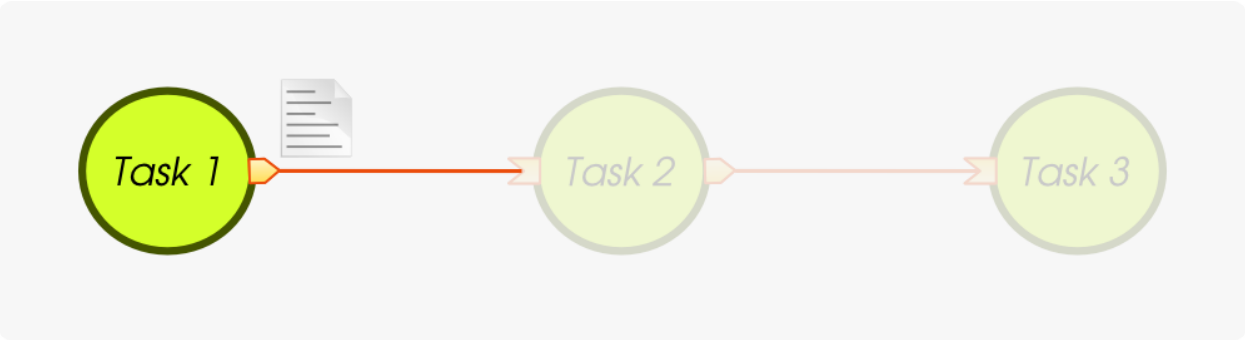
2 Central concepts

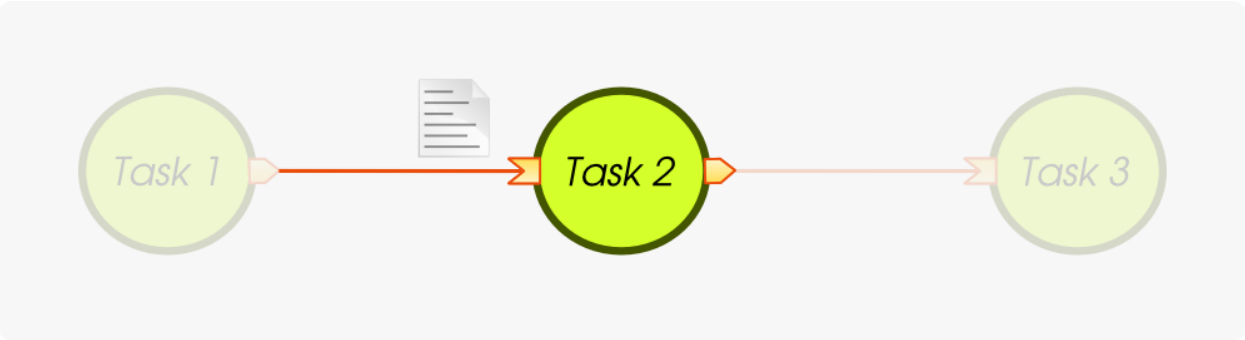
3 Task delegation

4 Detailed example

5 Scientific projects

Task 1
Task 2
Task 3


Task 1
Task 2
Task 3


Task 1
Task 2
Task 3

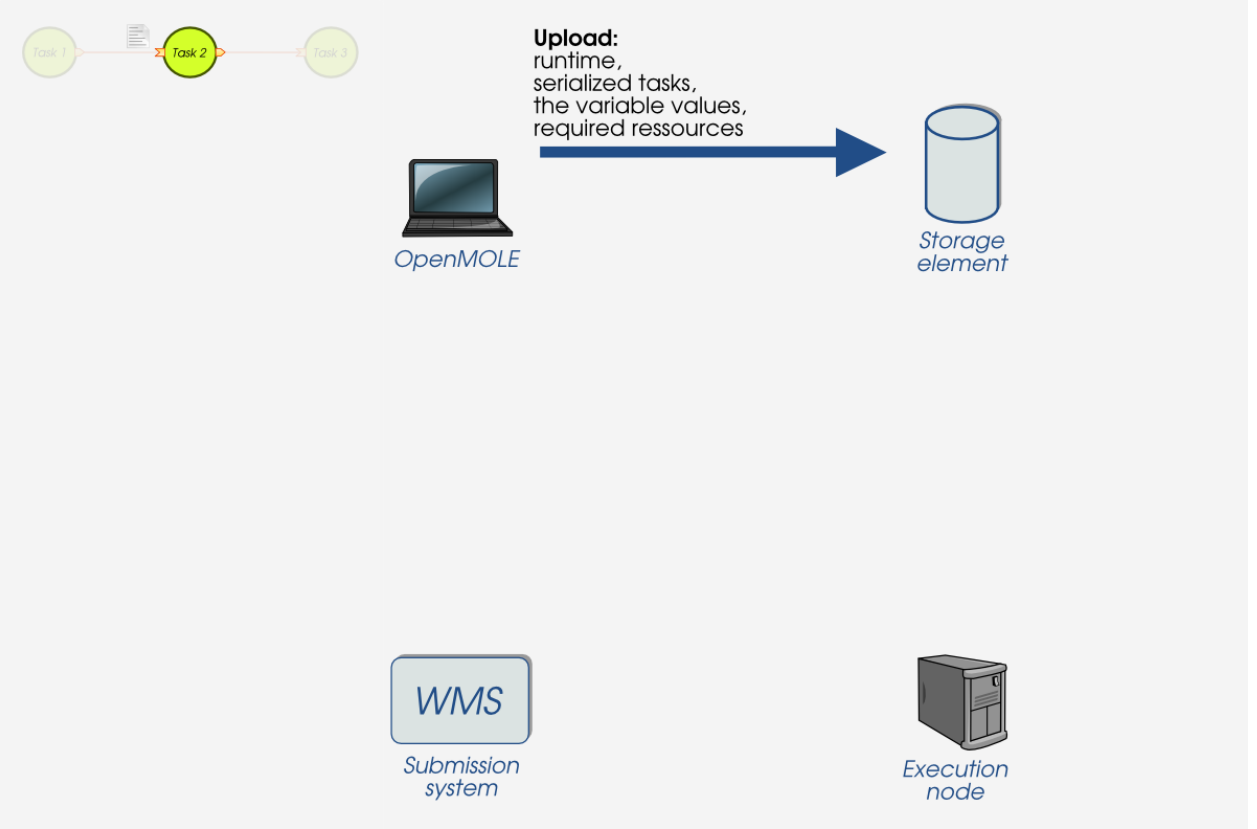
OpenMOLE

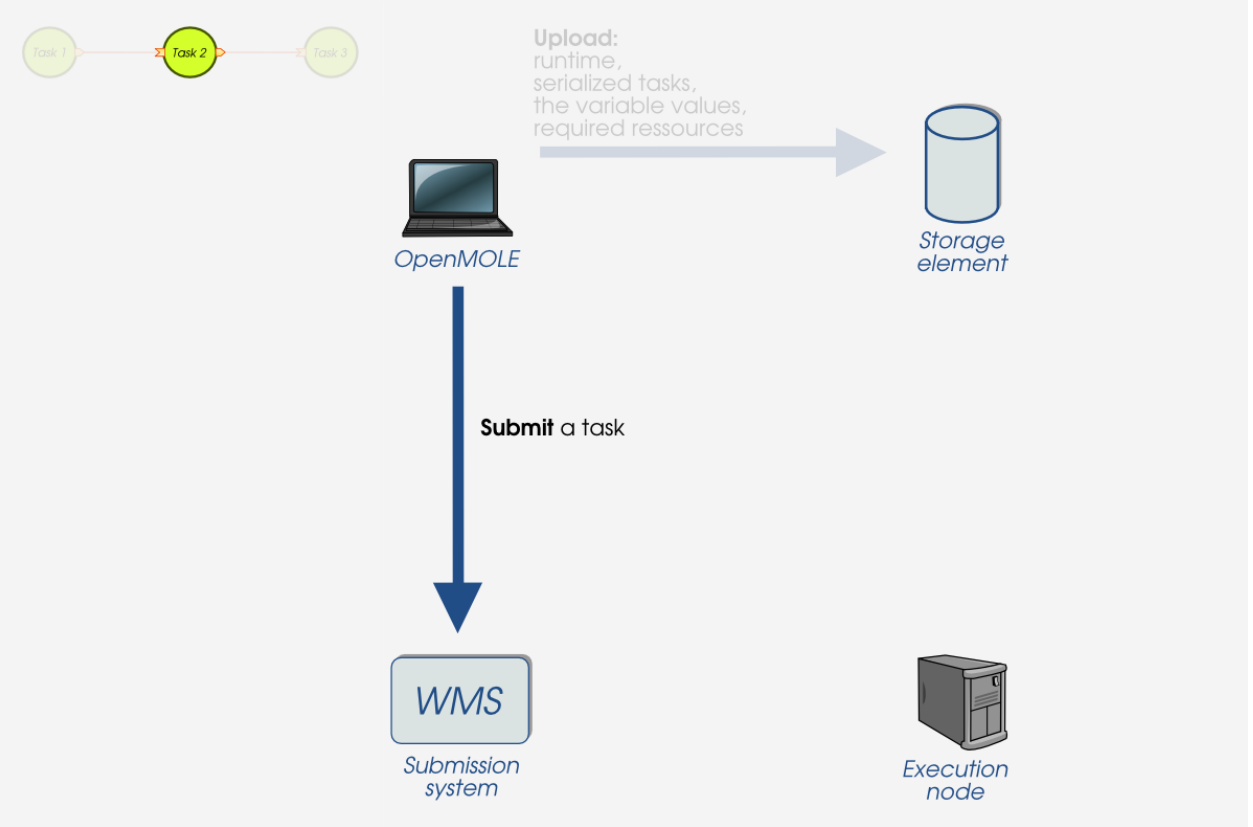
Storage element

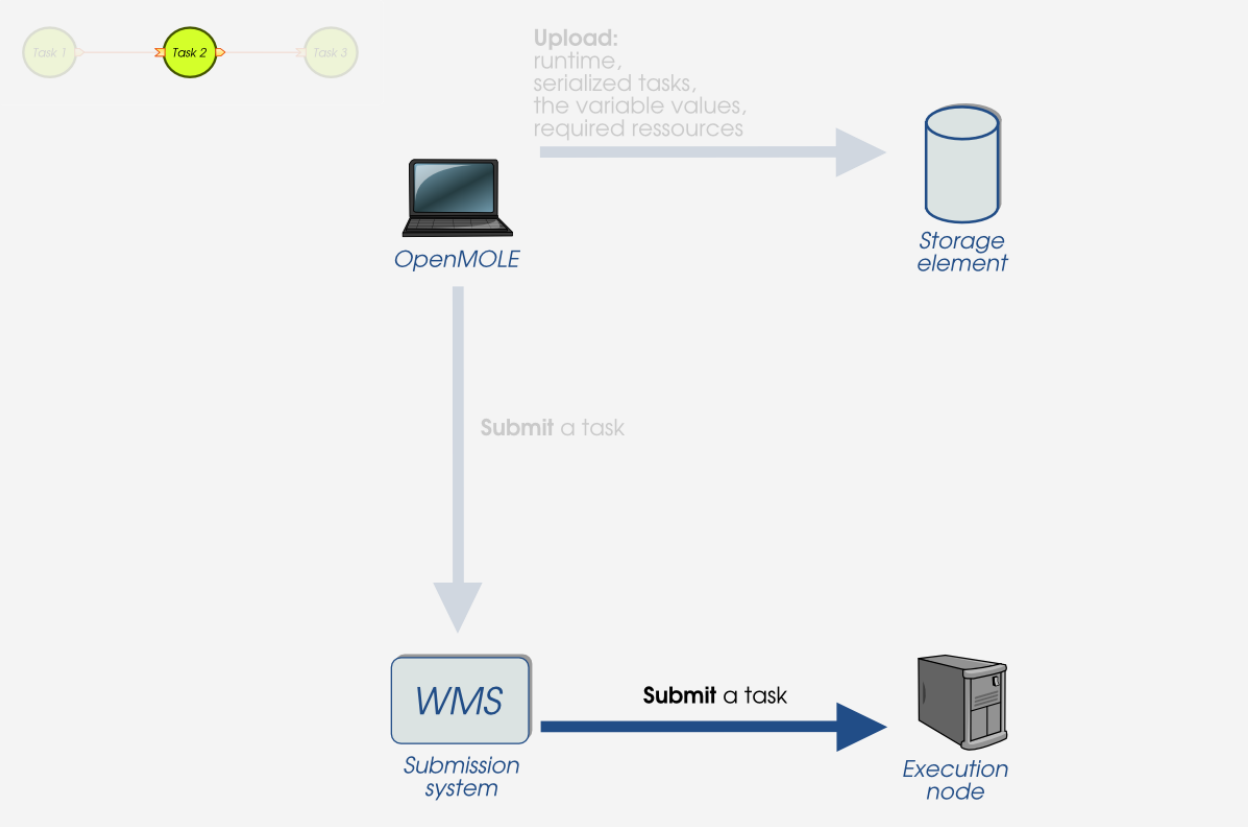
WMS

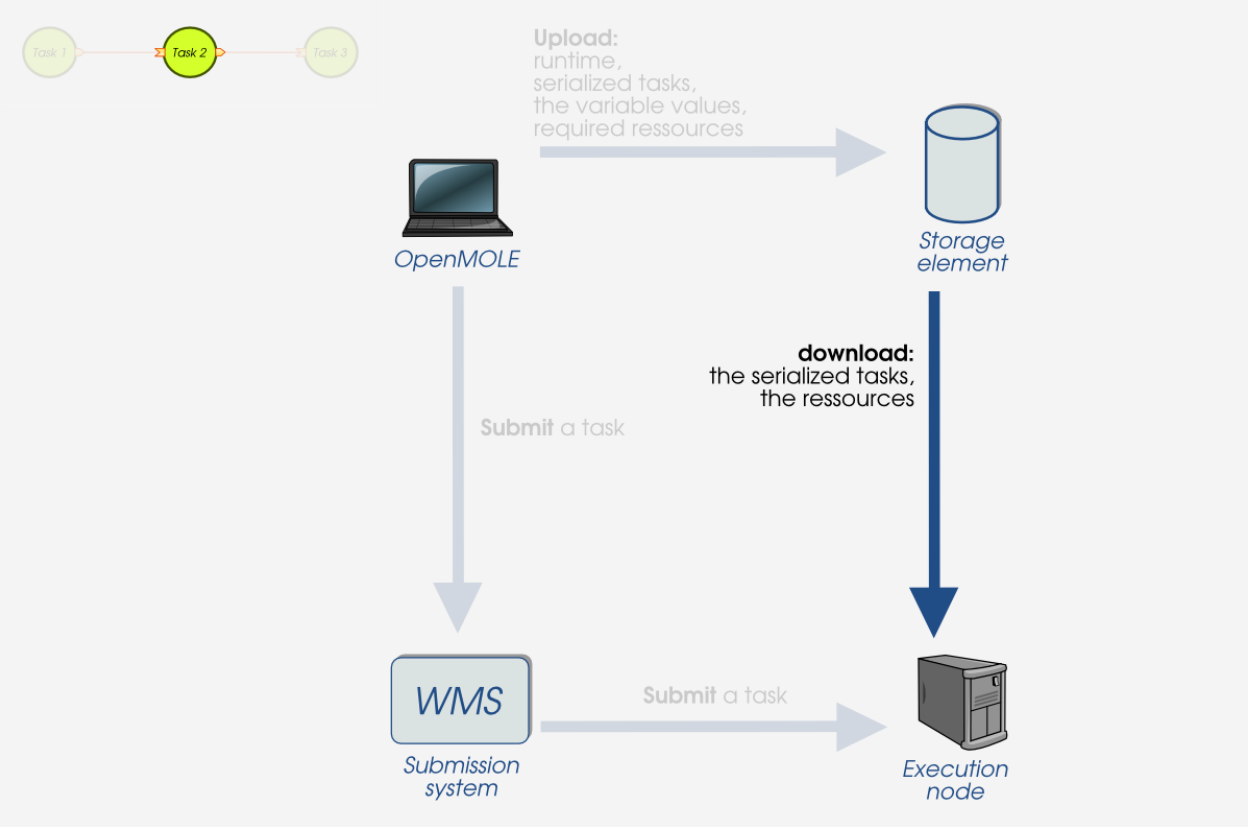
Submission system

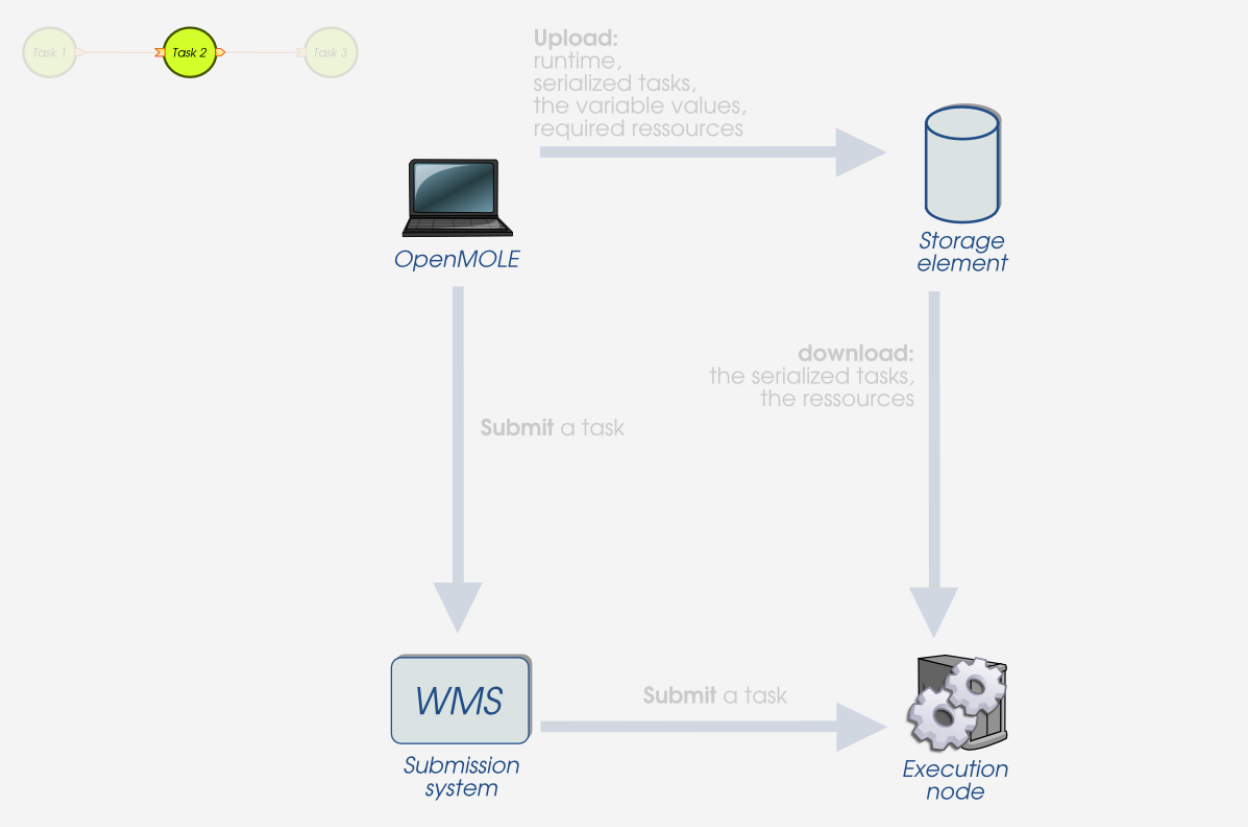
Execution node

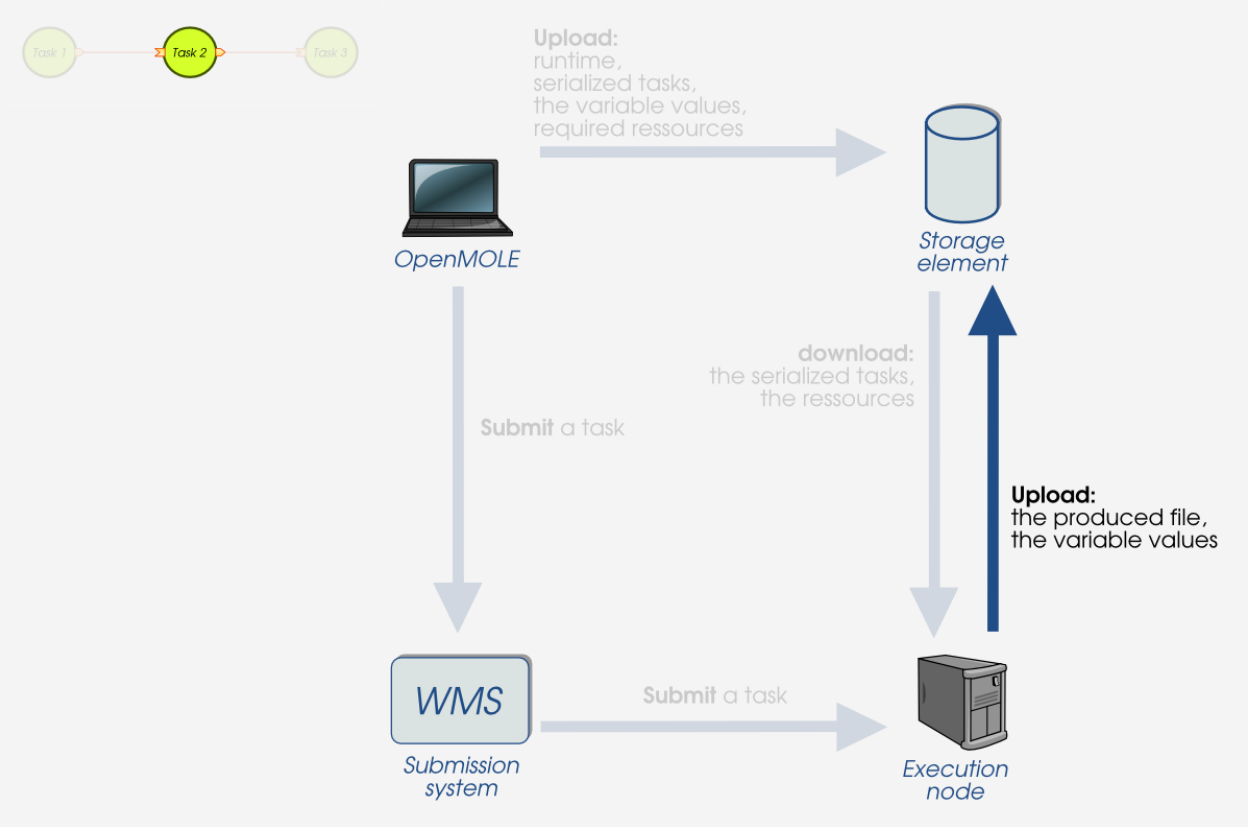

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
WMS
Submission system
Execution node

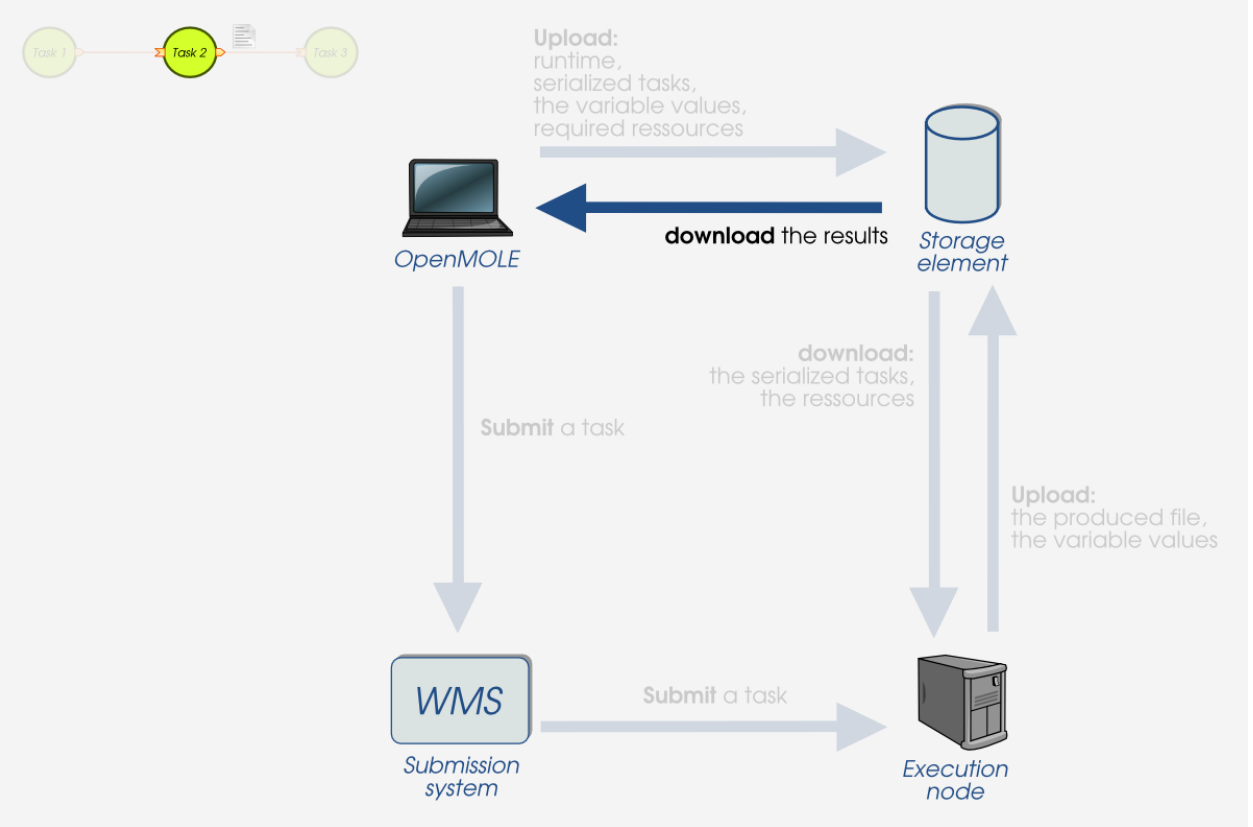

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
Submit a task
WMS
Submission system
Execution node

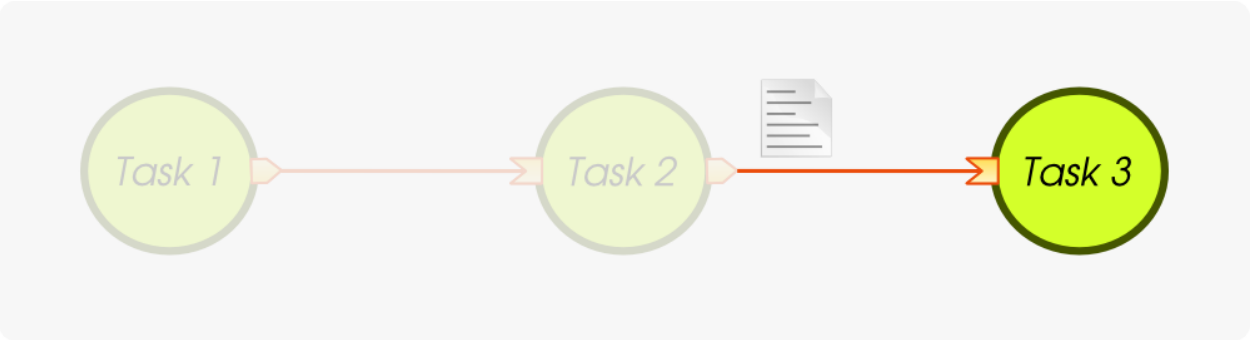

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
Submit a task
WMS
Submission system
Submit a task
Execution node

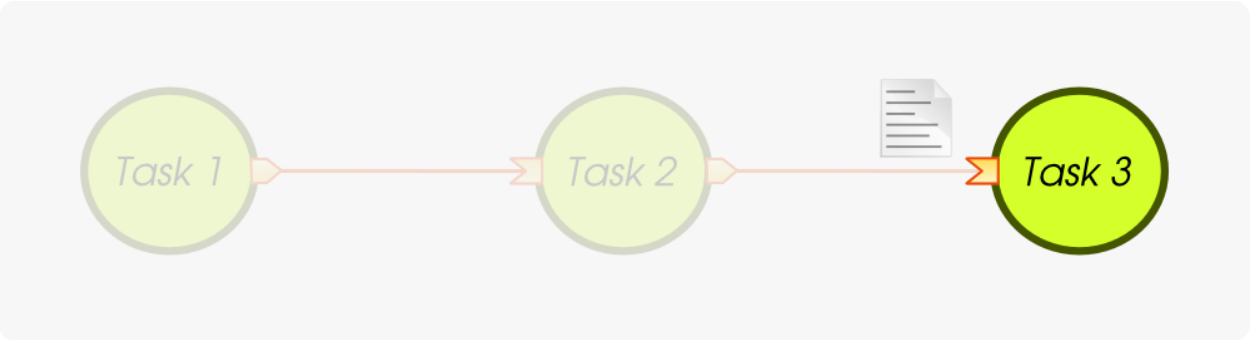

Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
WMS
Submission system
Submit a task
Execution node


Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
WMS
Submission system
Submit a task
Execution node


Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
Upload:
the produced file,
the variable values
WMS
Submission system
Submit a task
Execution node


Task 1
Task 2
Task 3
Upload:
runtime,
serialized tasks,
the variable values,
required ressources
download the results
OpenMOLE
Storage element
download:
the serialized tasks,
the ressources
Submit a task
Upload:
the produced file,
the variable values
WMS
Submission system
Submit a task
Execution node


Task 1
Task 2
Task 3


Task 1
Task 2
Task 3

Outline

1 Genesis of OpenMOLE

2 Central concepts

3 Task delegation

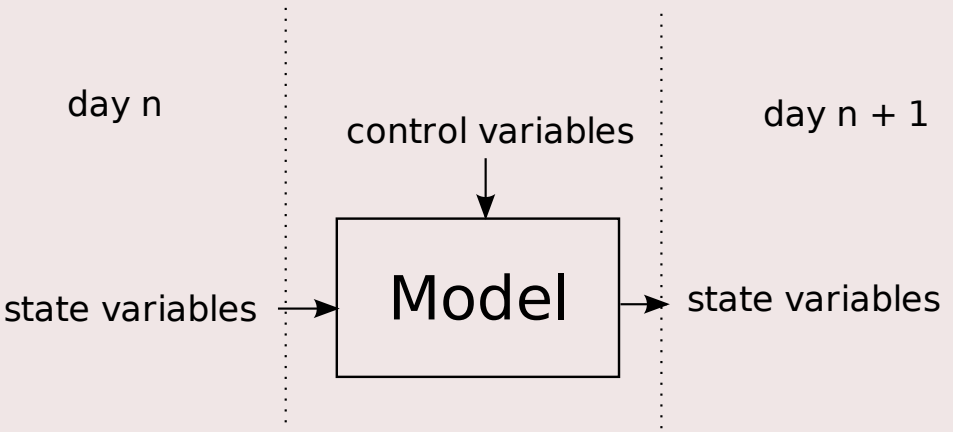
4 Detailed example

5 Scientific projects

The viability algorithm

The model

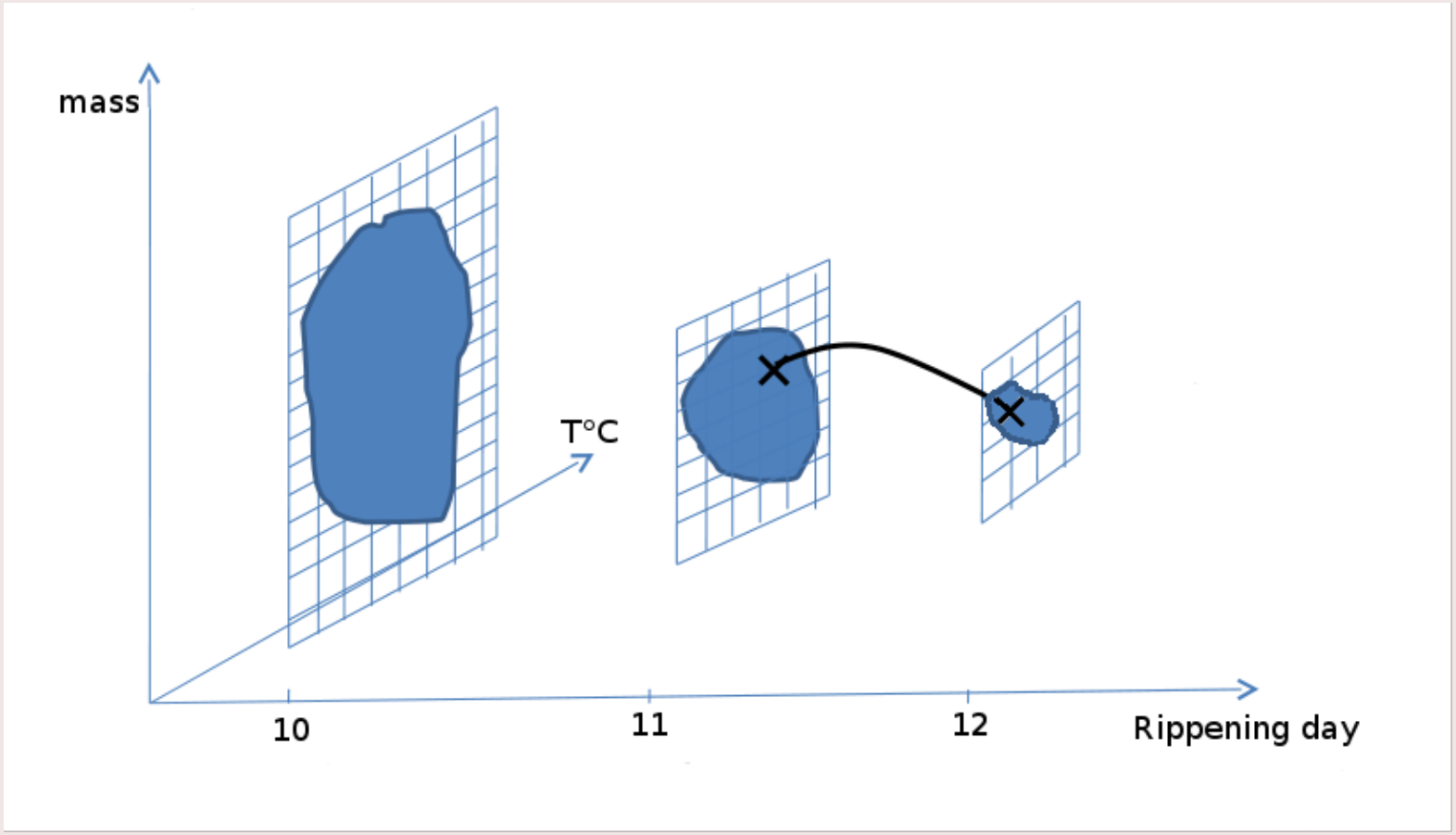
Controllable model of Camembert cheese ripening:

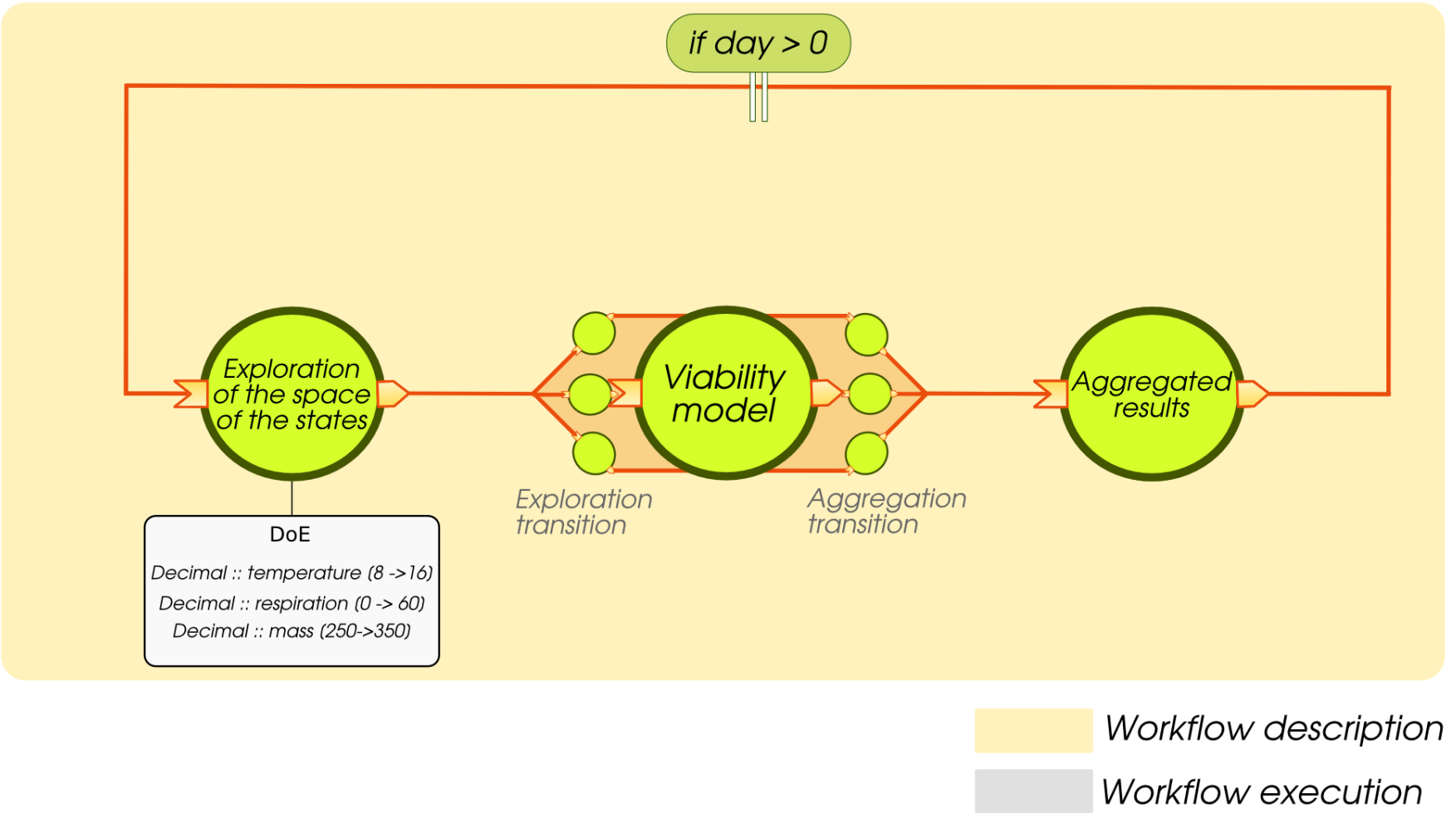


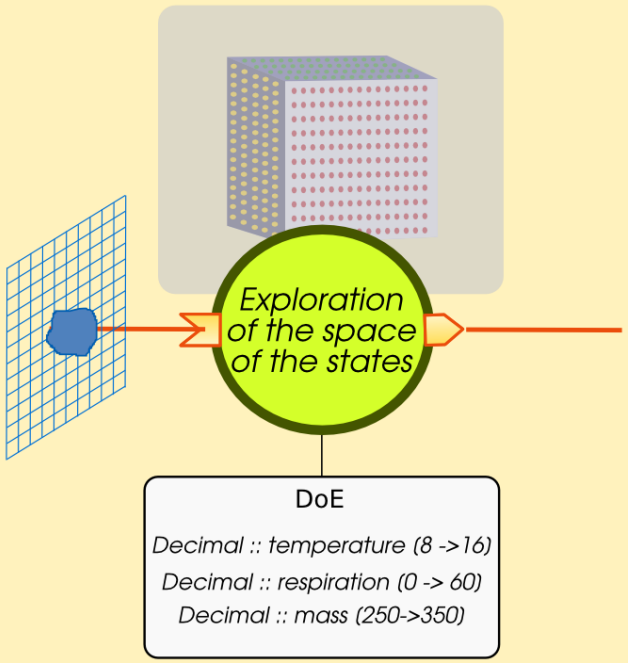
State = mass, micro-organisms activity and surface temperature
Control = temperature and humidity in the ripening chamber.

The viability algorithm

The tube computation

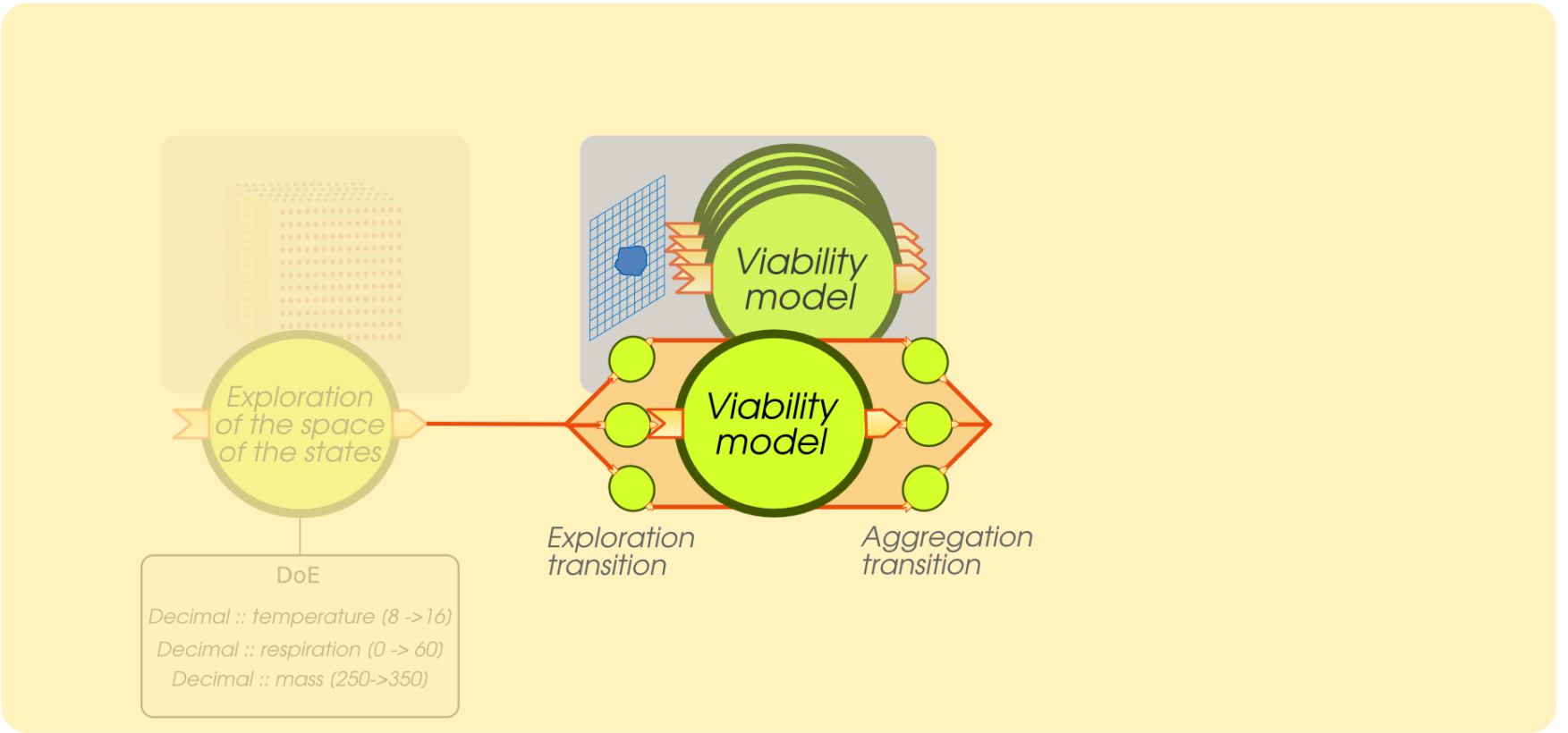



if day > 0
Exploration of the space of the states
Viability model
Aggregated results
Exploration transition
Aggregation transition
DoE
Decimal :: temperature (8 ->16)
Decimal :: respiration (0 -> 60)
Decimal :: mass (250->350)
Workflow description
Workflow execution



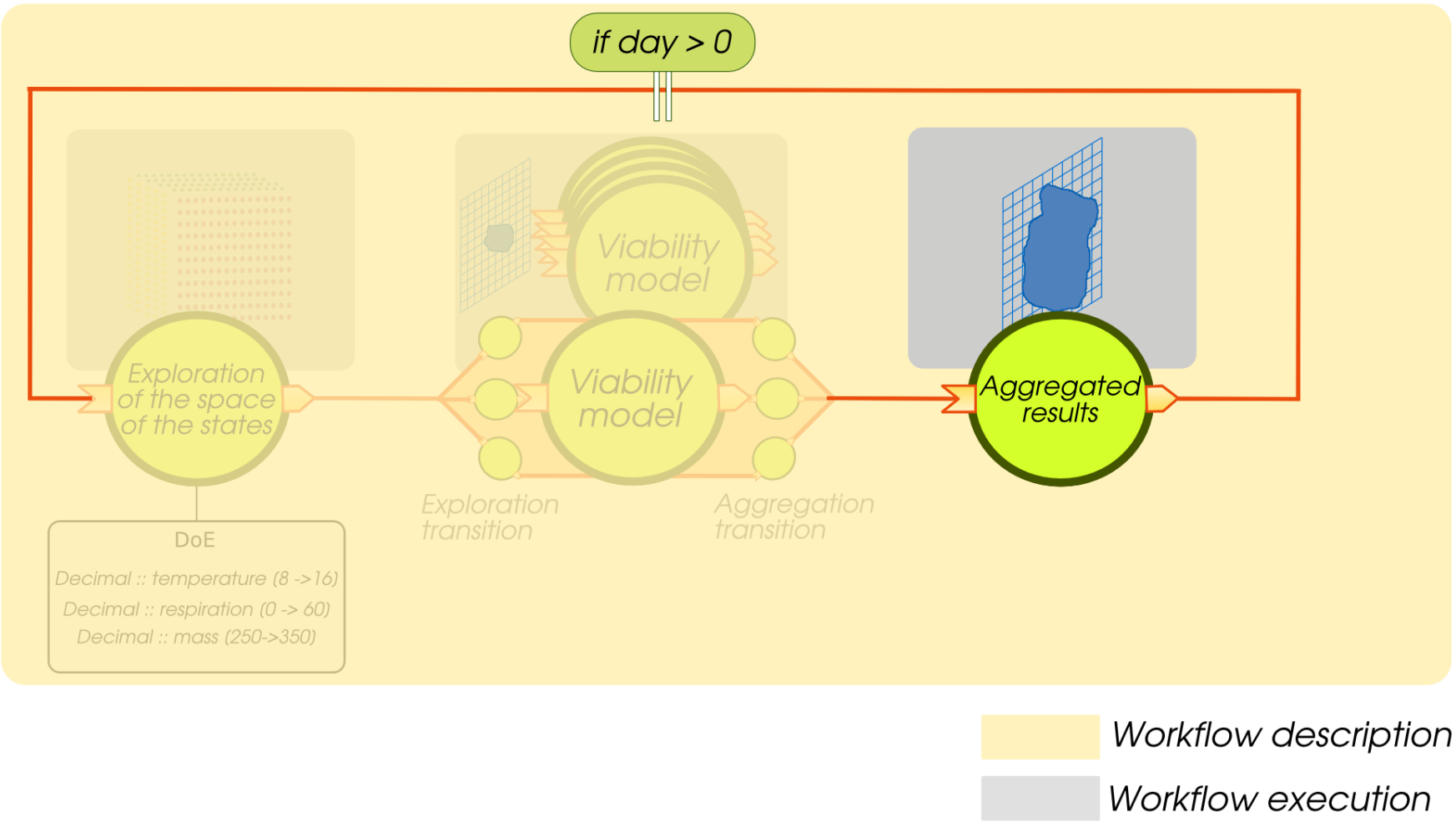
Workflow description

Workflow execution



Workflow description

Workflow execution


if day > 0
Exploration of the space of the states
Viability model
Viability model
Aggregated results
Exploration transition
Aggregation transition
DoE
Decimal :: temperature (8 ->16)
Decimal :: respiration (0 -> 60)
Decimal :: mass (250->350)
Workflow description
Workflow execution

Outline

1 Genesis of OpenMOLE

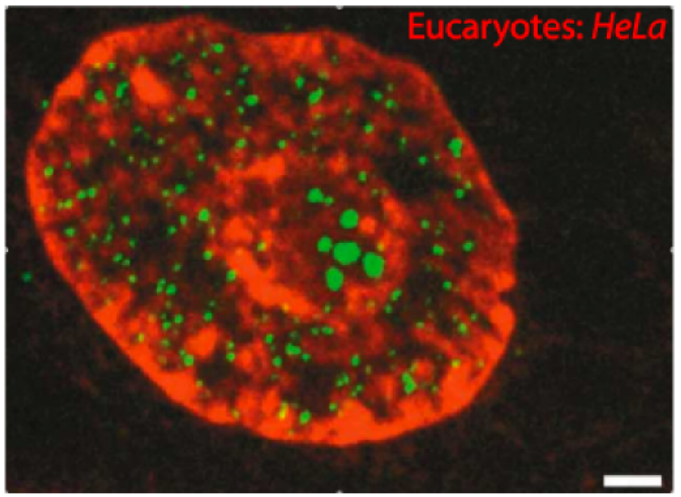
2 Central concepts

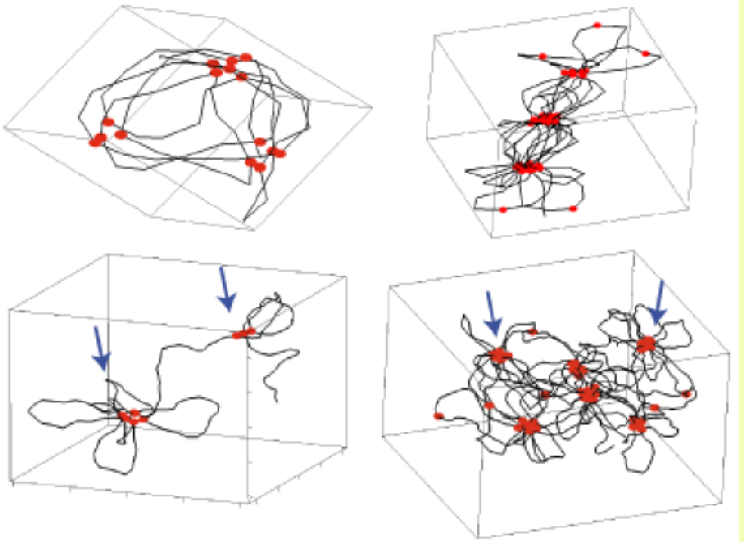
3 Task delegation

4 Detailed example

5 Scientific projects

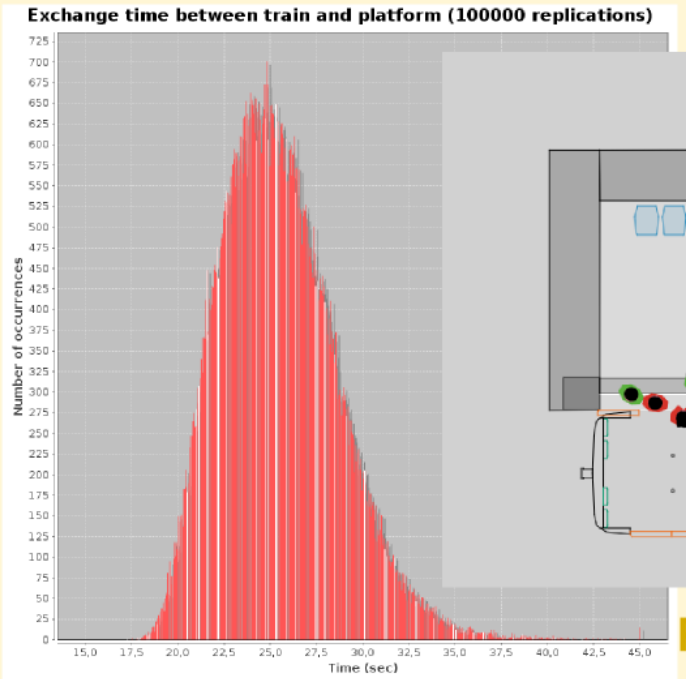
Bioinformatics

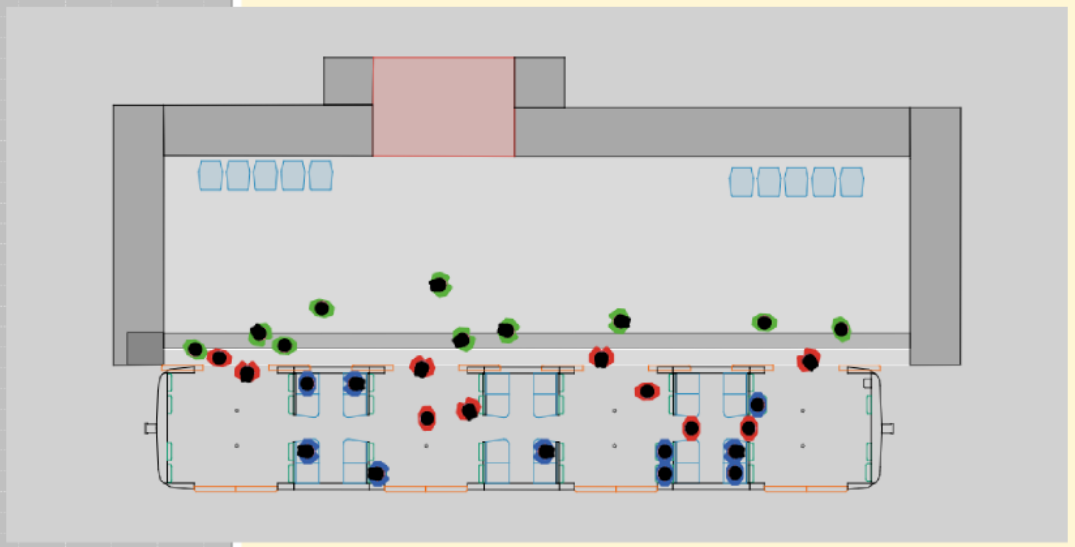




Chromosome structuring

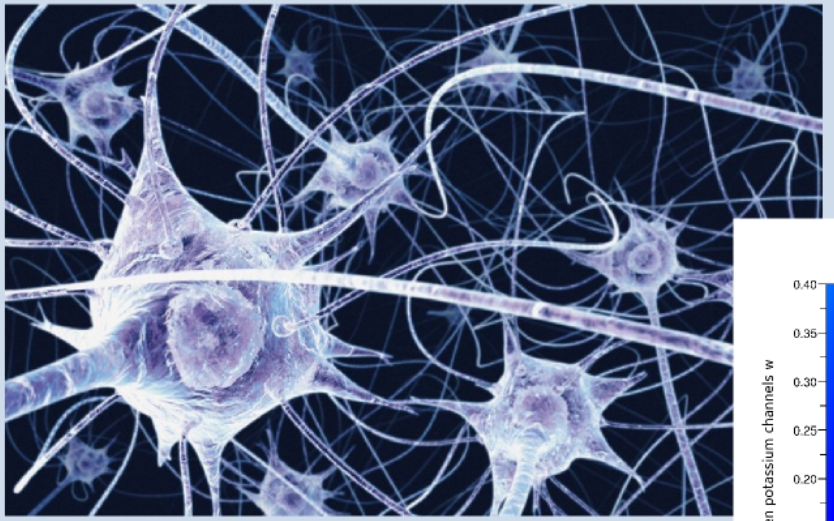
Agent based model



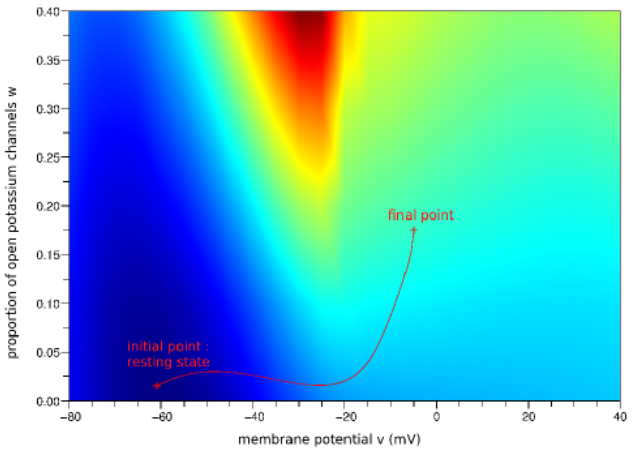


Passenger exchanges between trains and platforms

Neurosciences



Neuronal computing



Projects:

Computer science	multi-scale percolation in collaboration	**IRD institute**
Modeling theory	predator-prey model of savanna	**PATRES European project and CEMAGREF**
Food engineering	the viability theory to control complex food process	**INCALIN French research project and INRA**
Physics	self-propelled particles	**Complex System Institute**

Coming soon:

Image processing	cell tracking on embryos	**Bioemergences European research project**
Optimization	multi-objective optimization using distributed evolutionary algorithms	**DREAM European research project**
Image processing	functional analysis of brain images	**Curie institute**

Conclusion

The genericity of the platform has been assessed on many real use-cases.

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments.

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

- Extensible platform through plugins (20 plugins and the number is rapidly growing).

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

- Extensible platform through plugins (20 plugins and the number is rapidly growing).
- Statistically sounded over-submission algorithms for efficient grid execution.

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

- Extensible platform through plugins (20 plugins and the number is rapidly growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

- Extensible platform through plugins (20 plugins and the number is rapidly growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.
- Multi-scale workflows (tasks encapsulating workflows).

Conclusion

The genericity of the platform has been assessed on many real use-cases.

OpenMOLE reduces dramatically the engineering time to develop grid-enabled complex system in silico experiments. Advanced features:

- Extensible platform through plugins (20 plugins and the number is rapidly growing).
- Statistically sounded over-submission algorithms for efficient grid execution.
- Workflow jobs grouping in a single grid job.
- Multi-scale workflows (tasks encapsulating workflows).
- Data channels.

Conclusion

Work in progress:

Conclusion

Work in progress:

- Packaging a command line user interface version.

Conclusion

Work in progress:

- Packaging a command line user interface version.
- Designing a graphical user interface.

Conclusion

Work in progress:

- Packaging a command line user interface version.
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

Conclusion

Work in progress:

- Packaging a command line user interface version.
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.

Conclusion

Work in progress:

- Packaging a command line user interface version.
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.
- Collaborative development on workflows (decentralized versioning system).

Conclusion

Work in progress:

- Packaging a command line user interface version.
- Designing a graphical user interface.
- Writing consistent user documentation.

Perspectives:

- Generic workflow patterns for scientific algorithms.
- Collaborative development on workflows (decentralized versioning system).
- Meta-data management.

Acknowledgement & Question

We would like to thanks the JSAGA project and in particular Sylvain Reynaud (IN2P3) for his help.

Hello world!

```
helloFile = new Prototype<File>("helloFile", File.class)
generateTask = new GroovyTask("Generate file")
generateTask.setCode("helloFile = workspace.newTmpFile();\nhelloFile.write('Hello from groovy');\n")
generateTask.addOutput(helloFile)

displayTask = new GroovyTask("Display file")
displayTask.setCode("println helloFile.text")
displayTask.addInput(helloFile)

generateCapsule = new TaskCapsule(generateTask)
new SingleTransition(generateCapsule, new TaskCapsule(displayTask))
new Mole(generateCapsule).run()
```