SAND2011-7149C

COMBUSTION RESEARCH FACILITY

Morfeus:
Multiphysics Object-oriented Reconfigurable Fluid Environment for Unified Simulations

Damian Rouson, Karla Morris
Sandia National Laboratories

Xioafeng Xu, Joel Koplik
City University of New York

Sponsor: ONR

Review of FY08-FY11

Outline

- Introduction
  - Motivation
  - Objectives
- Methodology:
  - Design patterns
  - Analyses
- Results
  - Alpha/Beta Solvers
  - Navy applications
  - Scientific impact
- Conclusions
- Future Work

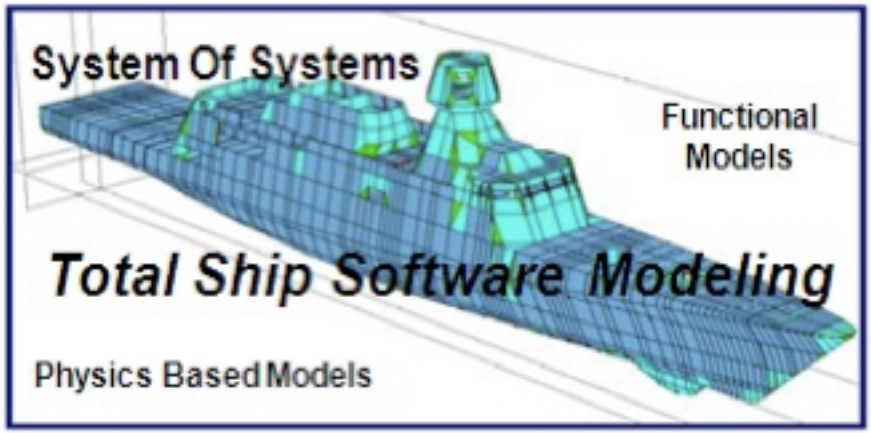
Motivation

"Taking concepts from Software Engineering and Service-oriented architecture (SOA) design, automation should be approached as distributed software systems - "a suite of interoperable services" - loosely coupled modules that are able to collaborate by subscription to a shared information model."

Seman, A. (2011) *A Vision for Next Generation Naval Machinery Monitoring and Control Guiding Principles for Navy After Next Machinery Automation Research*

Objectives

- To develop a SOA for physics-based models
  - Comprising a suite of loosely coupled, interoperable modules
  - Sharing a common, distributed information model
  - Collaborating to accomplish multiphysics simulations
- To develop analytical approaches to quantifying the desirable properties of the SOA.
- To promulgate the resulting software design patterns in publications and open-source software.
- To demonstrate the SOA on problems relevant to the Navy.

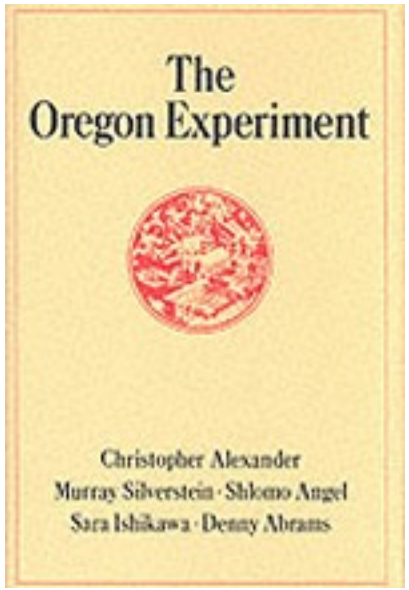


Outline

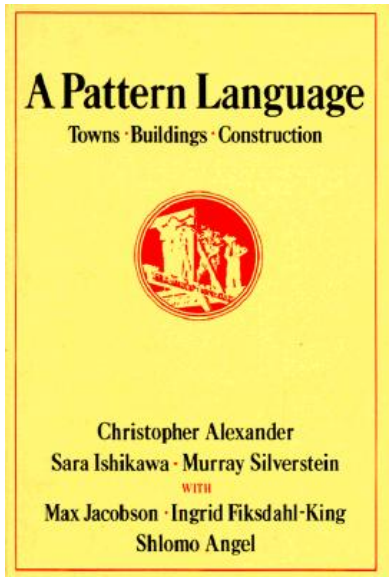
- Introduction
  - Motivation
  - Objectives
- Methodology:
  - Design patterns
  - Analyses
- Results
  - Alpha/Beta solvers
  - Navy applications
  - Scientific Impact
- Conclusions
- Future Work

Origin of Design Patterns

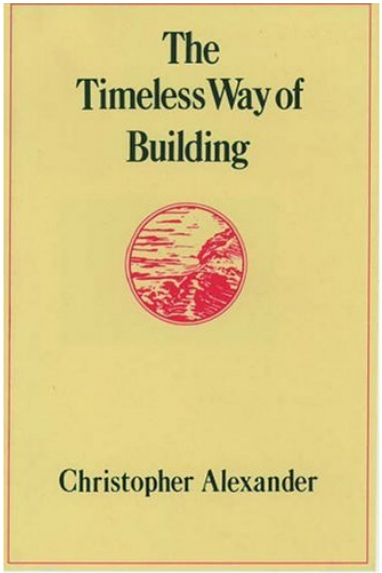
Building architecture: Alexander et al. (1975-'79)



Vol. III (1975)



Vol. II (1977)

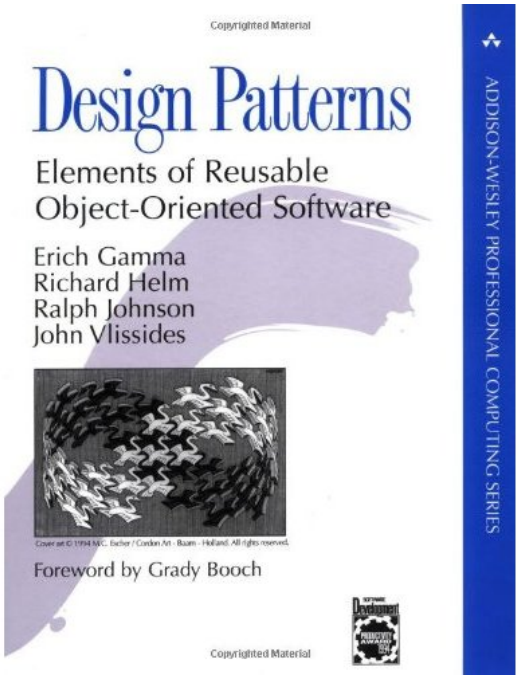


Vol. I (1979)

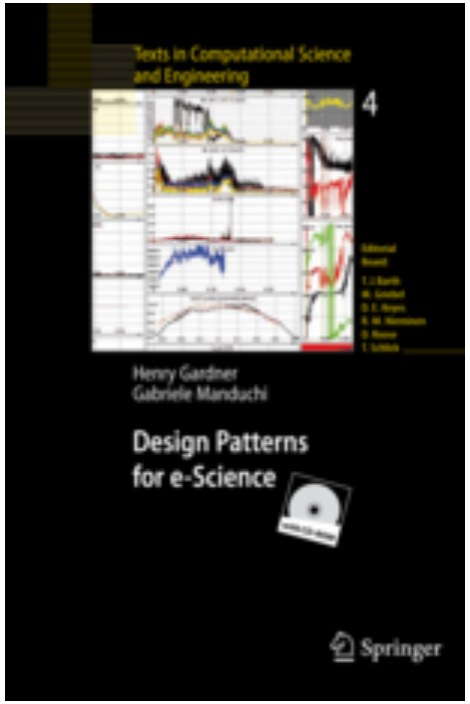
Positive Outdoor Space
Julian Street Inn
Shelter for the Homeless
San Jose, CA

OOD Patterns

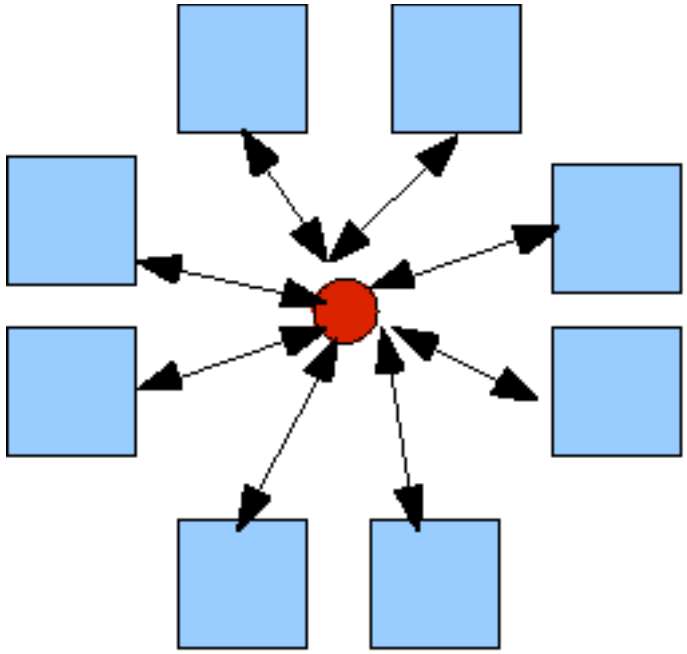
Software architecture: Gamma et al. “Gang of Four” (1995)
Scientific software architecture: Gardner & Manduchi (2007)



1995



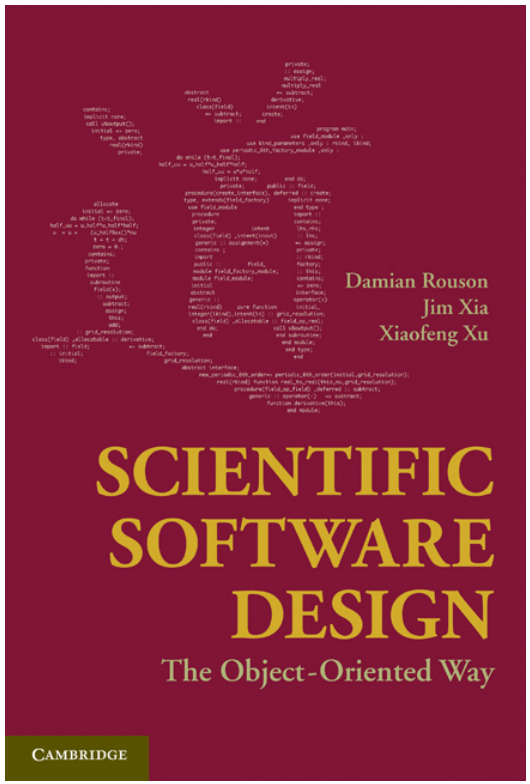
2007

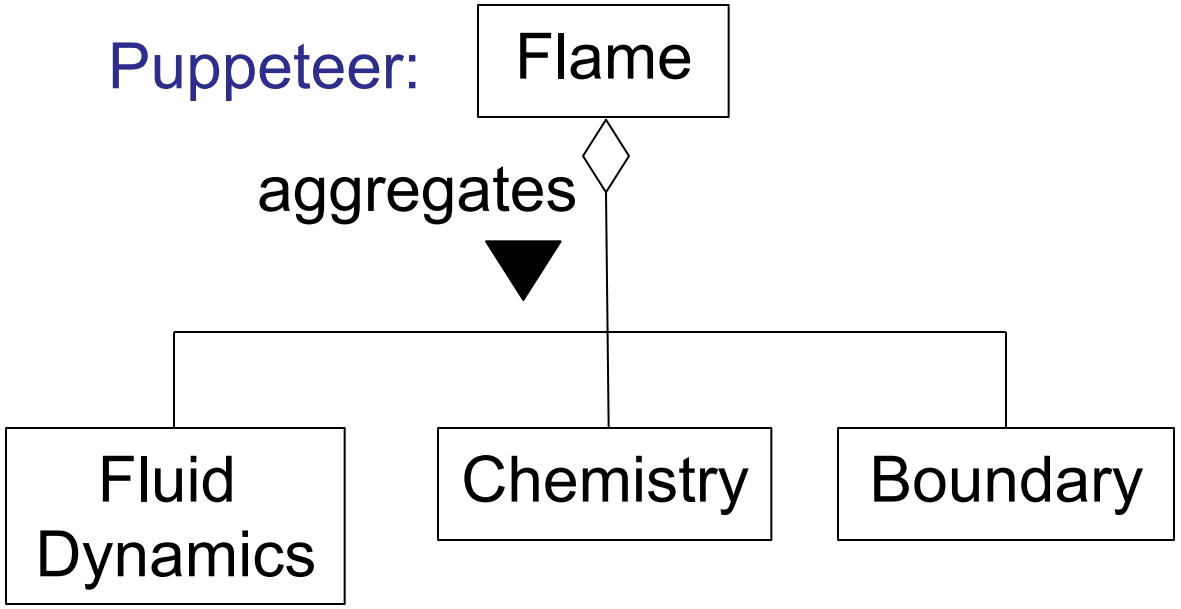


Mediator

Patterns in Fortran/C++

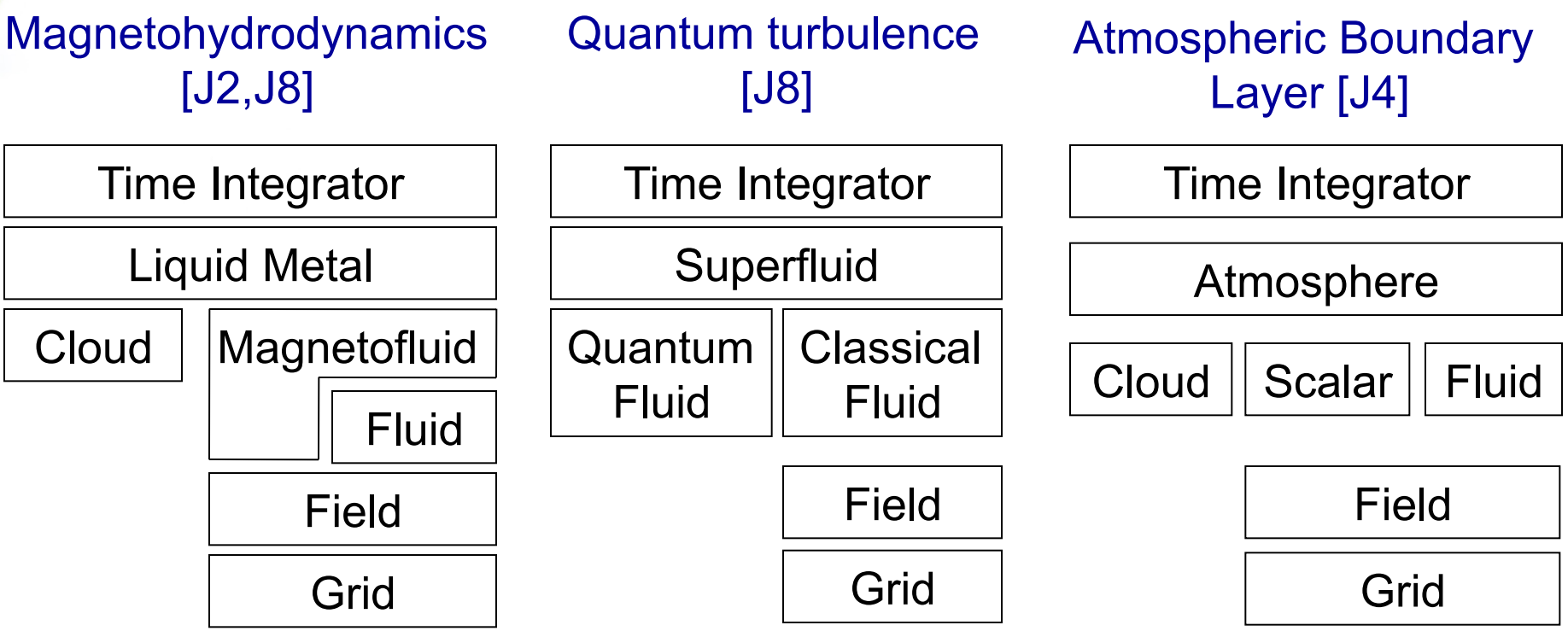
Multiphysics software architecture [J5] :
http://www.cambridge.org/Rouson





"Prefer aggregation over inheritance."
Gamma et al.

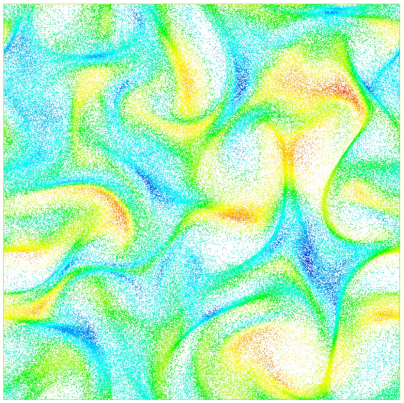
Morfeus Predecessors



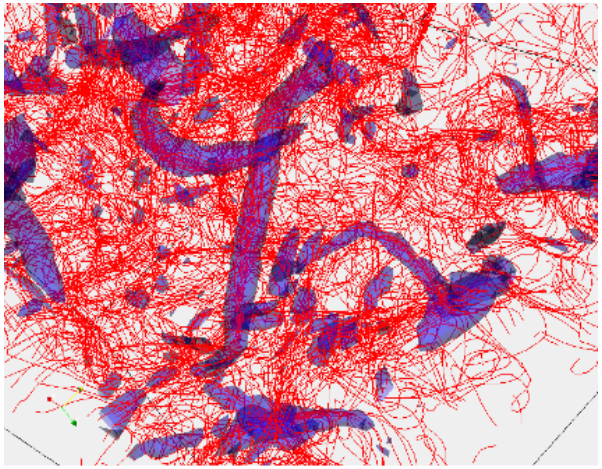
Lattice Boltzmann bio-fluid dynamics:

Xu & Lee (2008) "Application of the lattice Boltzmann method to flow in aneurysm with ring-shaped stent obstacles," *Int. J. Numerical Methods in Fluids.*

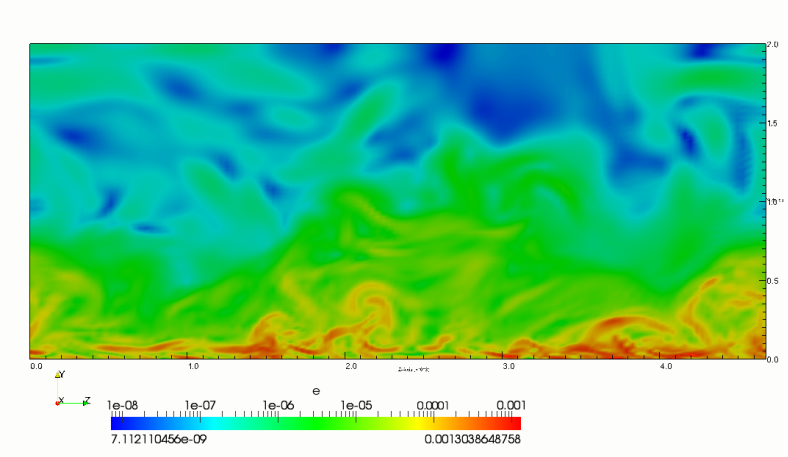
Morfeus Predecessors



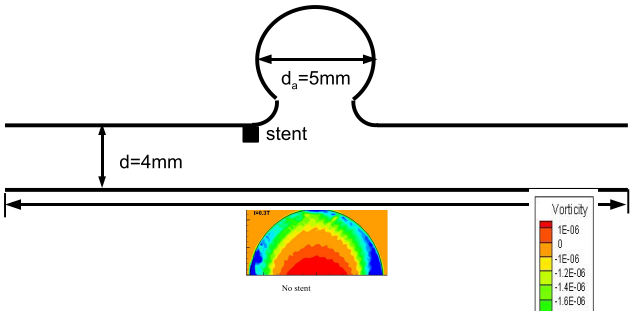
Particles in liquid metal MHD (red=fastest, blue=slowest) Rouson et al., *Phys. Fluids* 2008.



Quantum vortices (red) & classical vortices (blue) in superfluid helium (Morris, Koplik & Rouson, *Phys. Rev. Lett.* 2008).

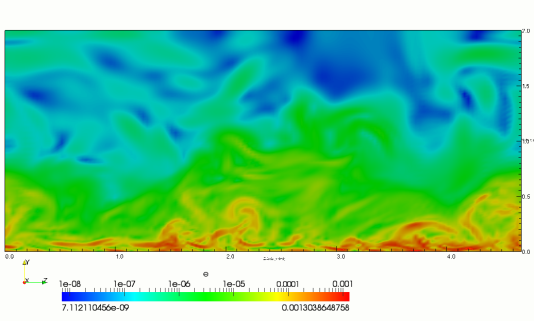


Navy application: Optical turbulence in the atmospheric boundary layer. (Morris, Handler & Rouson, *J. Turbulence*, 2010).



Flow in an aneurism with a ring-shaped stent. (Xu & Lee, *Intl. J. Num. Meth. Fluids*, 2010).

Navy Application



- Collaborator: Naval Research Laboratory
- Optical turbulence in the atmospheric layer
  - The pseudo-dissipation exhibits lognormal behavior,
  - Spatially localized regions of high and low pseudo-dissipation are found, with a magnitude ratio of about 104 between low and high regions,
  - The atmospheric boundary layer is found to be composed of a series of quasi-periodic plume-like structures, and
  - The pseudo-dissipation is found to be large at the outer edge of a typical plume, with much lower levels in the plume interior.
- Morris et al. (2010) offer conjectures on the relevance to known observations of clear-air radar scattering.

Abstract Data Type Calculus

Blackboard abstraction	Software abstraction
$u = u(x,t)$	`type(field) :: u,du_dt`
$u(x=0,t) = u_0$	`call u%boundary(x,0,u0)`
$u_t \equiv \frac{\partial u}{\partial t}$	`u%t()`
$u^{n+1} = u^n + u_t^n \Delta t$	`u = u + u%t()*dt`
$u_t = \nu u_{xx} - u u_x$	`u_t = nu*u%xx() - u*u%x()`

Abstract Calculus Pattern

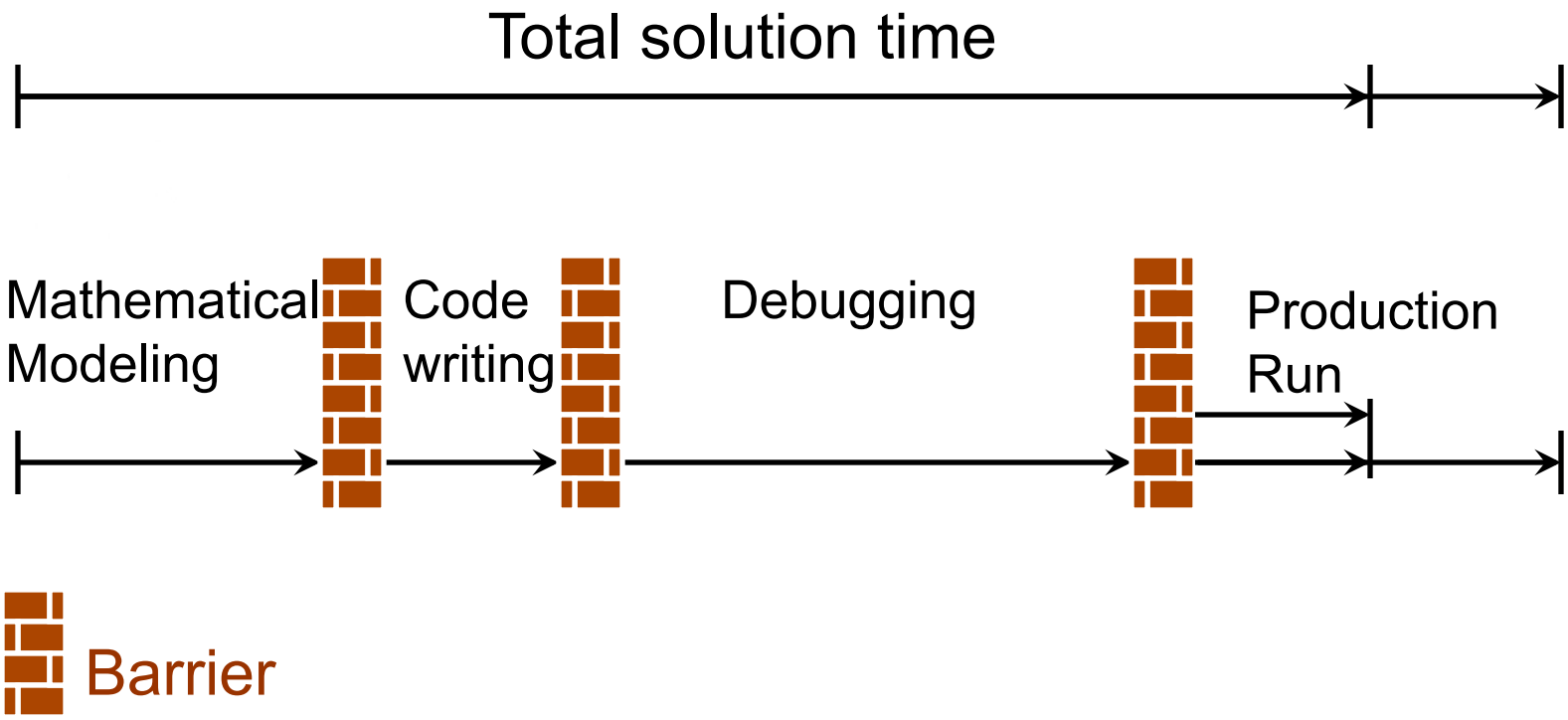
"Design to an interface, not an implementation."
Gamma et al. (1995)

```
class(field), pointer :: u,u_t
! Factory patterns (not shown)
u_t =  nu*u%xx() - u*u%x()
```

Analyses & Results

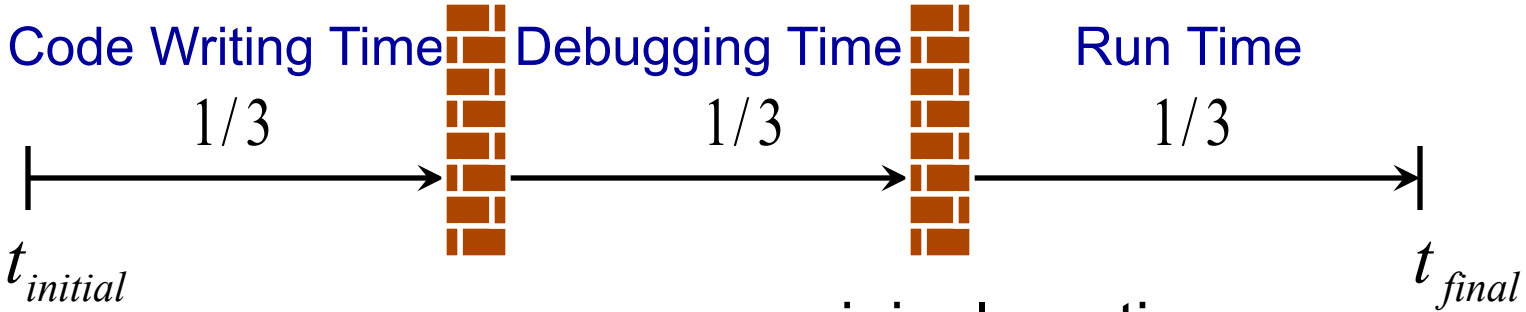
- Amdahl's Law
  - Result: focusing on runtime scalability to the exclusion of development-time scalability greatly limits speedup. [B1]
- Pareto Principle
  - Result: required runtime scalability determines % of code that can be dedicated to development-time scalability. [B1]
- Design metrics:
  - Morfeus patterns lead to high cohesion, low coupling, and low package instability [J5,J6].
- Complexity theory
  - Morfeus patterns render bug-search times roughly scale-invariant [J6].
- Information theory
  - Morfeus patterns limit the growth in developer communications as measured by interface information entropy [J6].

Conventional Development



Amdahl's Law

Representative case study for a published run [J8,J9]:



Run-time speedup: $S_{run} \equiv \dfrac{\text{original run time}}{\text{optimized run time}}$

Total speedup: $S_{tot} = \dfrac{1}{\frac{2}{3} + \frac{1}{3}\frac{1}{S_{run}}} \Rightarrow \lim_{S_{run} \to \infty} S_{tot} = 1.5$

The speedup achievable by focusing solely on decreasing run time is very limited.

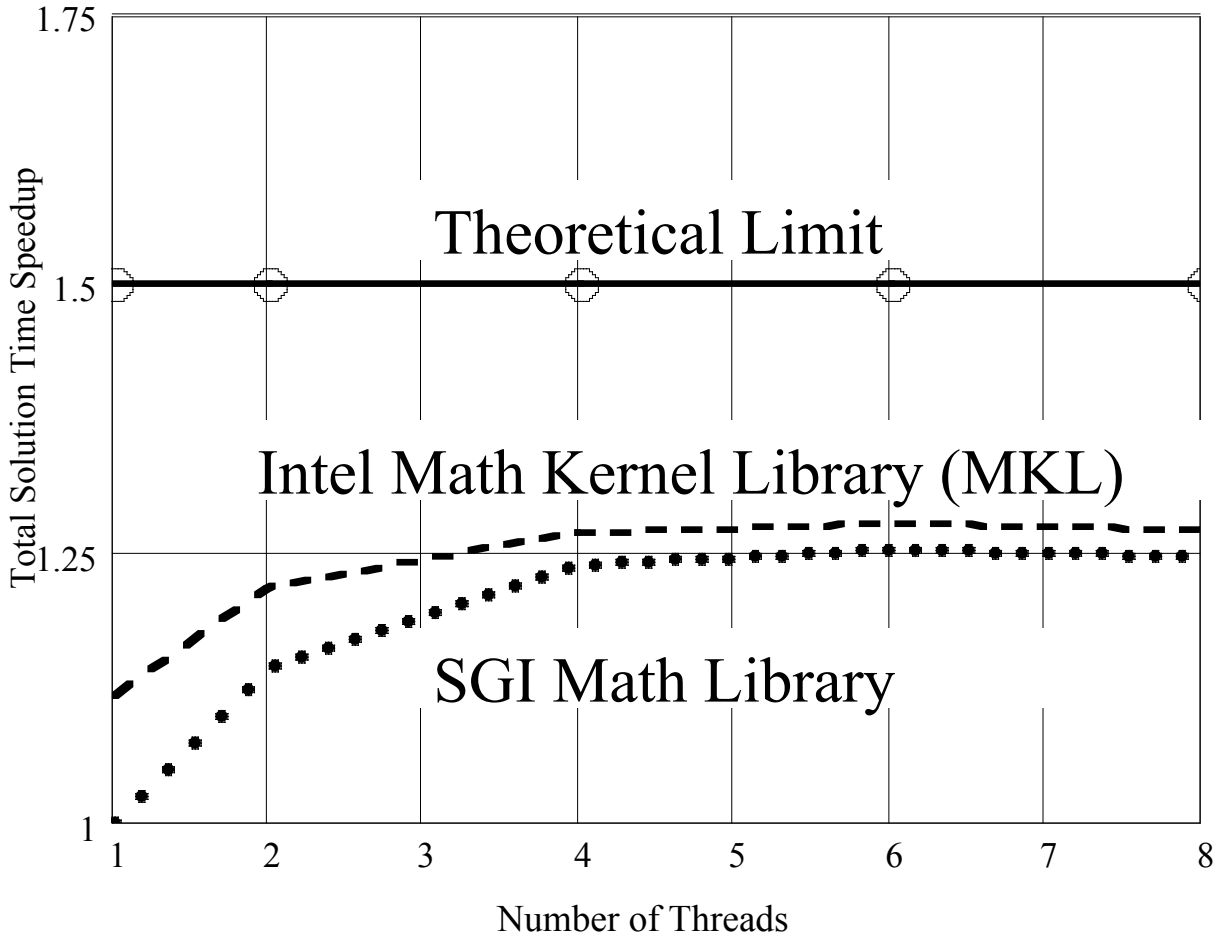
Case Study: Isotropic Turbulence

Procedure	Inclusive Run-Time Share (%)
`main`	100.0
`operator(.x.)`	79.5
`RK3_Integrate()`	47.8
`Nonlinear_Fluid()`	44.0
`Statistics_`	43.8
`transform_to_fourier`	38.7
`transform_to_physical`	23.6

Calls

- 5% procedures occupy nearly 80% of run time.
- Structure 95% of procedures to reduce development time.

Total Solution Time Speedup



Pareto Principle

When participants (lines) share resources (run time), there always exists a number $k \in [50,100)$ such that (1-k)% of the participants occupy k% of the resources:

Limiting cases:

- k=50%, equal distribution
- k→100%, monopoly

Rule of thumb: 20% of the lines occupy 80% of the run time

Scalability requirements determine the percentage of the code that can be focused strictly on programmability:

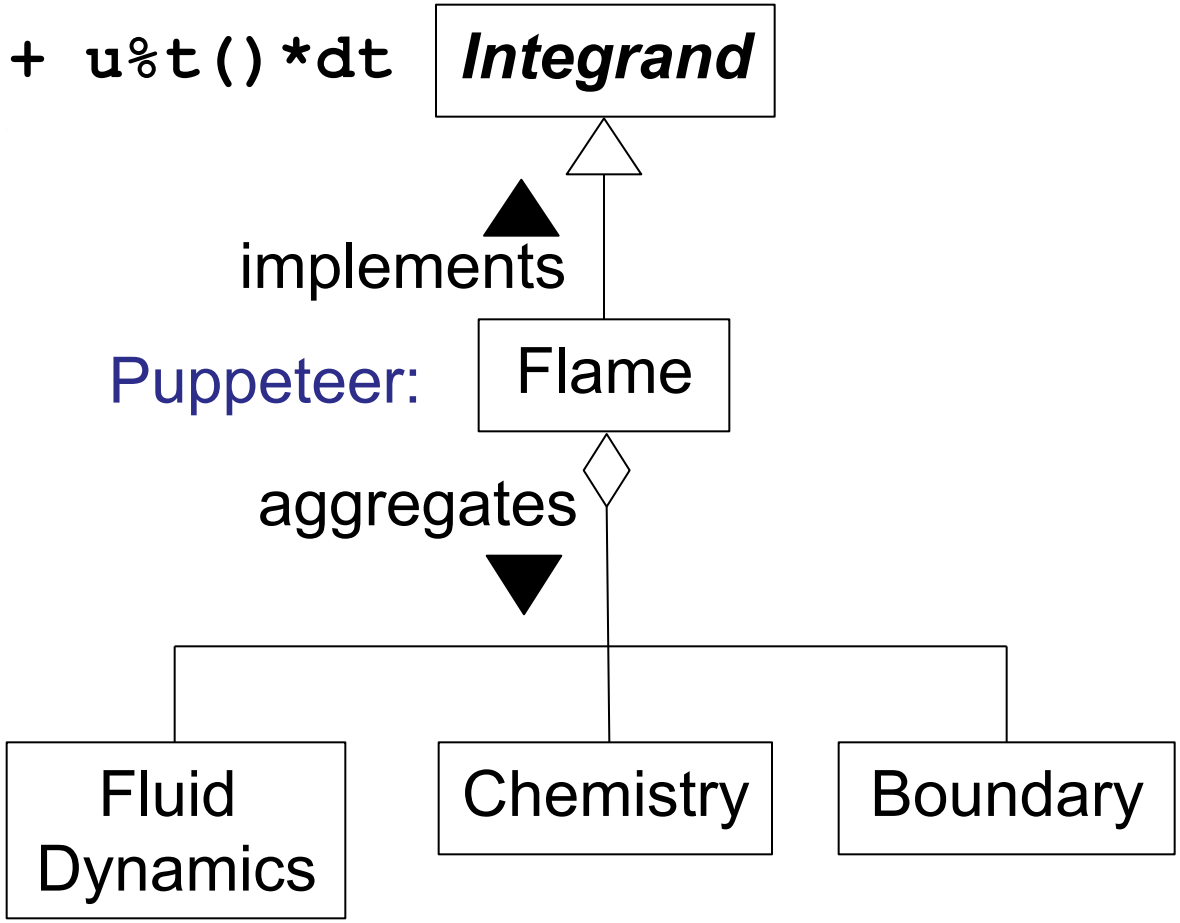
$$S_{\max} = \lim_{S_{k\%} \to \infty} \frac{1}{0.2 + 0.8 / S_{k\%}} = 5$$

Software Design Metrics

- Definition: A measure of some property of a software package or of some process associated with its development.
- Examples
  - Source Lines of Code (SLOC)
  - Cyclomatic complexity
  - Efferent couplings (Ce): # packages a given package depends on.
  - Afferent couplings (Ca): # packages that depend on a given one.
  - Instability: $$I \equiv \frac{Ce}{Ce + Ca}$$

Package Instability

```
u = u + u%t()*dt
```

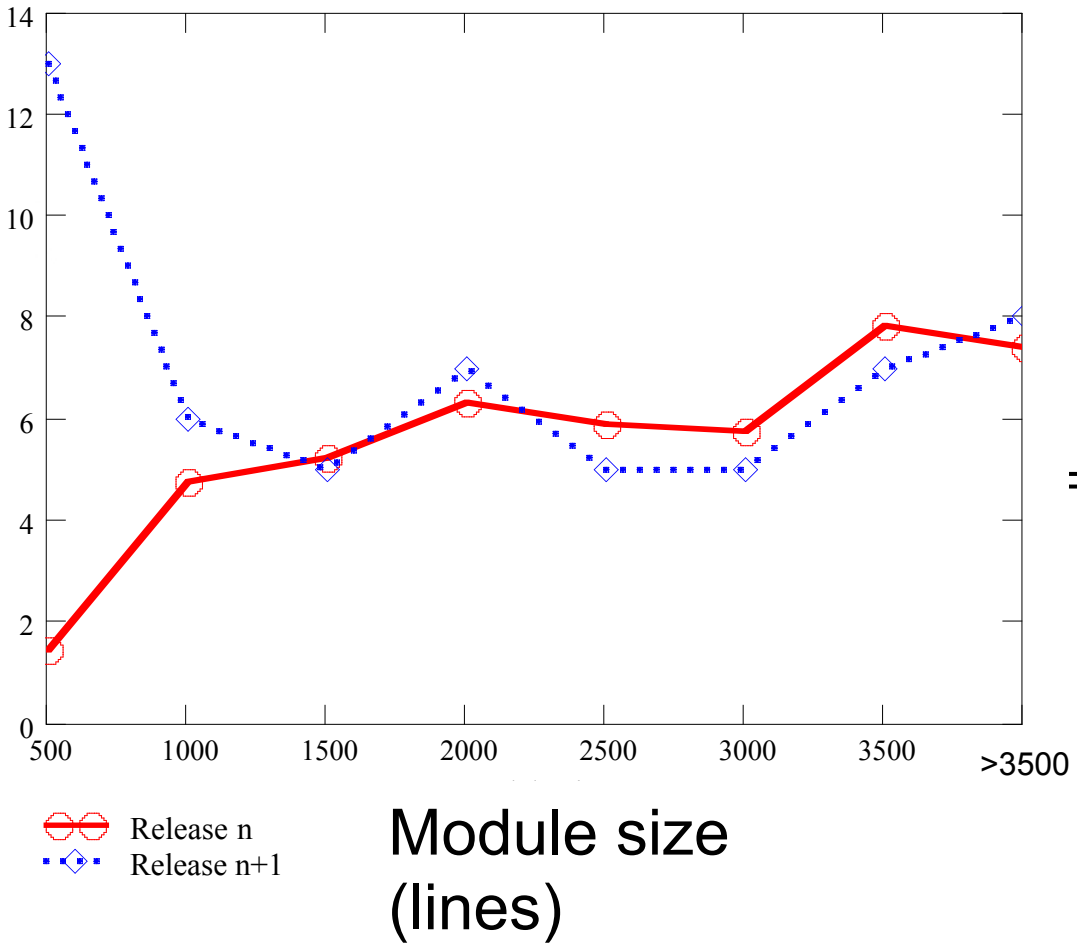


$$I = \frac{0}{1} = 0$$

$$I = \frac{4}{4+0} = 1$$

$$I = \frac{0}{1} = 0$$

Software Fault Rate



$$\Rightarrow r \approx 6/1000$$

$$\text{faults} = r\lambda$$

Source: Fenton & Ohlssen (2000) “Quantitative analysis of faults and failures in a complex software system,” *IEEE Trans. Soft. Eng.*

Scientific Code Faults

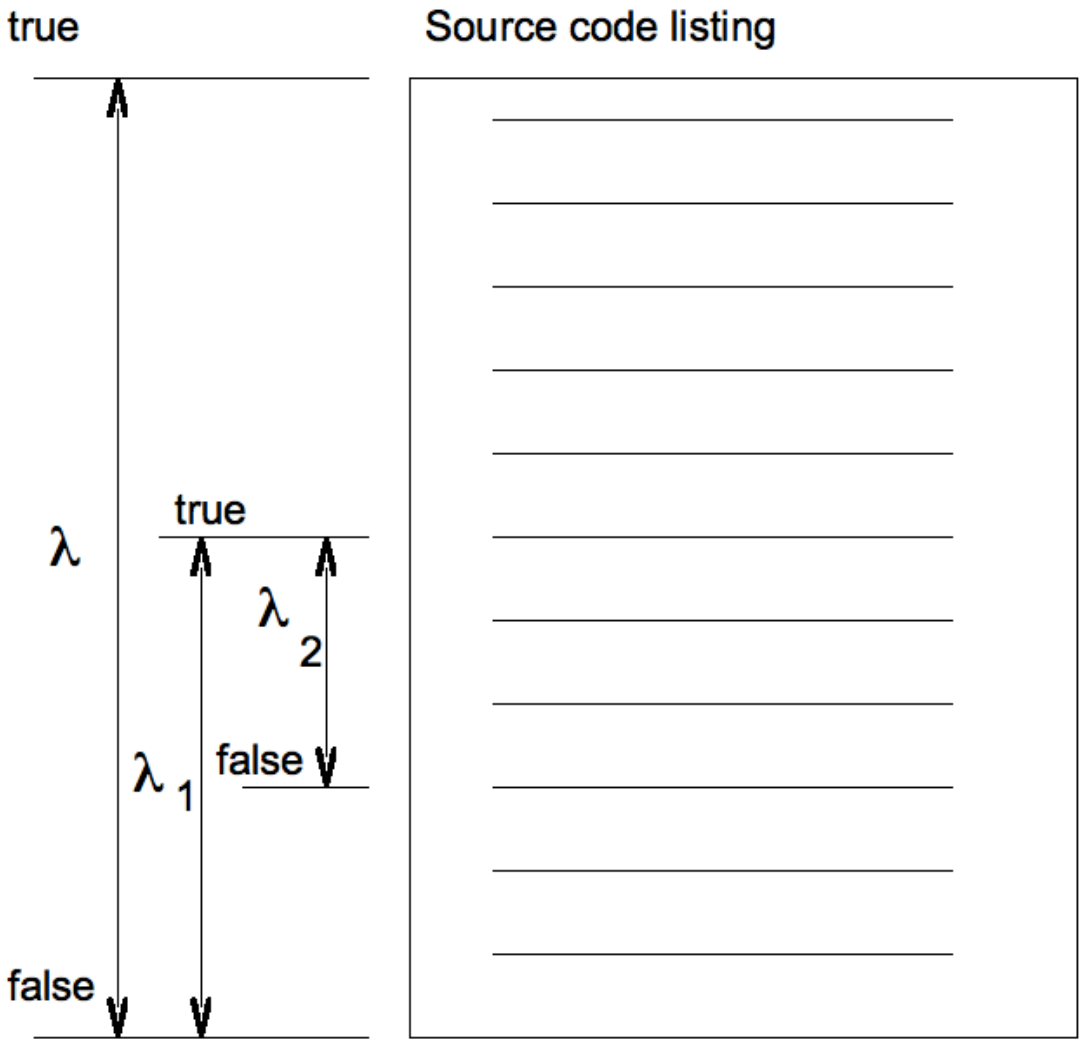
Observed faults in *commercially released* code:

- 8 statically detectable faults/1000 lines of C code.
- 12 statically detectable faults/1000 lines of Fortran 77 code.
- More recent data finds 2-3 times as many faults in C++ code.

$$\Rightarrow r \approx 0.006 - 0.036$$

Source: Hatton, L. (1997) "The `T' Experiments – Errors in Scientific Software," *Comp. Sci. Eng.*

Fault Localization Time



Search location error:

$$\lambda_n = \frac{\lambda}{2^n}$$

Convergence criterion:

$$\frac{\lambda}{2^m} = 1$$

$$m = \log_2 \lambda$$

Search time metric:

$$\lambda_{\text{searched}} = r\lambda \log_2 \lambda$$

Procedural Density

Integrand

`operator(+)(integrand,integrand) : integrand`
`operator(*)(real,integrand) : integrand`
`t(integrand) : integrand`

$$\lambda \approx \lambda_{\text{class}} = \rho p$$

$$\left.\begin{aligned} p &= 3 = const. \\ \rho &\approx const. \end{aligned}\right\} \Rightarrow \lambda_{\text{searched}} \approx const.$$

Developer Communications

Shannon (1948) "A mathematical theory of communication," *Bell System Tech. J.*

The class interfaces embody inter-developer communications. Consider the set of all (*N*) possible messages that can be transmitted between two developers:

> "If the number of messages in the set is finite, then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely."

Shannon chose the logarithm because it satisfies several constraints that match our intuitive understanding of information:

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i = -\sum_{i=1}^{N} \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N$$

Minimum Information Growth

```
subroutine integrate(integrand)
  class(integrable_model) :: integrand
  integrand = integrand + dt*integrand%t()
end subroutine
```

If only one class extends integrable_model, the executable line only has one possible interpretation, so N=0. Each subsquent subclass increases the information content by

$$\Delta H = \log_2(N+1) - \log_2 N$$

which is obviously the minimum information growth.

Outline

- Introduction
  - Motivation
  - Objectives
- Methodology:
  - Design patterns
  - Analyses
- Results
  - Alpha/Beta solvers
  - Navy applications
  - Scientific impact
- Conclusions
- Future Work

Morfeus Alpha

Blackboard abstractions (Burgers equation):

$$u_t = \nu u_{xx} - u u_x$$

$u = u(x,t)$: velocity field

ν : diffusion coefficient

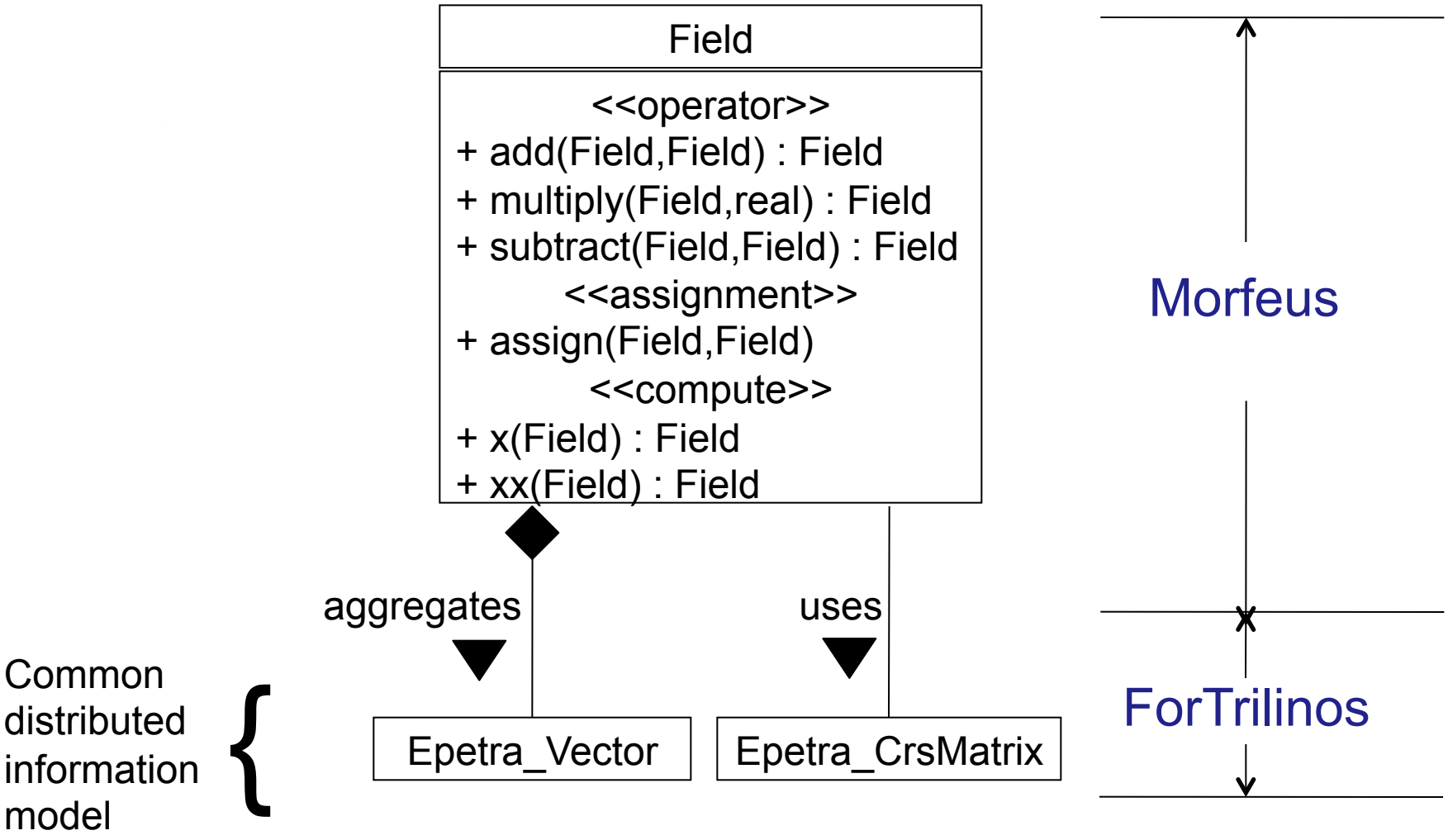
Software abstractions:

```
type(field) :: u=field(), u_t
u_t =  nu*u%xx() - u*u%x()
```

Synchronization

Asynchronous, purely functional operators and methods.

Alpha Architecture



Alpha Implementation

```
program main
!-------------- Dependencies ----------------------------
#include "ForTrilinos_config.h"
#ifdef HAVE_MPI
  use mpi
  use FEpetra_MpiComm,            only : Epetra_MpiComm
#else
  use FEpetra_SerialComm,         only : Epetra_SerialComm
#endif
  use ForTrilinos_utils,          only : valid_kind_parameters
  use iso_c_binding,              only : c_int,c_double
  use field_module,               only : field,initial_field
  use initializer,                only : u_initial
  implicit none
```

Alpha Implementation

```
!---------------- Declarations ----------------------------
#ifdef HAVE_MPI
  type(Epetra_MpiComm)    :: comm
#else
  type(Epetra_SerialComm) :: comm
#endif
  type(field)             :: u,u_t
  procedure(initial_field), pointer :: initial
!---------------- MPI Start-up ----------------------------
#ifdef HAVE_MPI
  call MPI_INIT(ierr)
  comm = Epetra_MpiComm(MPI_COMM_WORLD)
#else
  comm = Epetra_SerialComm()
#endif
```

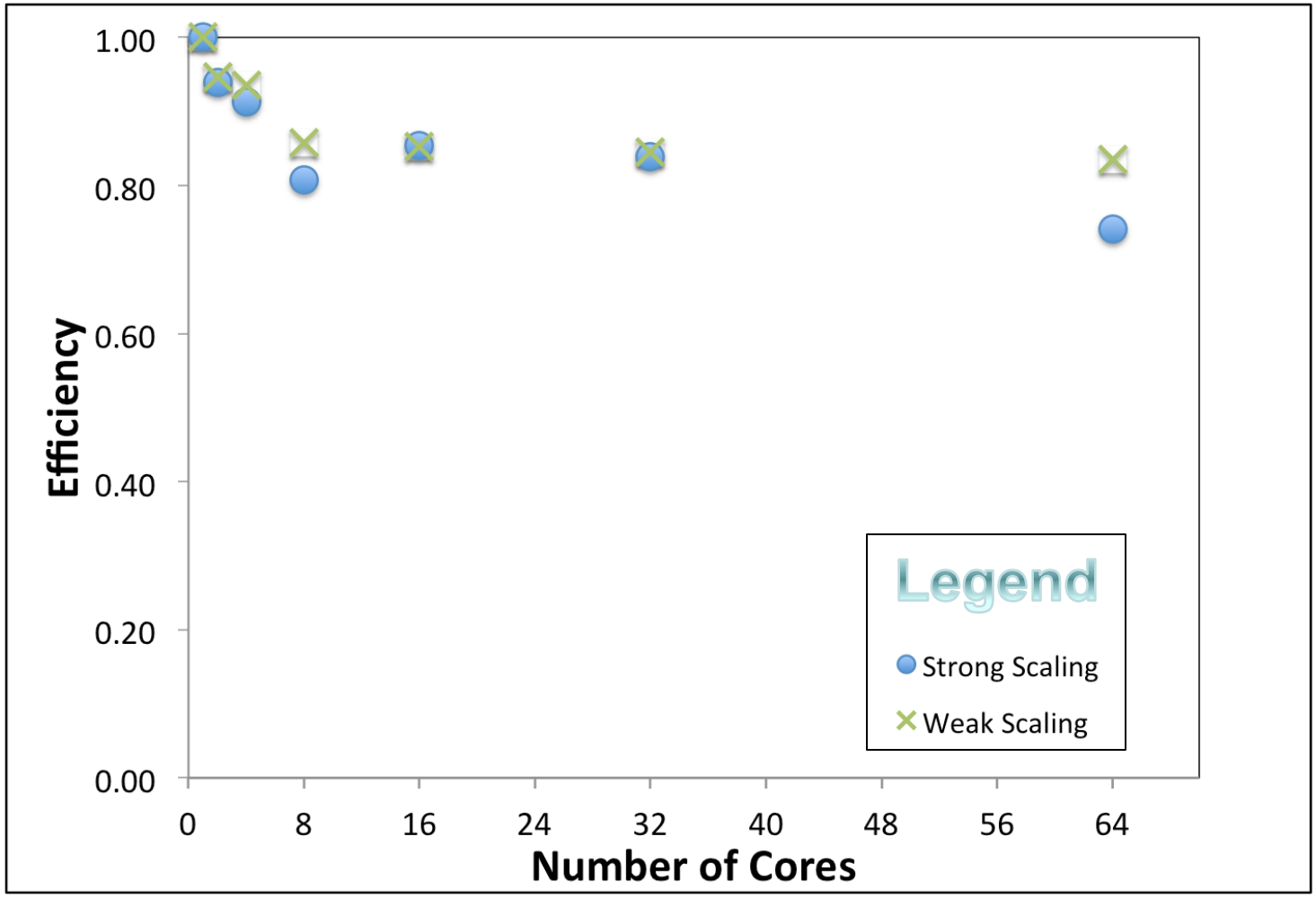
Alpha Implementation

```
!----------------- Object initialization ----------------------
  initial => u_initial
  u = field(initial,grid_resolution,comm)
!------------- Forward Euler time integration -----------------
  do tstep=1,1000
   dt = u%euler_step(nu ,grid_resolution)
   u_t = u%xx()*nu – u*u%x()
   u  = u + u_t*dt
   t = t + dt
  end do
```

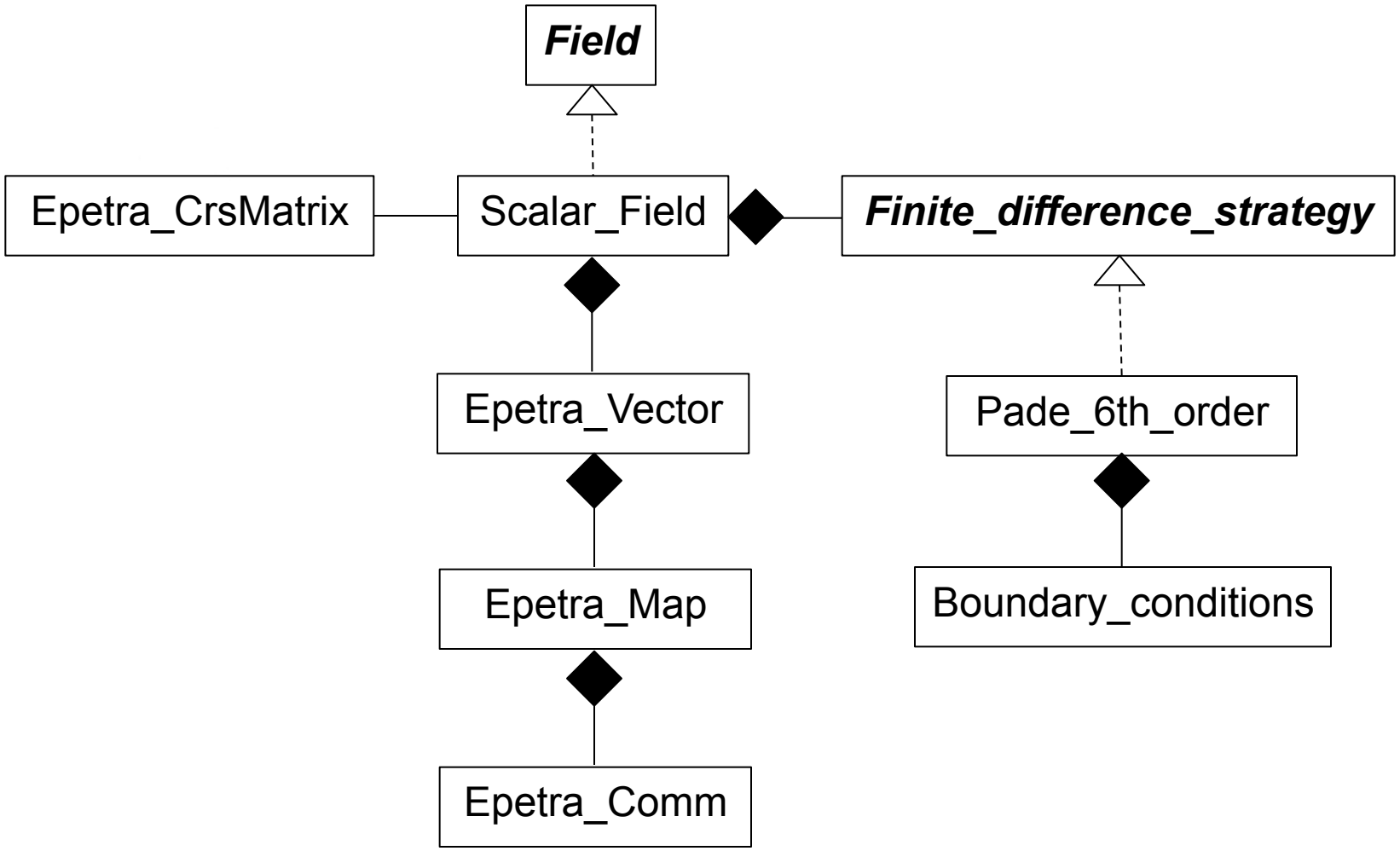
Alpha Implementation

```
!-------------------- Memory clean-up ---------------------
  call u%force_finalize
  call u_t%force_finalize
  call comm%force_finalize
!-------------------- MPI shutdown ------------------------
#ifdef HAVE_MPI
  call MPI_FINALIZE(rc)
#endif
end program
```

Alpha Solver Performance



Morfeus Beta Architecture



3D Navier-Stokes Solver

We solve a form of the 3D Navier-Stokes equations due to Kim, Moin & Moser (1987), that evolves one component of velocity (u_y) and a parallel component of vorticity $\omega_y \equiv (\nabla \times \vec{u})_y$ so that each time step involves backing out the other two velocity components via

$$\begin{bmatrix} -\frac{\partial}{\partial z} & \frac{\partial}{\partial x} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial z} \end{bmatrix} \begin{Bmatrix} u_x \\ u_z \end{Bmatrix} = \begin{Bmatrix} \omega_y \\ -\partial u_y / \partial y \end{Bmatrix}$$

where the first equation comes from the definition of vorticity and the second equation imposes the incompressibility constraint.

Compact Finite Difference Scheme

We approximate all derivatives via a compact finite difference scheme due to Lele (1992) that involves inversion of sparse linear systems of the form

$$\mathbf{A}\mathbf{f}' = \frac{1}{h}\mathbf{B}\mathbf{f}$$

so the aforementioned linear system becomes

$$\frac{1}{h}\begin{bmatrix} -\mathbf{C}_z & \mathbf{C}_x \\ \mathbf{C}_x & \mathbf{C}_z \end{bmatrix}\begin{Bmatrix} \mathbf{u}_x \\ \mathbf{u}_z \end{Bmatrix} = \begin{Bmatrix} \omega_y \\ -\mathbf{C}_y\mathbf{u}_y / h \end{Bmatrix}$$

where subscripts on the submatrices indicate the differentiation direction and where

$$C_x = \mathbf{A}_x^{-1}\mathbf{B}_x \qquad C_z = \mathbf{A}_z^{-1}\mathbf{B}_z$$

Dynamically Derived Stencil

To investigate the structure of the submatrices $\mathbf{C}_x$ and $\mathbf{C}_z$, we rewrite the RHS of a representative submatrix solution system using the identity matrix :

$$\mathbf{A}_x\mathbf{C}_x = \frac{1}{h}\mathbf{B}_x\mathbf{I}$$

Column i of $\mathbf{C}_x$ is thus the finite difference approximation to the derivative of unit vector $\mathbf{e}_i$:

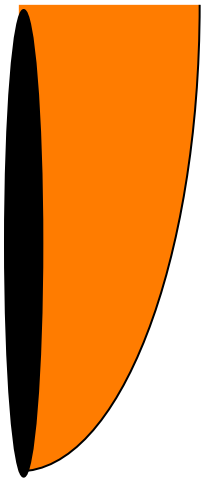
$$\mathbf{A}_x\left[\mathbf{f}'_1 \quad \cdots \quad \mathbf{f}'_N\right] = \frac{1}{h}\mathbf{B}_x\left[\mathbf{e}_1 \quad \cdots \quad \mathbf{e}_N\right]$$

where each $\mathbf{e}_i$ has only one non-zero value. Hence, its derivative, , must vanish everywhere outside of a small neighborhood of the one non-zero value. $\mathbf{C}_x$ is therefore sparse (!) but in a way that must be discovered on the fly, i.e. at runtime.

We know of no other attempts to exploit the sparsity of this matrix and we believe our ability to do so is the most significant demonstration of the flexibility of the Morfeus abstractions.

Navy Application

- Our next step involves applying this scheme to the problem of flame spread on vessel compartment walls.
- Collaboration with L. Bravo & A. Trouve', Fire Protection Engineering Dept. at University of Maryland, College Park:
  - Geometrical and physical similarity to scalar transport in the atmospheric boundary layer.
  - U Md. offers physics expertise.
  - Morfeus project offers scalable algorithms.



Impact on the Scientific Programming Community

- Two open-source projects are adopting Morfeus patterns:
  - ForTrilinos: http://trilinos.sandia.gov/packages/fortrilinos
  - PSBLAS: http://http://www.ce.uniroma2.it/psblas
- Nearly every Fortran compiler team on the planet has code from our project in their test suite:
  - IBM
  - Numerical Algorithms Group (NAG)
  - Intel
  - Cray
  - Portland Group
  - Gnu Compiler Collection (GCC)
- Short courses at domestic & international supercomputer centers:
  - HECToR (U.K.): http://www.hector.ac.uk/cse/training/oopinf2003/
  - NERSC (U.S.), March 2012

Conclusions

- Morfeus design patterns produce
  - Interoperable
  - Loosely coupled
  - Services based on a distributed information model
- Morfeus code scales well in terms of
  - Package instability (as measured by design metrics)
  - Bug search time (as measured by fault localization complexity)
  - Developer communication (as measured by information entropy)
- Morfeus code also
  - Scales up to dozens of cores at minimal development cost.
  - Advances the state of the art in modern Fortran OOP/OOD
  - Provides for impacting Navy operations in terms of understanding optical turbulence and flame spread.

Publications & Patents

Book:

[B1] Rouson, D.W.I., J. Xia, J. and X. Xu (2011) *Scientific Software Design: The Object-Oriented Way*, Cambridge University Press.

Refereed Journal articles:

[J1] Rouson, D. W. I., K. Morris. and J. Xia "Managing C++ objects with Fortran in the driver's seat: This is not your father's Fortran," *Computing in Science and Engineering*, in press.

[J2] Xu, X., Rouson, D. W. I., Kassinos, S. C. and Radhakrishnan, H. "Dispersed-phase structure in sheared MHD turbulence," *Journal of Turbulence*, in review.

[J3] Morris, K.. Rouson, D. W. I. and Lemaster, M. N. "On the scalable development of portable object-oriented Fortran interfaces to C++: A PDE solver prototype," *ACM Transactions on Mathematical Software*, in review.

[J4] Morris, K., Handler, R. and Rouson, D. W. I. (2010) "Intermittency in the turbulent Ekman layer," *Journal of Turbulence*, **12**:12, 1-25.

[J5] Rouson, D. W. I., Xia, J. and Adalsteinssohn, H. (2010) "Design patterns for multiphysics modeling in Fortran 2003 and C++,"*ACM Transactions on Mathematical Software* **37**:1.

[J6] Rouson, D.W.I. (2008) "Towards analysis-driven scientific software architecture: The case for abstract data type calculus", *Scientific Programming* **16**:4.

Publications & Patents

Refereed Journal articles (cont.):

[J7] Morris., K., J. Koplik and D. W. I. Rouson (2008) "Vortex locking in direct numerical simulations of quantum turbulence," *Physical Review Letters* **101**, 015301.

[J8] Rouson, D.W.I., S. C. Kassinos, I. Moulitsas, I. Sarris and X. Xu (2008) "Dispersed-phase structural anisotropy in homogeneous magnetohydrodynamic turbulence at low magnetic Reynolds number," *Physics of Fluids* **20**, 025101.

[J9] Rouson, D.W.I., Rosenberg, R., Xu, X., Moulitsa, I. and Kassinos, S.C. (2008) "A grid-free abstraction of the Navier-Stokes equations in Fortran 95/2003," *ACM Transactions on Mathematical Software*, **34**:1.

Open-Source Software:

[S1] ForTrilinos, http://trilinos.sandia.gov/packages/ForTrilinos, © 2010, Sandia National Laboratories.

Patents:

[P1] System and method for reference counting with user-defined structure constructors, U.S. patent application 13/197,118 (pending), filed 3 August 2011.

Publications & Patents

Refereed Conference Papers:

[C1] Barbieri, D., Cardellini, V., Filippone, S., and Rouson, D. (2011) "Design Patterns for Scientific Computations on Sparse Matrices," *Euro-Par 2011 Parallel Processing: 17th International Euro-Par Conference*, Bordaeux, France, Aug. 29 – Sep. 2.

[C2] Morris, K., D. W. I. Rouson, and J. Xia (2011) "On the object-oriented design of reference-counted shadow objects in Fortran 2003," *4th Intl. Workshop on Software Engineering for Computational Science & Engineering*, Honolulu, Hawaii, May 28.

[C3] Rouson, D. W. I., J. Xia and X. Xu, (2010) "Object construction and destruction design patterns in Fortran 2003," *International Conference on Computational Science 2010*, Amsterdam, Netherlands, May 31–June 2.

[C4] Akylas, E.E., S. C. Kassinos, D. W. I. Rouson, and X. Xu, (2009) "Accelerating stationarity in linearly forced isotropic turbulence," *The Sixth International Symposium on Turbulence and Shear Flow Phenomena*, Seoul, Korea, June 22-24.

Collaborators

- R. Handler, Texas A&M University (formerly NRL)
- S. Kassinos et al., University of Cyprus
- S. Filippone, Universitas di Roma Tor Vergata
- J. Xia, IBM Canada
- X. Xu, General Motors
- M. Haveraaen, University of Bergen, Norway
- J. Flores, University of Puerto Rico, Mayaguez
- L. Bravo and A. Trouve', University of Maryland, College Park
- S. Garrick, University of Minnesota