Domain Model

Definitions

Global mesh: a collection of element blocks

Field: An unknown degree of freedom associated with a discrete representation on the mesh defined on at least one element block but not necessarily on all alement blocks. All fields must be registered with the global indexer to define the relations to the global mesh.

Element Block: A heterogeneous set of elements. By heterogeneous we mean that multiple topological cells of the same dimension are present (e.g., hex and tet combined).

Equation Set: A description of equations that can be replicated within a physics block. It defines it's own set of fields, quadrature rules, and basis functions.

Physics Block: A collection of equation sets and an associated element block that define all fields for that element block. A physics block is instantiated for each element block. A description of a physics block can be used for multiple element blocks. A physics block will have multiple field managers, one for each topological cell for the volume integration and one for each boundary condition applied to the physics block.

* A physics block is built using a single topological entity for a single element block
  + A volume physics block is cloned with a different cell topology to create the side physics blocks. We keep the volume physics blocks around and can always clone for boundary conditions. This allows for all the volume assembly machinery to be reused for boundary conditions with a different mdarray layout
* It can be constructed for a volume or side cell topology
* Builds all evaluators associated with a (cell topo, e block) pair
* cell topo: can be volume topo or side topo
* Fills a single field manager
* Unique phsyics block for each (the two objects below are essentially a pairing of a set of worksets with unique layouts):
  + (volume cell topo,Eblock)
  + number of FM/Physics block objects = (side topo+side ID, boundary condition+sideset+eblock)
* Is also used for initial conditions

Global Indexer: Provides a unique numbering for element indexed discrete field values using the global mesh topology (i.e. provides field global ids on each element)

Connection Manager:

Field Manager:

Field Pattern:

FieldLibrary:

LinearObjFactory

LinearObjContainer:

Worksets:

Workset Container:

Evaluator:

MDArray/MDField:

ModelEvaluator:

ModelEvaluatorFactory:

Scatter:

Gather:

Basis:

BasisValues:

IntegrationRule:

IntegrationValues:

PointRule:

PointValues:

BasisIRLayout:

BC Strategy:

Factory:

Composite:

Parameter Library:

Response Library:

Global Data:

Assembly Engine:

These terms are distinct form the Phalanx domain model.

Issues to address:

1. Domain Model

1. [ROGER] Unification of Assembly, Equation Sets – more aptly named methods in DefaultImpl
   1. [FINISHED] Split the registerGatherScatter into:
      1. registerGatherAndOrientations
      2. registerScatter
      3. registerDOFFieldsAtQP (values,grad,curl,div)
   2. Eliminate BCStrategy → make them specially defined equation set default implemetions. UPDATE (2012.05.25): I don't think this is possible. Since bcs span physics from multiple equation sets, it needs the physics block to build certain evaluators. However the interface for equation sets knows nothing about the physics block since the physics block own multiple equation sets. We would have to modify the equation sets to accept the physics block as an argument which is a little strange and doesn't make sense for volume assembly. We will have to think more about this. Most of the machinery for both equation sets and bc\_strategies is the same so maybe we build some basic tools that both can specialize from. But the more I think about it, I think the generalized part of the tools already exist. We may have to live with this for now. It might take a really big redesign to address this.
   3. Derive IC from equation set as another default imp of equation set
   4. Drekar specific: superobject for simple dirichlet and neumann conditions that combines the bcstrategy object (soon to be equation set derived object) and the user specific evaluator
   5. [Eric] Update FieldLibraryBase interface to use vectors and not lists
   6. [Eric] Make sure PointValues and Evaluators are ready for prime time
   7. Think about a defaultImpl for response library but be careful – responses aggregate across physics blocks!
2. [bring up at TUG] Vector support in intrepid (+gather, scatter in panzer)
3. Mixed discretizations in eqn sets: could be handled with switch to using field library in the register<gather,scatter> refactor in [2]. Not clear if this should be handled as part of 2 or pushed off.
4. Package infrastructure:
   1. Panzer
      1. Assembly
      2. DOFIndexer
      3. STKAssembly
      4. STKDOFIndexer
      5. exmaples/main\_driver → leave under examples
   2. Drekar
      1. Exe
      2. Lib
5. Response Library:
   1. Assembly engine trimmed down/specialized/more intelligent to evaluate other responses.
   2. Residuals/JAc/SGRes/SGJac
6. ParameterList parsing: clean up for non-const storing of defaults
7. [ROGER] Possible phalanx change, add string description to the DataLayout for supporting multiple integration rules within a PhysicsBlock
8. [Eric] Global assembly data passed in through pre evaluate method on the evaluators
   1. workset is simplified and const
   2. lazy evaluation of “values” quantities
   3. Field pointers instead of copies of values objects