Agile Components for the Rapid Development of Production Applications

SAND2008-7503P

Andy Salinger, CSRI,
Sandia National Laboratories

Strategic Goals:
To align the many efforts at Sandia involved in developing software for the modeling and simulation of physical systems (mostly PDEs):

1. To enable the Rapid development of new Production codes
2. To embed the codes with Transformational design, analysis, and decision-support capabilities
3. To benefit from Synergy & reduce Redundancy

Consensus Vision: Our goals will best be met by delivering:

- A full range of independent yet interoperable software components
  - ✓*Capabilities*
  - ✓*Interfaces*
- Software quality tools and procedures
- Prototype applications that demonstrate use cases, vertical integration, and verification, and provide pull

Agile Components

Analysis Tools (*black-box*)

- Optimization
- Parameter Studies
- UQ (non-invasive)
- V&V, Calibration
- OUU, Reliability
- Computational Steering

Analysis Tools (*embedded*)

- Nonlinear Solver
- Time Integration
- Continuation
- Sensitivity Analysis
- Stability Analysis
- Constrained Solves
- Optimization
- UQ Solver

Linear Algebra

- Data Structures
- Iterative Solvers
- Direct Solvers
- Eigen Solver
- Preconditioners
- Matrix Partitioning
- Architecture-Dependent Kernels

Composite Physics

- MultiPhysics Coupling
- Solution Control
- System Models
- System UQ

Mesh Tools

- Mesh I/O
- Inline Meshing
- Partitioning
- Load Balancing
- Adaptivity
- Remeshing
- Grid Transfers
- Mesh Quality

Particle Code Tools

- Data Structures
- Neighbor Search / Sort

PostProcessing

- Visualization
- Embedded Verification
- Feature Extraction
- Data Reduction
- Model Reduction

Mesh Database

- Mesh Database
- Geometry Database
- Solution Database

Local Fill

Discretizations

- Discretization Library
- Field Manager

Derivative Tools

- UQ / PCE Propagation
- Sensitivities
- Derivatives
- Adjoints

Physics Fill

- Element Level Fill
- Material Models

- Objective Function
- Constraints
- Error Estimates
- MMS Source Terms

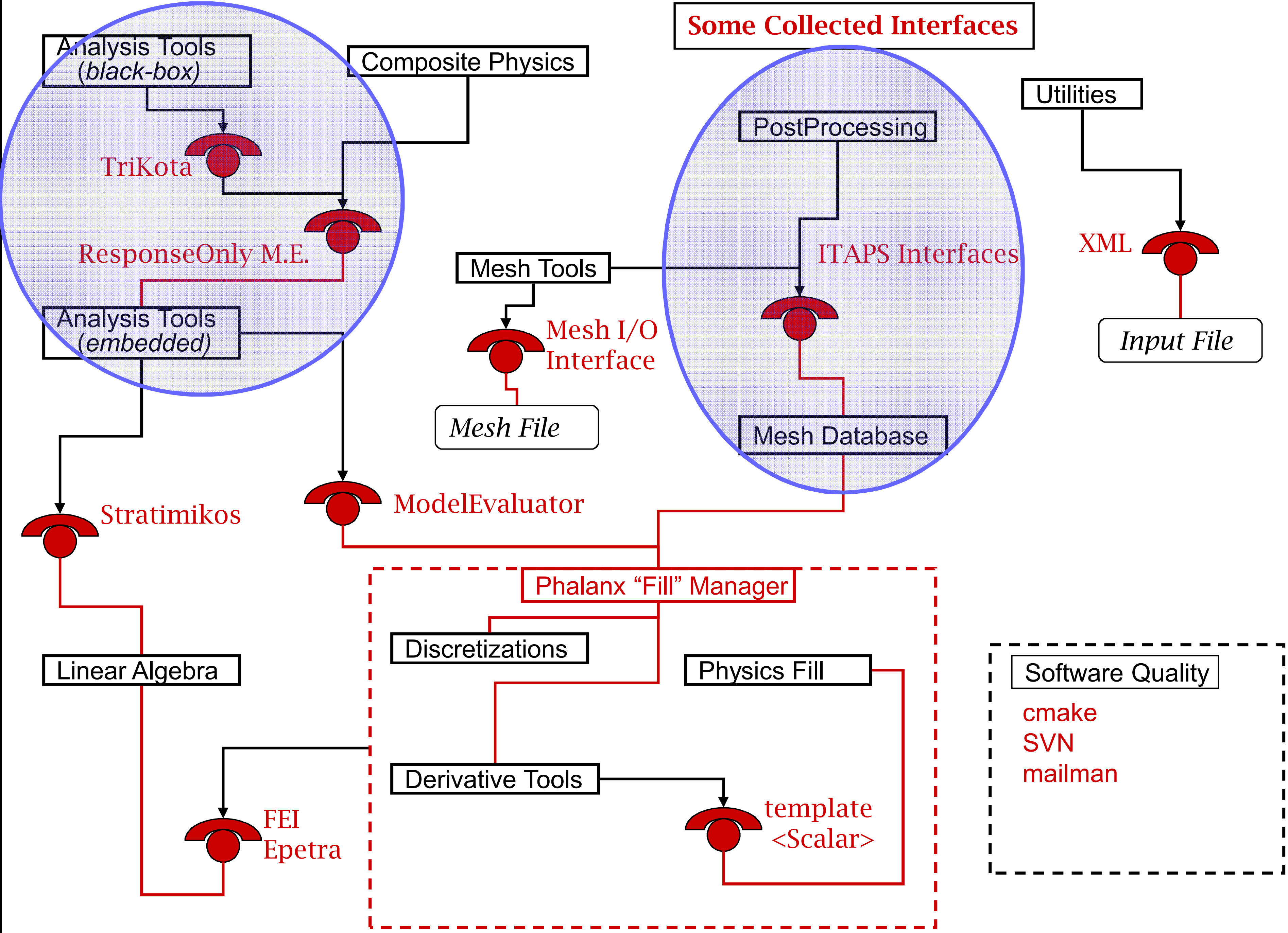
Utilities

- Input File Parser
- Parameter List
- I/O Management
- Memory Management
- Communicators
- Runtime Compiler
- MultiCore Parallelization Tools

Software Quality

- Version Control
- Regression Testing
- Build System
- Backups
- Mailing Lists
- Unit Testing
- Bug Tracking
- Performance Testing
- Code Coverage
- Porting
- Web Pages
- Release Process

Some Collected Interfaces



Analysis Tools (*black-box*)

Composite Physics

TriKota

ResponseOnly M.E.

Analysis Tools (*embedded*)

Mesh Tools

Mesh I/O Interface

Mesh File

PostProcessing

ITAPS Interfaces

Mesh Database

Utilities

XML

Input File

Stratimikos

ModelEvaluator

Linear Algebra

FEI Epetra

Phalanx “Fill” Manager

Discretizations

Physics Fill

Derivative Tools

template <Scalar>

Software Quality

cmake

SVN

mailman

Response Only Model Evaluator: an “Application” or “Problem” Abstraction

A Model Evaluator registers what Input and Output it can handle.

I can make use of:

$$\cancel{x}$$

$$\cancel{\dot{x}}$$

$$p$$

$$\cancel{t}$$

I can compute:

$$\cancel{f(\dot{x}, x, p, t)}$$

$$\cancel{W = \alpha\frac{df}{d\dot{x}} + \beta\frac{df}{dx}}$$

$$\cancel{\frac{df}{dp}}$$

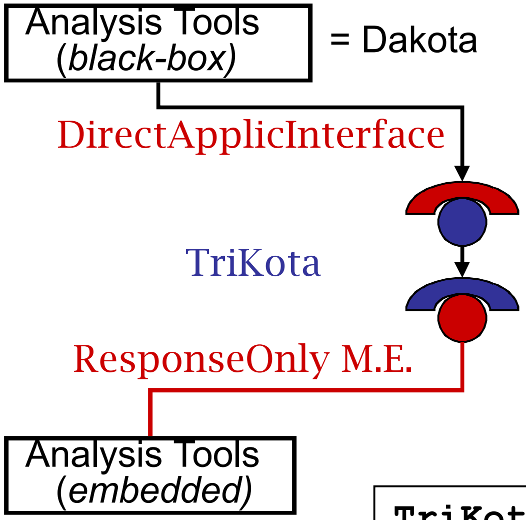
$$g(\cancel{x}, p)$$

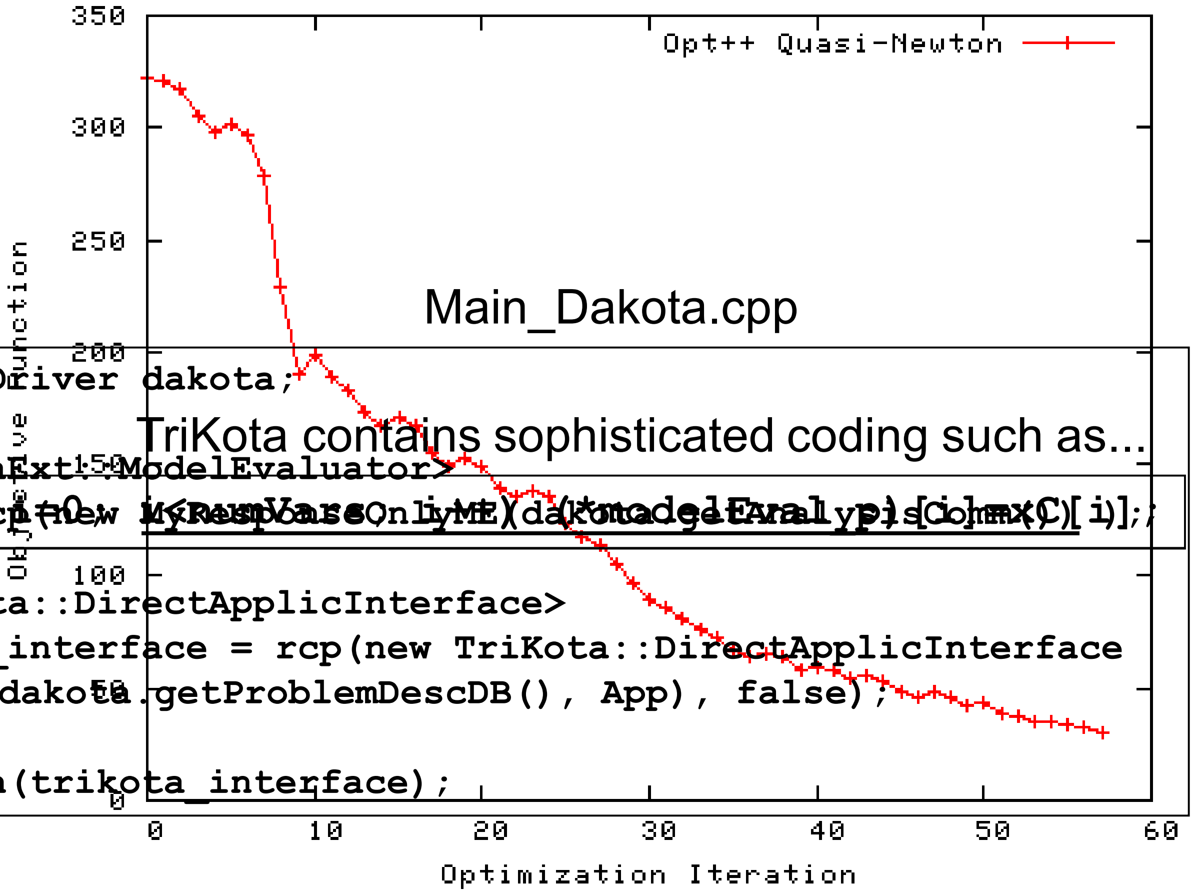
$$\cancel{\frac{dg}{dx}}$$

$$\frac{dg}{dp}$$

Response Only Model Evaluator: $p \rightarrow g(p), \frac{dg}{dp}$

"TriKota" is an adapter between Dakota's "DirectApplicInterface" and Trilinos' "ResponseOnlyModelEvaluator"





Main_Dakota.cpp

```
TriKota::Driver dakota;

RCP<EpetraExt::ModelEvaluator>
  App = rcp(new MyResponseOnlyME(dakota.getAnalysisComm()));

RCP<TriKota::DirectApplicInterface>
  trikota_interface = rcp(new TriKota::DirectApplicInterface
          (dakota.getProblemDescDB(), App), false);

dakota.run(trikota_interface);
```

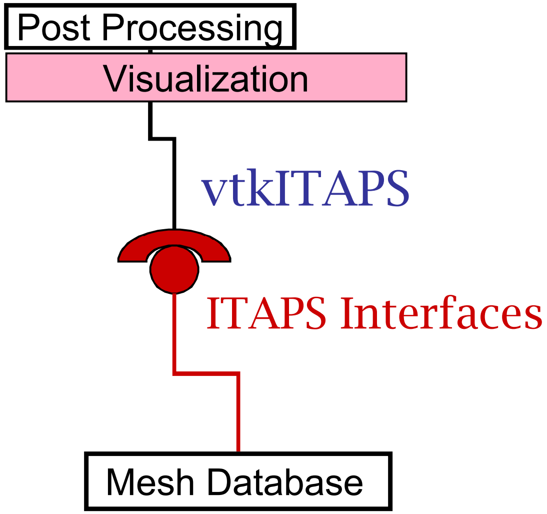
TriKota contains sophisticated coding such as...

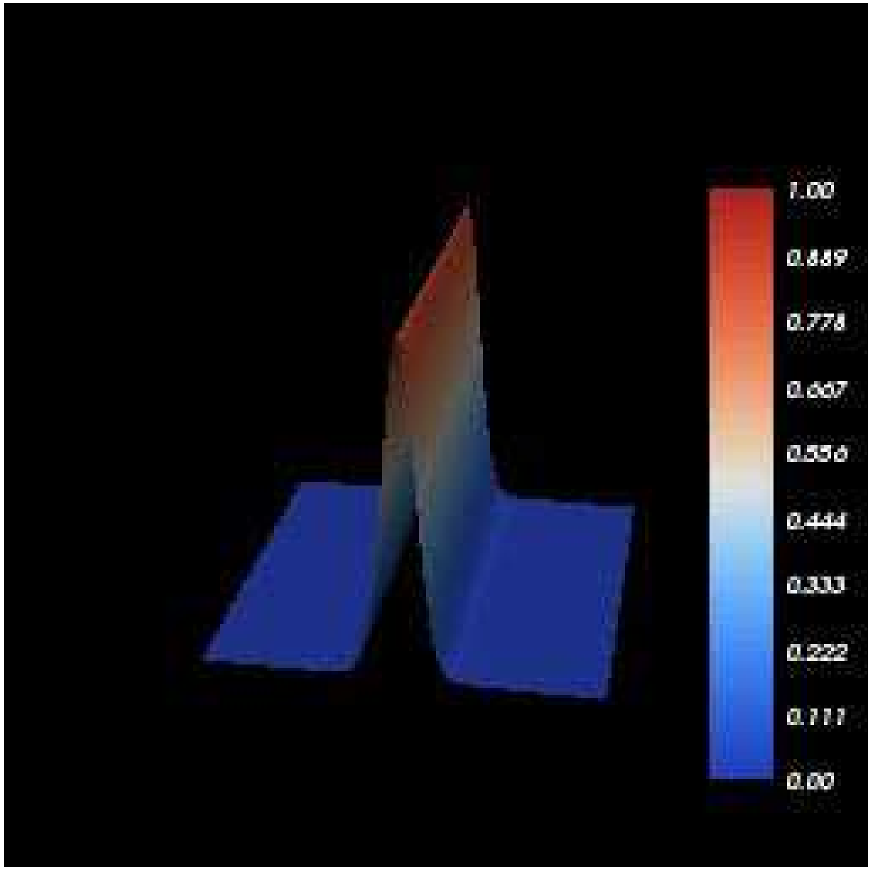
```
for (int i=0; i<numVars; i++) (*modelEval_p)[i] = xC[i];
```

Joint work with Bartlett, Adams, Eldred, Gay

VTK visualization software hooked up to generic ITAPS Mesh Database Interface

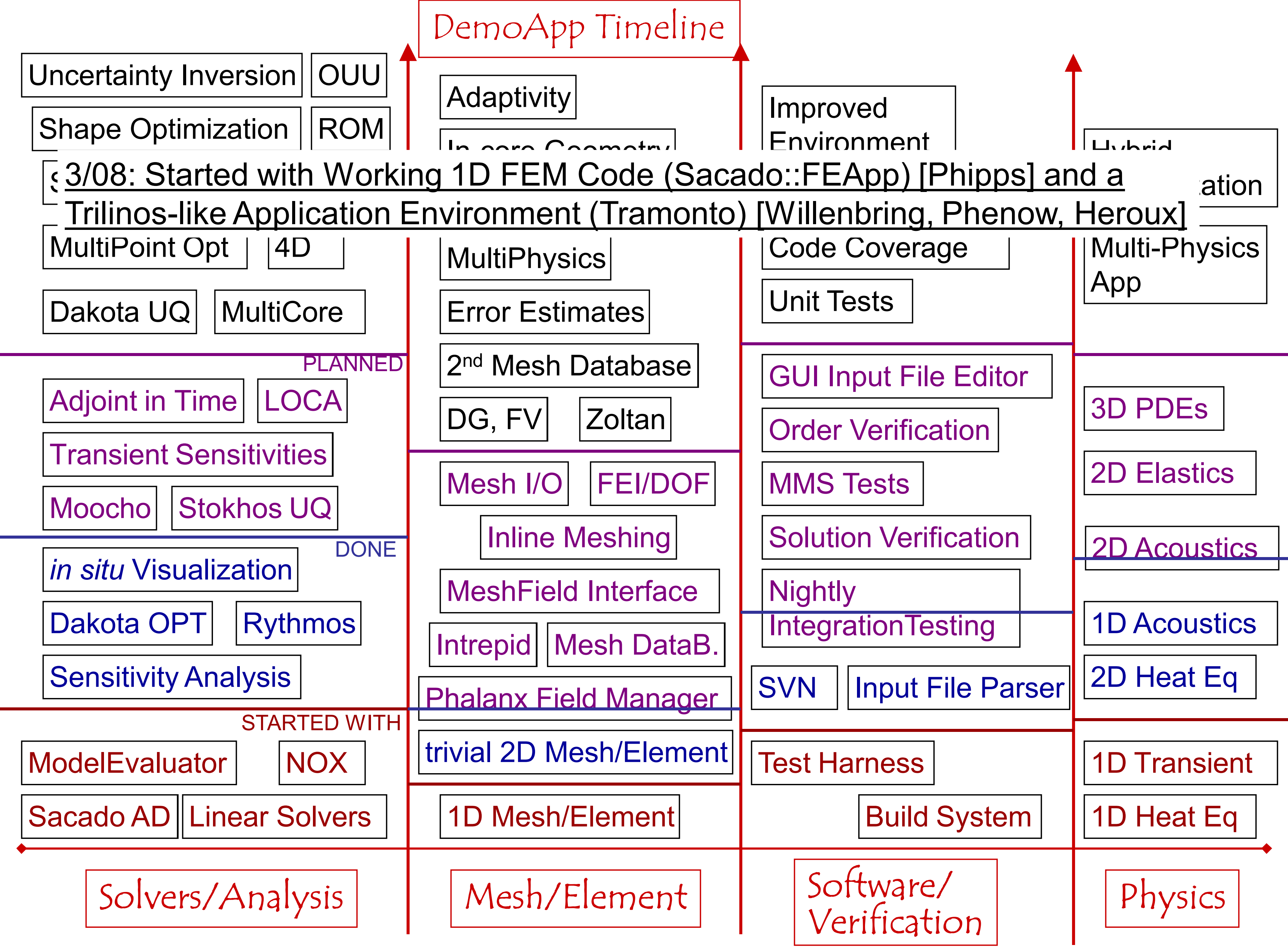
Joint work with Nathan Fabian (1426)
with help from Vitus Leung & Karen Devine





A frame is dumped every time step from a RythmosObserver object.

DemoApp Timeline



3/08: Started with Working 1D FEM Code (Sacado::FEApp) [Phipps] and a Trilinos-like Application Environment (Tramonto) [Willenbring, Phenow, Heroux]

Uncertainty Inversion
OUU
Shape Optimization
ROM
S
MultiPoint Opt
4D
Dakota UQ
MultiCore

PLANNED

Adjoint in Time
LOCA
Transient Sensitivities
Moocho
Stokhos UQ

DONE

in situ Visualization
Dakota OPT
Rythmos
Sensitivity Analysis

STARTED WITH

ModelEvaluator
NOX
Sacado AD
Linear Solvers

Solvers/Analysis

Adaptivity
In-core Geometry
MultiPhysics
Error Estimates
2nd Mesh Database
DG, FV
Zoltan
Mesh I/O
FEI/DOF
Inline Meshing
MeshField Interface
Intrepid
Mesh DataB.
Phalanx Field Manager
trivial 2D Mesh/Element
1D Mesh/Element

Mesh/Element

Improved Environment
Code Coverage
Unit Tests
GUI Input File Editor
Order Verification
MMS Tests
Solution Verification
Nightly IntegrationTesting
SVN
Input File Parser
Test Harness
Build System

Software/Verification

Hybrid
ation
Multi-Physics App
3D PDEs
2D Elastics
2D Acoustics
1D Acoustics
2D Heat Eq
1D Transient
1D Heat Eq

Physics