MPhyScas

Multi-Physics and Multi-Scales Solver Environment

Tutorial

Universidade Federal de Pernambuco - UFPE

Centro de Informática - CIn

2012

1 Introduction

MPhyScaS (Multi-Physics and Multi-Scale Solver Environment) is an environment dedicated to the automatic development of simulators based on the Finite Element Method (FEM). The term multi-physics is a qualifier to a set of phenomena that interact in time and space. A multi-physics system can also be called a system of coupled phenomena. These phenomena are of different natures and behavior scale.

Usually, simulators based on the Finite Element Method can be organized in a layers architecture. In the top layer global iterative loops can be found, corresponding to the overall scenery of the simulation. The second layer contains what is called the solution algorithms. Each solution algorithm dictates the way linear systems are built and solved. The third layer contains the solvers for linear systems and all the machinery for operating with matrices and vectors. The last layer is the phenomenon layer, which is responsible for computing local matrices and vectors at the finite element level and assembling them into global data structures.

The MPhyScaS architecture establishes a computational representation for the computational layers using patterns (see Figure 1). The Kernel level represents the global scenery level, the level of the solution algorithms is represented by the Block level, the level of solvers is represented by the Group level, and the phenomena level is represented by the Phenomenon level.

The original architecture of MPhyScaS provides support to the automatic building of sequential simulators only. For instance, it does not have abstractions that could automatically define the distribution of data and procedures and their relationships across a cluster of PC's. The MPhyScaS parallel architecture (called MPhyScaS-P) satisfy a number of new requirements, including the support of parallel execution of the simulators in clusters of PC's.

Whenever the architecture of a computational system allows for a hierarchy of procedu-

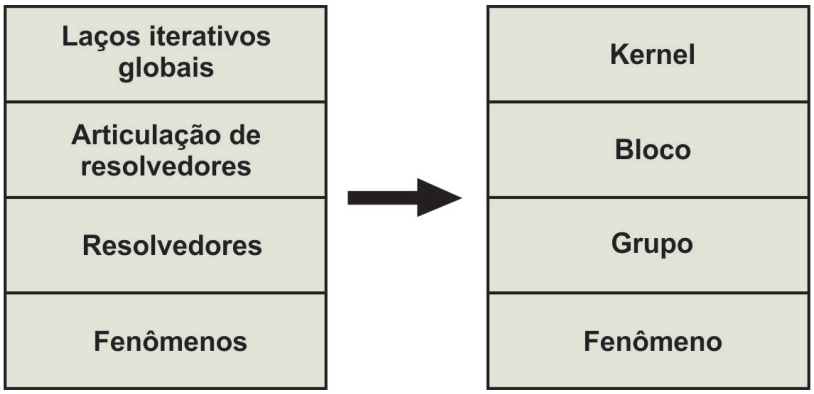

Figura 1: Computational representation for the layers of the simulator.

res, it may be a good idea to define a hierarchy of processes in such a way that few of them would accumulate some very light management tasks. The benefits for this strategy include: (i) procedures can be hierarchically synchronized (from coarse to fine grain), reducing management concerns and increasing correctness; (ii) since locality concerns change along the hierarchy levels, memory management can become more specialized from top to bottom; (iii) the hierarchy allows for the encapsulation of services and procedures, making it easier the exchange of components.

The topology of the procedures in the workflow of MPhyScaS-S is implemented in MPhyScaS-P in a hierarchical form with the aid of a set of processes, which are responsible for the procedures synchronization. There are four types of leader processes (see Figure 2):

- **Kernel Rank:** Processes that execute the Kernel algorithm. However, they execute all simulator tasks. One of these processes is the leader one of this level. These processes communicate (as master) with processes of Block rank type, for the execution of Block algorithms, before executing the algorithms of their Blocks. There must be only one Kernel rank process for each simulation.

- **Block Rank:** Processes that execute the Block algorithms required by a Kernel rank process. These processes communicate (as master) with processes of Group rank type, for the execution of GroupTasks (tasks programed by data) of the Groups, before

executing the GroupTasks of their Groups.

- **Group Rank:** Processes that execute the GroupTasks required by a Block rank process. These processes communicate (as master) with processes of Phenomenon rank type, for the execution of the execCodes (tasks programed by data for the execution of Phenomena specialized components) of Phenomena, before executing the execCodes of their Phenomena.

- **Phenomenon Rank:** Processes that execute the execCodes of Phenomena required by a Group rank process. When this process initiates its activity, all other processes are also executing similar tasks at Phenomena level.

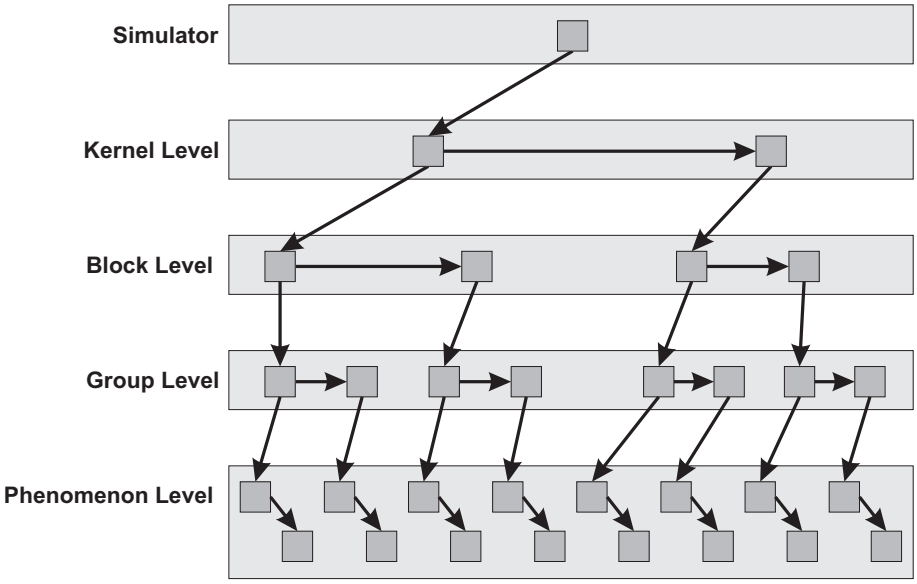

Figura 2: Layers with procedures executed by ClusterRank in MPhyScaS-P.

1.1 Systems

MPhyScas is composed by three distinct systems (see Figure 3):

- **Builder:** it is responsible for the management and the integration of software components to the simulator core in order to produce specific simulators for different proposes.

- **Simulator:** it is the product generated by the Builder, and it is responsible for making simulations using the available software components.

- **Repository:** it is responsible for the store and the management of software components, it makes difference between their utilities and it verifies the correctness of each one.

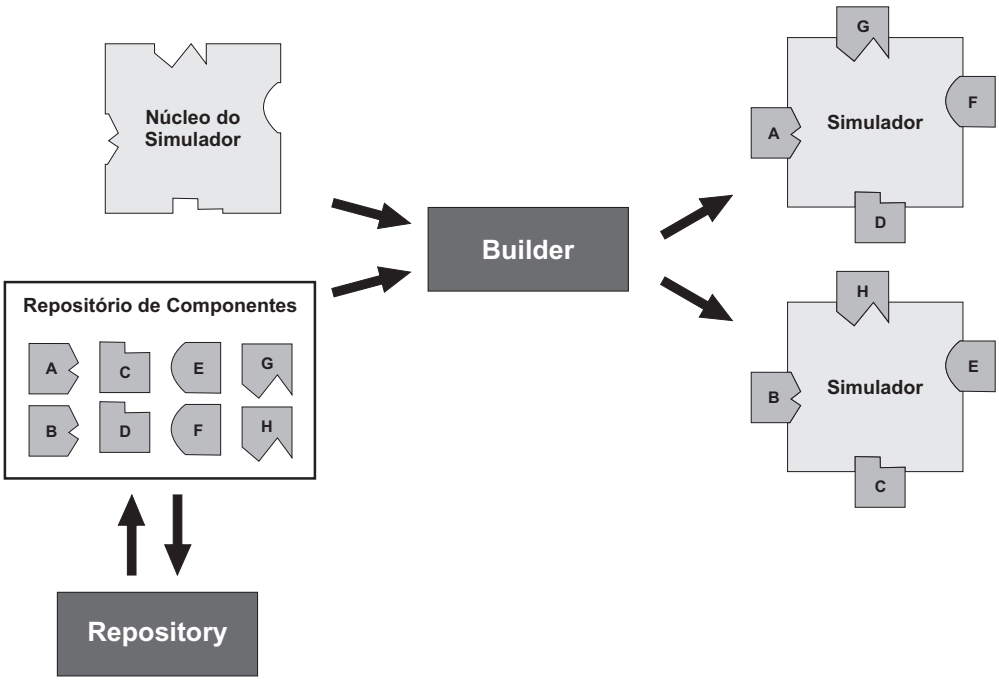

Figura 3: MPhyScas systems.

2 System Requirements and Installation

The available version for download at http://www.cin.ufpe.br/~mphyscas temporarily requires:

1. **Operation Sytems:**

   Linux or Mac

2. **Libraries**

   GSL - GNU Scientific Library (http://community.schemewiki.org/?GEE-Guile-GSL-install)

   MPICH (http://ftp.mcs.anl.gov/pub/mpi/mpich2-doc-install.pdf)

   LAMBOOT (http://www.lam-mpi.org/tutorials/lam/boot.php)

   XERCES-C (http://xerces.apache.org/xerces-c/)

3 Builder

3.1 Kernel

The Kernel is in the first layer of MPhyScas architecture. It represents the overall scenery of the simulation. It is responsible for initialization of procedures (transferred to Blocks in the lower level), for global time loops and iterations, for global adaptive iterations, and for articulations of activities to be executed by the Blocks in the lower level. The Kernel stores system data related to the parameters for its loops and iterations. Therefore, in a simulation only one Kernel can exists.

3.1.1 Kernel View

Figure 4 shows a screenshot of Kernel view in Builder system.

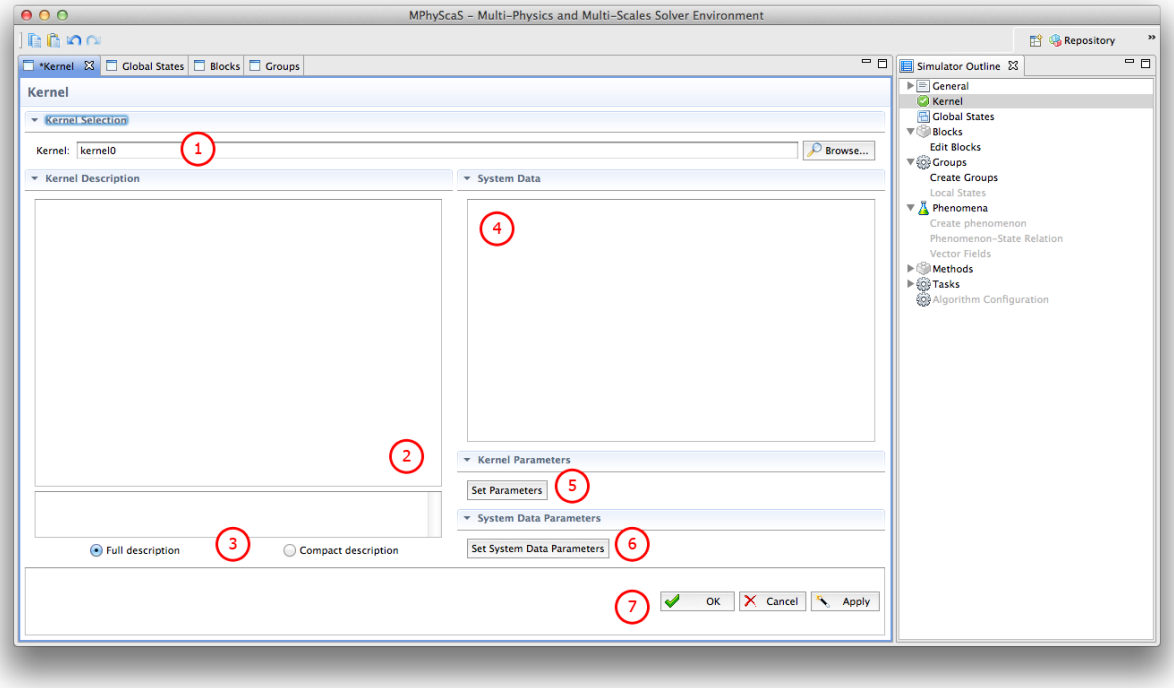

Figura 4: Kernel view.

3.2 Global States

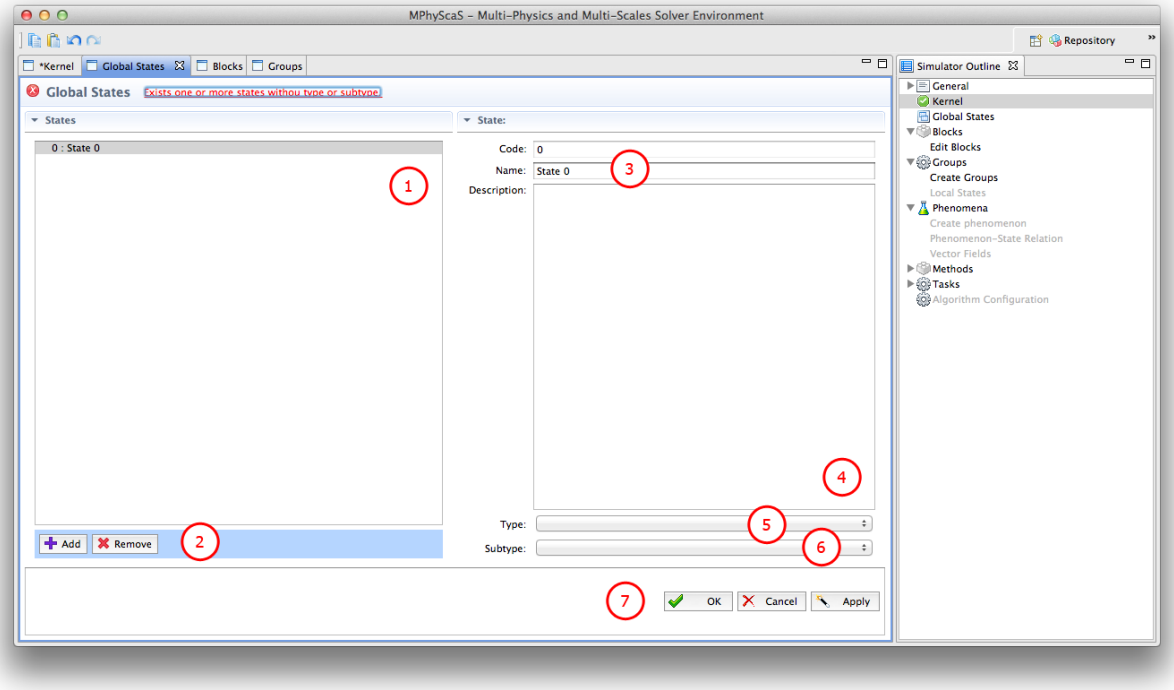

Figura 5: Global States view.

3.3 Blocks

The Block is in the second layer of MPhyScas architecture. It represents the solution algorithms, in other words, the way linear systems are built and solved. It is responsible for the transfer of incoming demands form the Kernel to its Groups in the lower level (initialization procedures, for instance); for Block local time loops and iterations (inner loops and iterations inside a global time step, restricted to groups of Phenomena); for procedures inside time stepping schemes; for operations with global quantities (transfered to Groups in the lower level, which are the owners of global quantities). The Blocks serve the Kernel level. Each Block is responsible for a certain number of Groups, which can not be owned by other Block. All demands for a Block to the lower level should be addressed to its Groups. The Blocks

store system data related to parameters for their own loops and iterations, and parameters for their procedures.

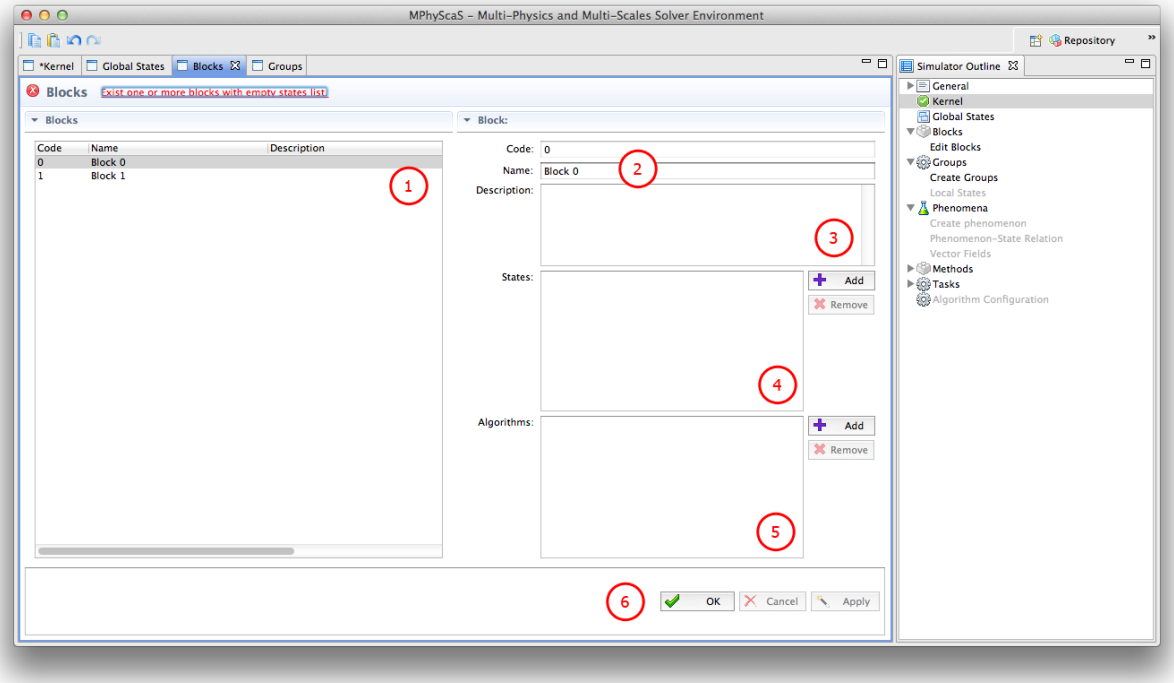

Figura 6: Blocks view.

3.4 Groups

The Group is in the third layer of MPhyScas architecture. It represents the solvers of linear systems. It is responsible for the transfer of incoming demands from its Block to Phenomena in the lower level (initialization procedures, for instance); for the assembling coordination and solution os systems of linear algebraic equations (the method used depends on the solver component); for operations with global quantities (by demand from its Block); for articulation of activities to be executed by its Phenomena in the lower level (basically concerned with computation and assembling of global matrices and vectors). The Groups serve their respective Blocks. Each Group is responsible for a certain number of Phenomena, which can not be owned by other Group. All demands from a Group to the lower level should

be addressed to its Phenomena only. The Groups store global matrices, vectors and scalars and store GroupTasks.

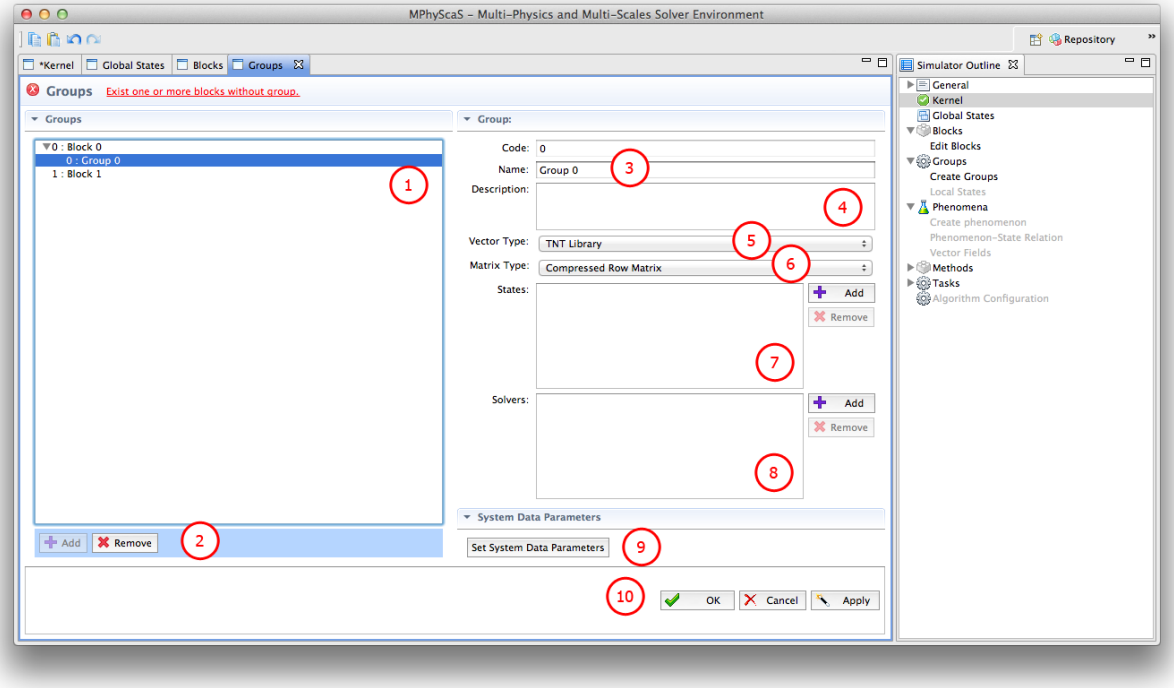

Figura 7: Groups view.

3.5 GroupTasks

The GroupTasks are stored by the Groups. The GroupTasks are objects encapsulating standard procedures, where articulation of the Group's Phenomena are needed. The GroupTasks are programmable and their data are standard pieces of information, depending only on the type of GroupTask.

3.6 Phenomena

The Phenomenon is the last MPhyScas layer. It is responsible for the computation of local matrices, vectors and scalars (Phenomenon quantities); for operations involving matrices

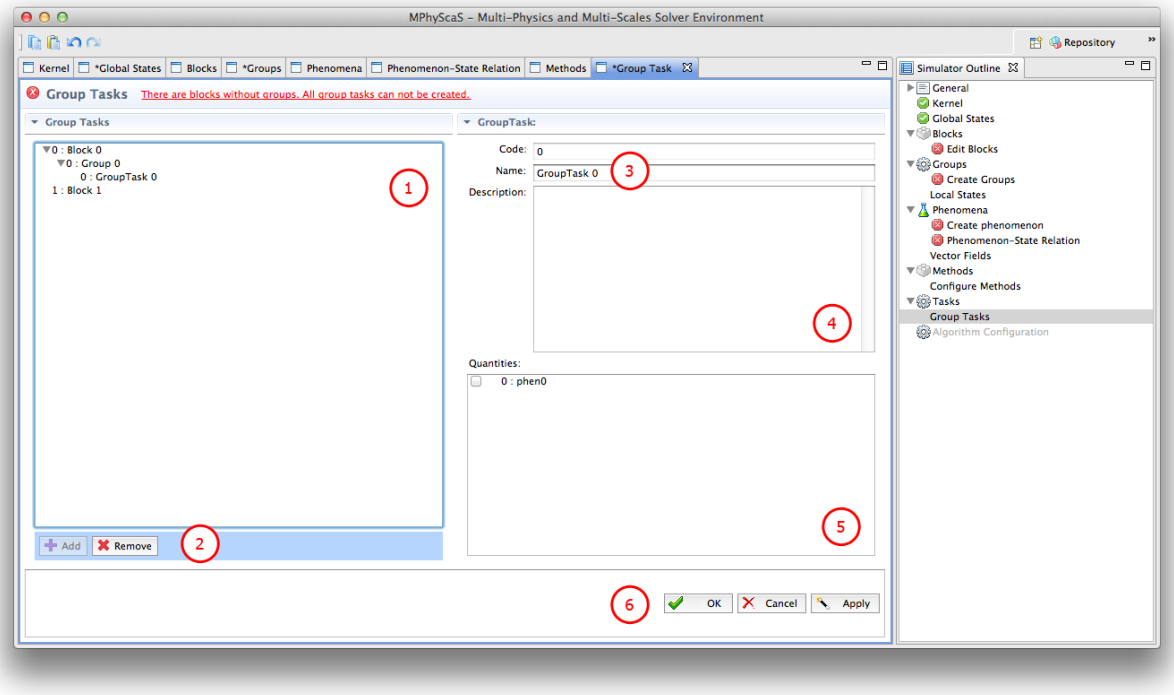

Figura 8: GroupTasks view.

and vectors at the finite element level and their assembling into given global matrices and vectors. The Phenomena serve their respective Groups. The Phenomena store data related to constitutive parameters or other parameters, which are specific of the respective Phenomenon; store the geometry where the respective Phenomenon is defined (different Phenomena may share a geometry or a part of it); store WeakForms, which are tools for computing and assembling quantities defined on a certain part of the geometry. A WeakForm may store parameters, which are related to specific simulation data (for instance, functions for the definition of the boundary conditions or parameters needed for the computation of a quantity, which should be given together within a simulation data set). The Phenomenon should also store methods.

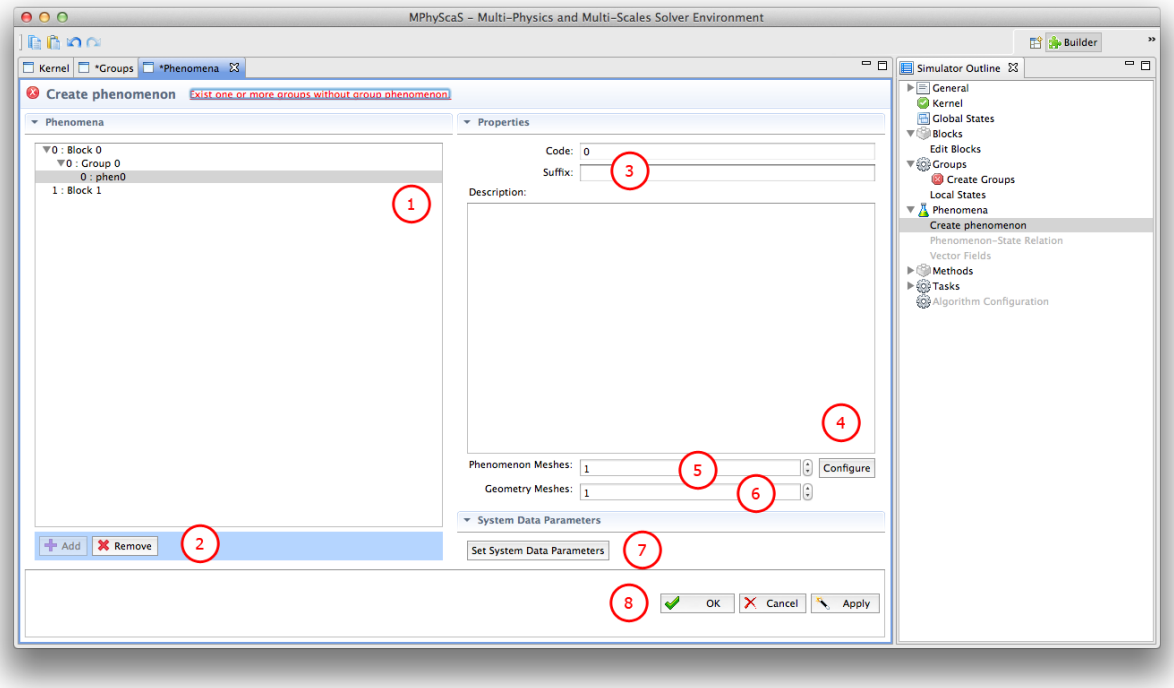

Figura 9: Phenomena view.

3.7 Phenomenon-State

3.8 Vector Fields

3.9 Methods

The methods are stored by the Phenomena. They are tools to be used in certain Phenomenon specific tasks. For instance, those tasks can be generation of geometric and Phenomenon meshes, numerical integration at the element level, shape function, etc.

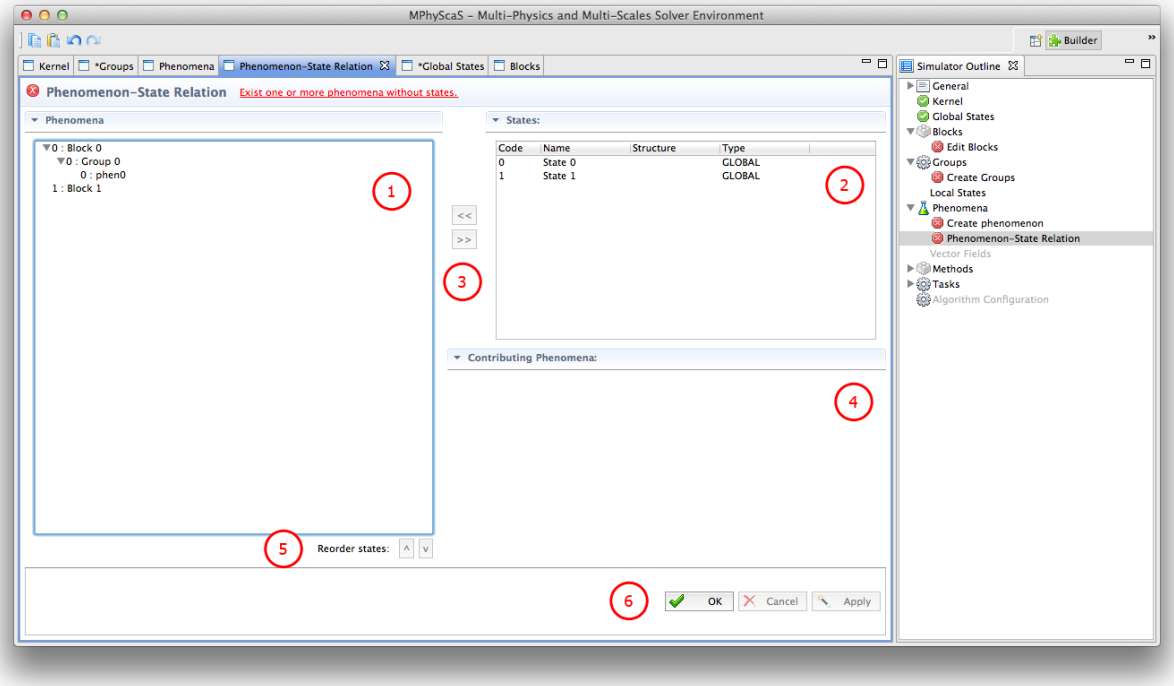

Figura 10: Phenomenon-State view.

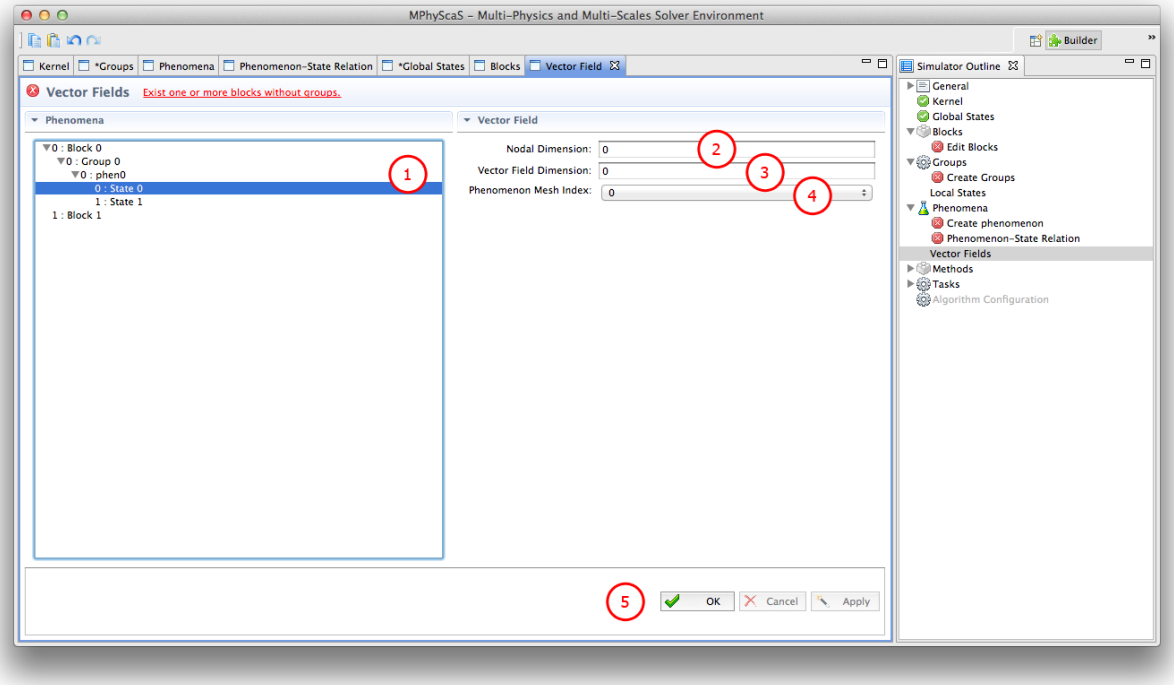

Figura 11: Vector Fields view.

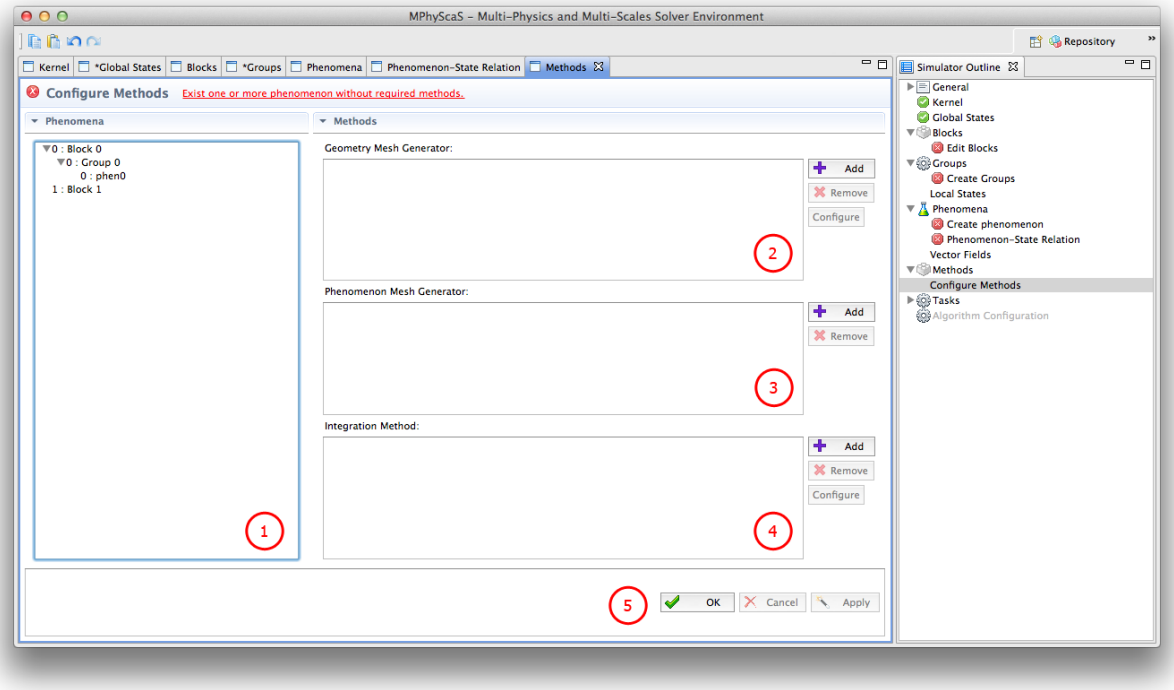

Figura 12: Methods view.

4 Simulator

5 Repository