• **Numerical Algorithm Interoperability and Vertical Integration**
  – Abstract Numerical Algorithms (ANAs)
  – Thyra (Interoperability and vertical integration of ANAs)
  – Epetra (Interoperability of element-based numerical algorithms)

• **General Software Interoperability and Integration**
  – Memory management (Teuchos::RCP, ...)
  – User input and configuration control (Teuchos::ParameterList, ...)
  – User introspection (Teuchos::FancyOStream, ...)

• **Skin packages (wrappers for other languages)**
  – PyTrilinos
  – ForTrilinos

• **General Software Quality and Design**

• **Lean/Agile Software Engineering Principles and Practices**
  – Internal Trilinos issues
  – External customer issues
Trilinos Strategic Goals

- **Scalable Computations**: As problem size and processor counts increase, the cost of the computation will remain nearly fixed.
- **Hardened Computations**: Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.
- **Full Vertical Coverage**: Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

**Algorithmic Goals**

- **Grand Universal Interoperability**: All Trilinos packages will be interoperable so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Accessibility**: All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.
- **Universal Solver RAS**: Trilinos will be:
  - **Reliable**: Leading edge hardened, scalable solutions for each of these applications
  - **Available**: Integrated into every major application at Sandia
  - **Serviceable**: Easy to maintain and upgrade within the application environment.

**Software Goals**

*Thyra* is being developed to address this issue.

Courtesy of Mike Heroux, Trilinos Project Leader
Example: Numerous interactions exist between layers of abstract numerical algorithms (ANAs) in a transient optimization problem.

Key Points
• Higher level algorithms, like optimization, require a lot of interoperability
• Interoperability and vertical integration must be “easy” or these configurations will not be achieved in practice

What is needed to solve problem?
• Standard interfaces to break $O(N^2)$ 1-to-1 couplings

Thyra is being developed to address interoperability of ANAs by defining interfaces for:
- Linear operators/vectors
- Preconditioners / Linear solvers
- Nonlinear models
- Nonlinear solvers
- Transient solvers

Diagram:
- Linear Solver
  - AztecOO, Amesos, Belos, ???
- Nonlinear Optimizer
  - MOOCHO, ???
- Nonlinear Solver
  - NOX, PETSc, ???
- Transient Solver
  - Rythmos, SUNDIALS, ???
- Operators and Vectors
  - Epetra, Tpetra, PETSc, ???
- Application
  - Charon, Aria, ???
General Software Interoperability and Integration

• Memory management
  – Replace all raw C++ pointers in all higher level C++ code
  – Single objects: Teuchos::RCP, Teuchos::Ptr,
  – Arrays of objects: Teuchos::Array, Teuchos::ArrayRCP, Teuchos::ArrayView, ...

• User input and configuration control
  – Teuchos::ParameterList:
    • General parameter database
    • Self documenting
    • Validation support
    • XML input and output
  – Teuchos::ParameterListAcceptor:
    • Standard interface & protocol for handling ParameterList

• User introspection
  – Teuchos::FancyOStream
    • Formatted nested output
  – Teuchos::Describable
    • Flexible output of the state of an object
  – Teuchos::VerboseObject
    • Output showing dynamic behavior of an object
  – Teuchos::TimeMonitor
    • Targeted timing of critical computations and performance monitoring
Lean/Agile Software Engineering Principles and Practices

• **Internal Trilinos development tools principles and practices**
  – Scalability and robustness of build system and test tools
  – Continuous integration development principles and practices
  – Release process principles and practices

• **Integration with customer application codes**
  – Coordination of co-development with customer application codes (i.e. daily integration and asynchronous continuous integration)
  – Coordination of release schedules with customer application codes