

# Existing Code Analysis Team



**Douglass Post, Team Leader**

*HPCS*

DoD High Performance Computing Modernization Program

**Richard Kendall**

**Los Alamos National Laboratory**

For the team: **Jeff Carver**-Miss.State, Sunita Chulani-IBM, Kemal Ebcioğlu-IBM, Stuart Faulk-U.of Ore., Andy Funk-Lincoln, Howard Gordon-NSA, **Christine Halverson**-IBM, **Richard Kendall**-LANL, Jeremy Kepner-Lincoln, Bob Lucas-ISI, Bill Mann-Lincoln, **Andrew Mark**-DoD, Theresa Meuse-Lincoln, David Mizell-Cray, **Dolores Shaffer**-STA, Dale Shires-DoD, **Susan Squires**-SUN, Mike van de Vanter-SUN, Jeff Vetter-ORNL, Larry Votta-SUN, Trey White-ORNL, Clay Williams-IBM

**DARPA HPCS Workshop**

**SC2005, Seattle, Washington, November 14, 2005**



Office of Science  
U.S. Department of Energy



OAK RIDGE



Nov. 14, 2005



# The Total Productivity Challenge

- All the other talks describe how to address the Performance and Parallel Programming Challenge
- Our team addresses both the Parallel Programming and the **Prediction Challenge**
  - Developing applications programs of high complexity
  - Codes that have
    - High resolution in time and space
    - Include all the important effects
    - Model a complete system
  - Codes that produce the answers that the DoD, NASA, DOE, industry are paying to get
- Case studies identify the nature of the challenges
  - Develop ways to overcome these challenges

# Case Studies provide information on how computational science and engineering is accomplished and how to improve the process and the product.

- Improve productivity prediction
  - Learn how to measure and predict productivity
  - **Build productivity into the procurement process**
- Improve code development process and products

Characterization of code projects provides key productivity information on:

  - Requirements, goals and deliverables
  - Project Characteristics, Structure, and Organization
  - Code Characteristics and Structure
  - Staffing
  - Workflows
    - Identify bottlenecks, roadblocks and impediments
    - Identify best tools and methodologies
  - Testing → verification and validation
  - Success measurements
  - Lessons Learned
    - What works and what doesn't

# Completing Four Case Studies.

	<b>Falcon</b>	<b>Hawk</b>	<b>Condor</b>	<b>Eagle</b>
Identify Project	√	√	√	√
Negotiate study with sponsor	√	√	√	√
Complete pre-interview questionnaire	√	√	√	√
Analyze questionnaire and plan on-site interview	√	√	√	√
Conduct on-site interview	√	√	√	√
Analyze on-site interview and integrate with questionnaire	√	√	√	√
Conduct follow-up to resolve unanswered questions	√	√	√	
Write report	√	√	√	
Present/publish report	√			

# Case Studies beginning to “Span” the community

	Falcon	Hawk	Condor	Eagle
Application Domain	Product Performance	Manufacturing	Product Performance	Signal Processing
Project Duration	~10 years (since 1995)	~6 years (since 1999)	~20 years (since 1985)	~3 years
Number of Releases	9 Production	1	7	1
Earliest Predecessor	1970s	early 1990s	1969	?
Staffing	15 FTEs	3 FTEs	3-5 FTEs	3 FTEs
Customers	<50	10s	100s	Demonstration code
Nonimal Code Size	~405,000	~134,000	~200,000	<100,000
Primary Languages	F77 (24%), C (12%)	C++ (67%), C (18%)	Fortran 77 (85%)	C++, Matlab
Other Languages	F90,Python,Perl, ksh/csh/sh	Python, Fortran 90	Fortran 90, C, Slang	Java Libraries(~70%)
Target Hardware	Parallel Supecomputers	Parallel Supercomputers	PCs to Parallel Supercomputers	Embedded App Demonstration
Status	Production	Production ready	Production	code
Sponsors		DoD	DoD	DoD

# Development Objectives

	Falcon	Hawk	Condor	Eagle
Parallel Scalability	Medium	High	Medium	
Execution Time	Medium	Medium	Medium	
Portability	High	High	High	
Speed to Solution	High	Medium	Medium	
Code Reuse	Medium	High <sup>†</sup>	Medium	
Complexity	Medium	Medium	High	
Maintainability	High	Low <sup>‡</sup>	High	

<sup>†</sup> For new code

<sup>‡</sup> Survey response; the code is highly maintainable, but this was not an explicit design goal

# Software Development Approach

- **FALCON**

- Increasingly formality as the project has matured and the team size has grown
- Object-oriented F77 framework with 8 other languages represented
- Growing role for formal software engineering
- Managed to CMM level 2 (no formal certification)



- **HAWK**

- Formal Project Plan– a “development” project from the beginning
- C++ framework with C solvers
- Strong role for formal software engineering
- Managed to CMM level 2 (no formal certification)



# Approaches vary among projects.

- **CONDOR**

- Informal Project Plan—started life as an engineering research project
- F77 → F90 code
- Weak role for formal software engineering (“agile” team philosophy)
- Managed to CMM level 2 (would be hard to verify)



- **EAGLE**

- Informal Project Plan—this was a demonstration development project for embedded software
- C++ code
- Weak role for formal software engineering (“agile” team philosophy)
- CMM-2 level project (would be hard to verify)



# We found sparse use of Software Metrics.

<b>Metric</b>	<b>Falcon</b>	<b>Hawk</b>	<b>Condor</b>	<b>Eagle</b>
Lines of code	x	x	x	x
Function points	x			
Stories, project velocity				
Cyclomatic complexity				
Data coupling				
Comment lines				
Locality				
Concurrency				
Defect rates				
Time-to-fix defects		x		
Number of debug runs/unit time				
Test Coverage	x	x	x	
Frequency that regression testing uncovers problems	x			
Code performance	x	x	x	x
Degree of performance optimization	x			x
Parallel scaling		x	x	x
Number of users		x	x	
Number of production runs/unit time				
Computer time for code development/unit time				
Computer time for production/unit time			x	

# All of the projects made use of Testing.

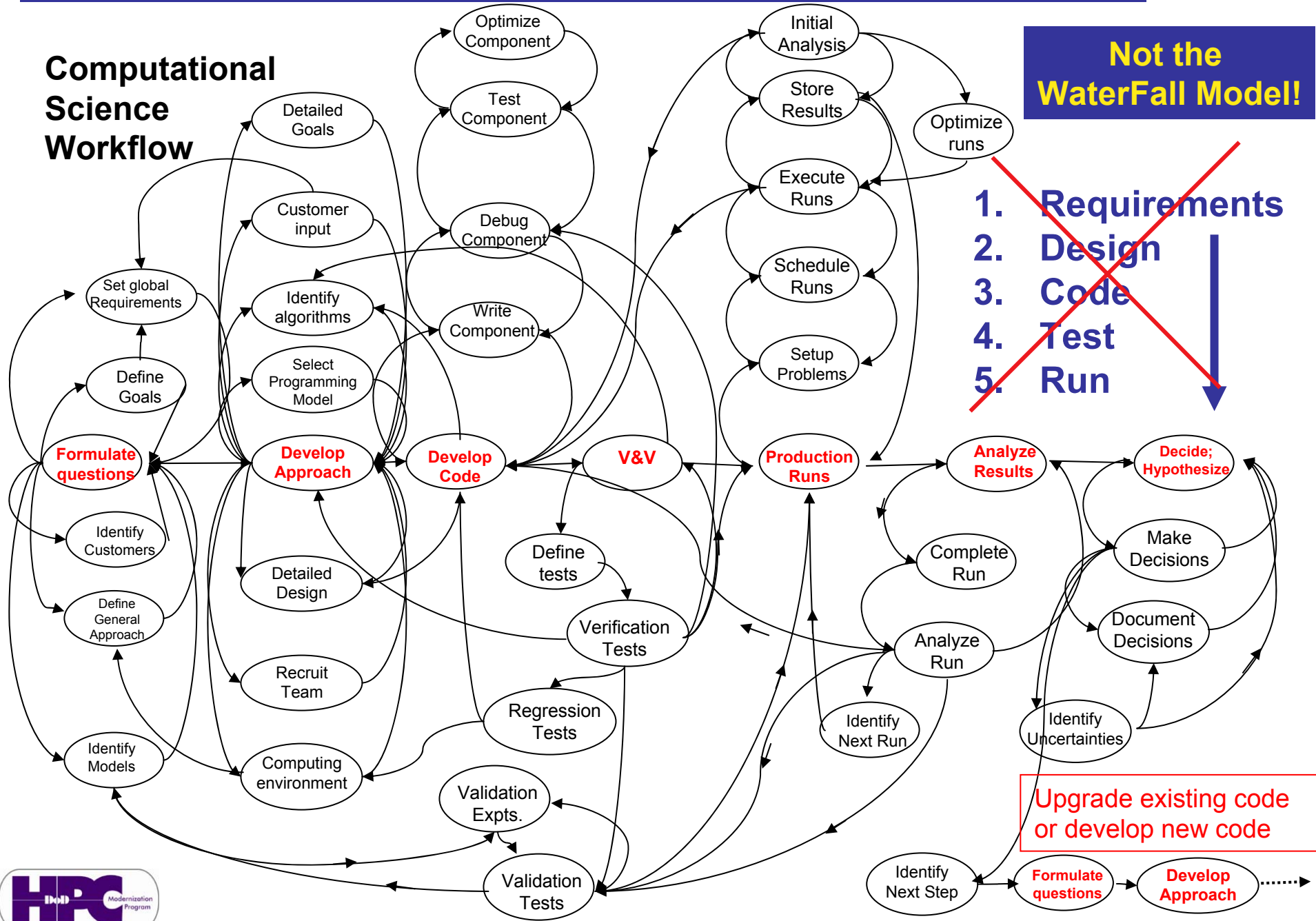
Case Study	Falcon	Hawk	Condor	Eagle
Fraction of Code Tested	43%	51-75%	51-75%	>75%
Conformance between scalar and parallel	yes, no formal bounds	< 2%	yes, no formal bounds	
Conformance with experimental tests	yes, no formal bounds	<32%	yes, no formal bounds	
Verification				
• Compare to exact answer	yes	yes	yes	
• Monitor conserved quantities	yes	yes	yes	
• Preservation of symmetries	yes	yes	yes	
• Compare with existing codes	yes	yes	yes	yes
• Controlled experiments				yes
Regression Tests	yes	no	yes	

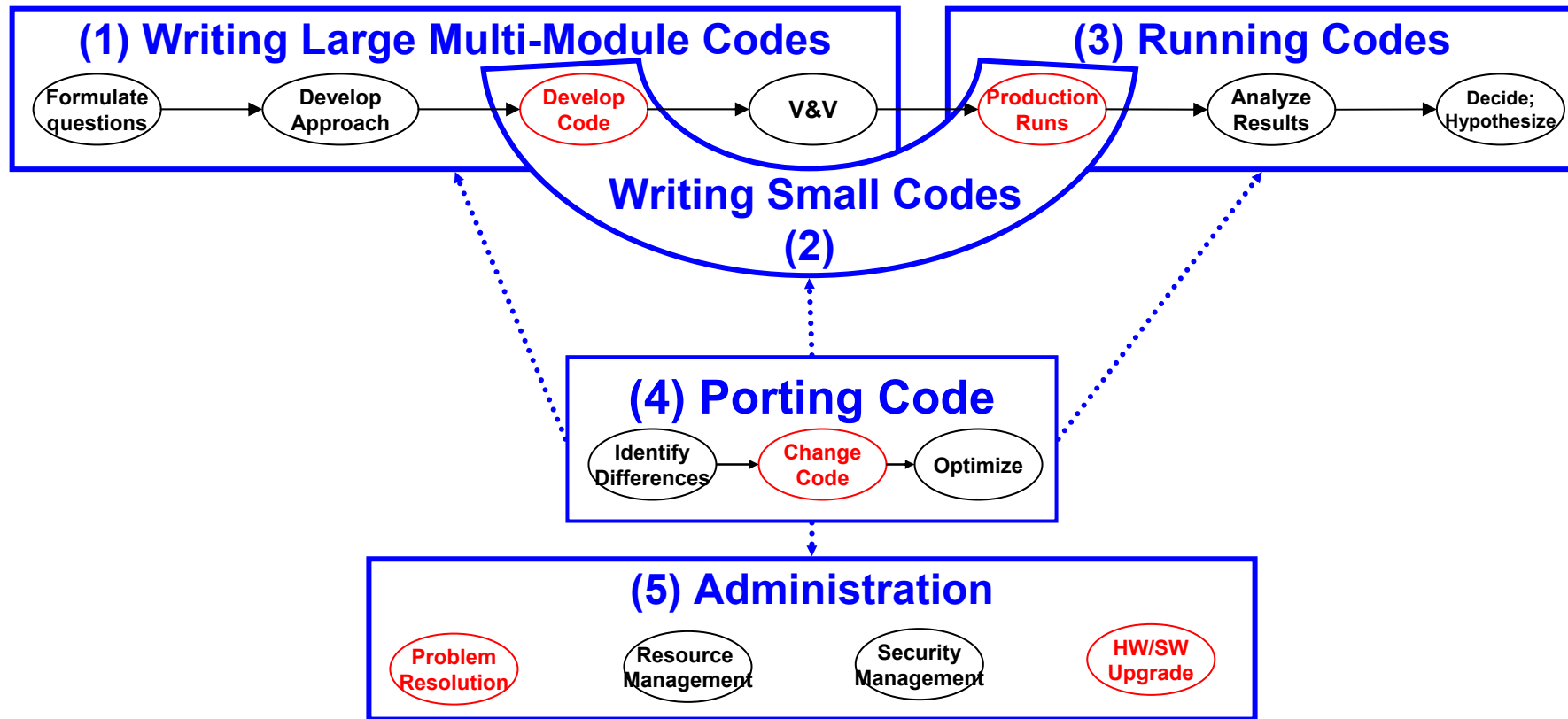
# Our Workflows are being used by the Productivity Team and Vendors.

## Computational Science Workflow

**Not the WaterFall Model!**

- ~~1. Requirements~~
- ~~2. Design~~
- ~~3. Code~~
- ~~4. Test~~
- ~~5. Run~~

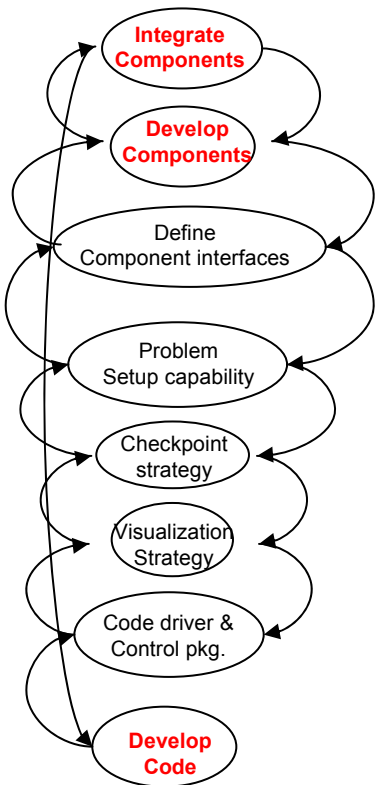




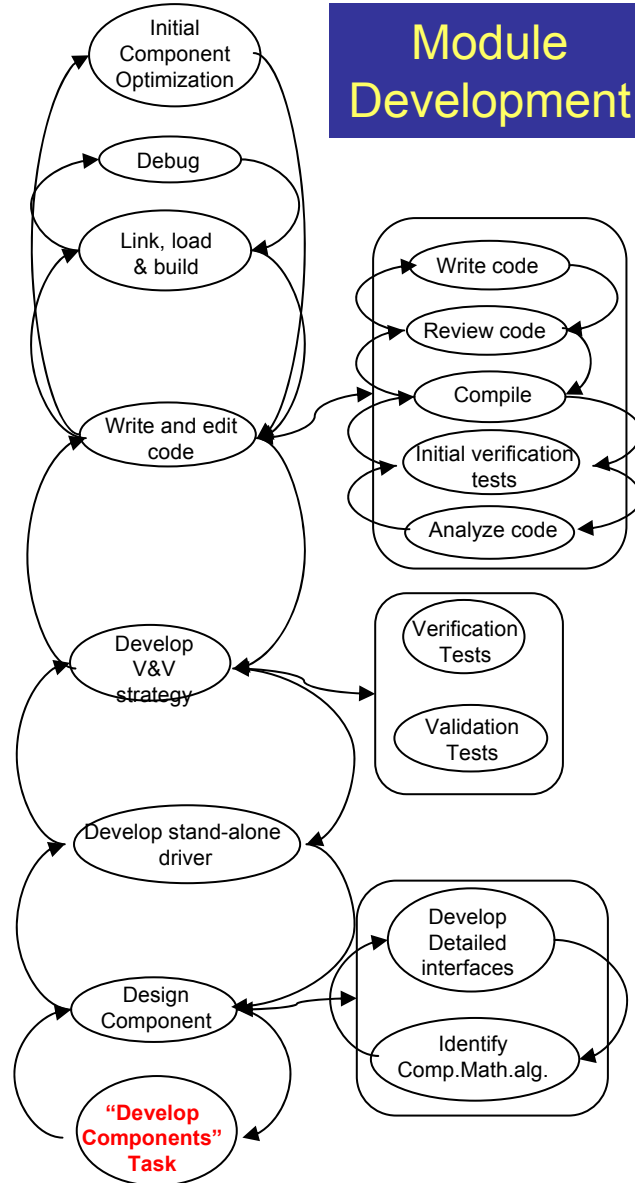
- Workflows comprise many steps; many overlapping
- Item in red represent areas with highest HPC specific interest

# Code Development Involves nested development of code modules, integration of code modules and development of integrated code

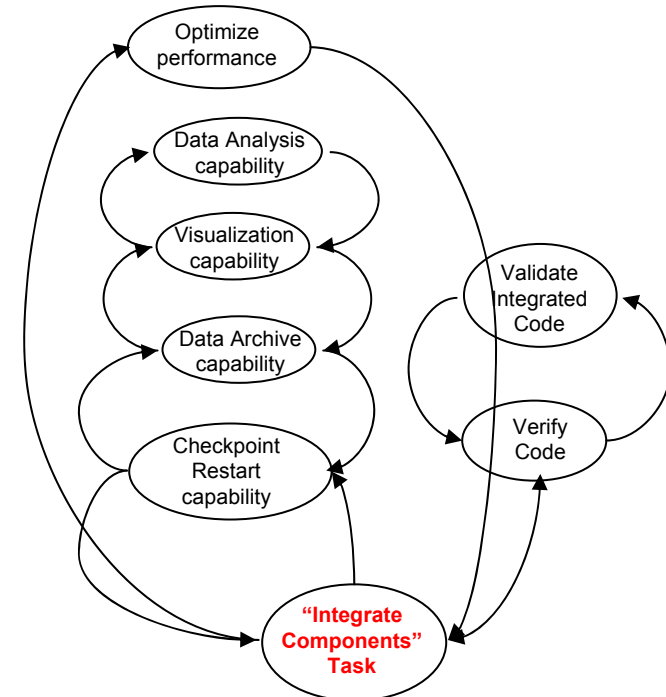
## Main code



## Module Development

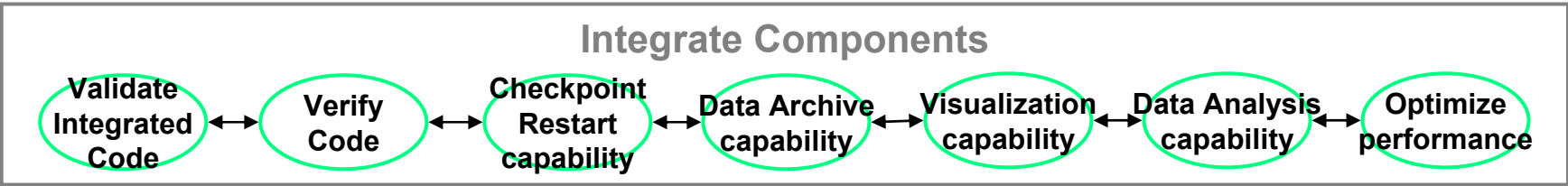
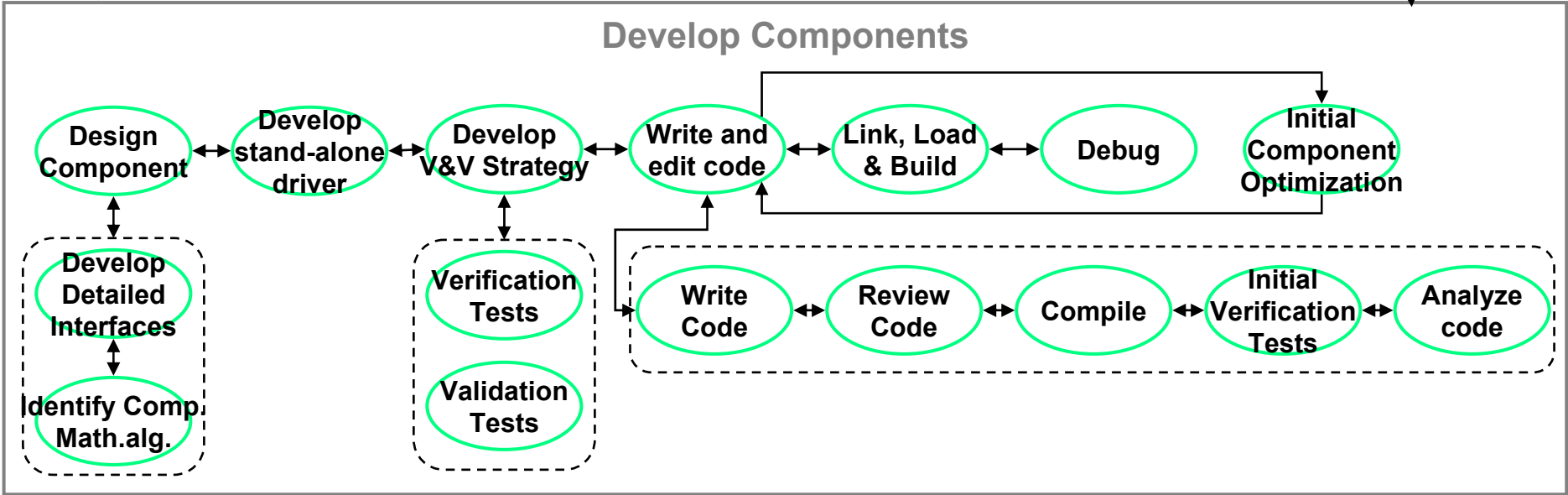


## Module Integration





# Multi-Module Development

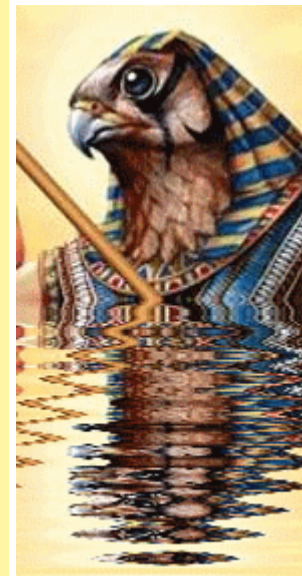


# Software Development Tools were identified.

Case Study	Falcon	Hawk	Condor	Eagle
<b>Code Development Environment</b>				
<b>Compilers</b>	F77, F90, C	C++,C, Fortran,Java	F77, F90	C++, Matlab,Java
<b>Scripts</b>	Perl,Python,ksh,csh,sh, SCHEME,Gmake	Python	None	csh,perl,make, cmake.ANT
<b>Debuggers</b>	TotalView, SourceForge	Valgrind, gbd	TotalView, gbd	TotalView, gbd, DBX
<b>Performance Monitoring</b>	Pixie,DCPI,Speedshop, Prof	Speedshop, PAPI	None	Mercury TATL
<b>Domain Decomposition</b>	Metis			
<b>Execution Environment</b>				
<b>Element Generation</b>	CAD ProE		In-house tools	N/A
<b>Visualization</b>	ICE,VTK, Paraview, Tecplot CEI Ensignt, Paraview			Matlab
<b>Data Analysis</b>	XDMF (supports Paraview)			Matlab
<b>Code Development Process Tools</b>				
<b>Configuration Management</b>	CVS	CVS	CVS	Perforce, Subversion
<b>Bug Tracking</b>	Custon(~Bugzilla)		None	no formal system
<b>Code Documentation</b>	Web-based	Doxygen	MS Word	In-code comments
<b>Support Libraries</b>				
<b>Computational Mathematics</b>	PETc, VSS,PSPASES,CG		In-House tools	FFTs
<b>Parallel Programming Libraries</b>	MPI	MPI	MPI	MPI, PVL (~POOMA)

# We developed further lessons learned.

- Computational Scientific and Engineering projects don't have detailed requirements, need "agility" to develop solution methods, identify necessary modules,...
- Customers, not the marketing department, project team or sponsor, determine usefulness of code
  - Interaction with and support of customers is of paramount importance.
- Higher level languages (e.g. C++) can be useful for HPC if deployed conservatively (minimal use of templates, levels of inheritance, etc.)
- Portability is **essential** for long life cycle codes



2005

SC2005 DARPA HPCS



16

# Plans for 2006

- Complete 5 to 6 case studies to increase span of projects, including:
  - Geo-physics project
  - Atomic/Molecular project
  - Academic research project
  - Commercial Engineering project
  - Project developed at a DoD laboratory
- Complete initial comparative project analysis
- Identify better metrics and start process of gathering them-work with Development Experiments team
- Characterize issues of distributed, non-collocated “collaborative” projects
- Continue to refine workflows for Phase III vendor support
- Support development of productivity models