Overview of Best Practices in HPC Software Development

David E. Bernholdt
Oak Ridge National Laboratory

Anshu Dubey, Katherine M. Riley
Argonne National Laboratory

Better Scientific Software Tutorial
SC19, Denver, Colorado

License, Citation and Acknowledgements

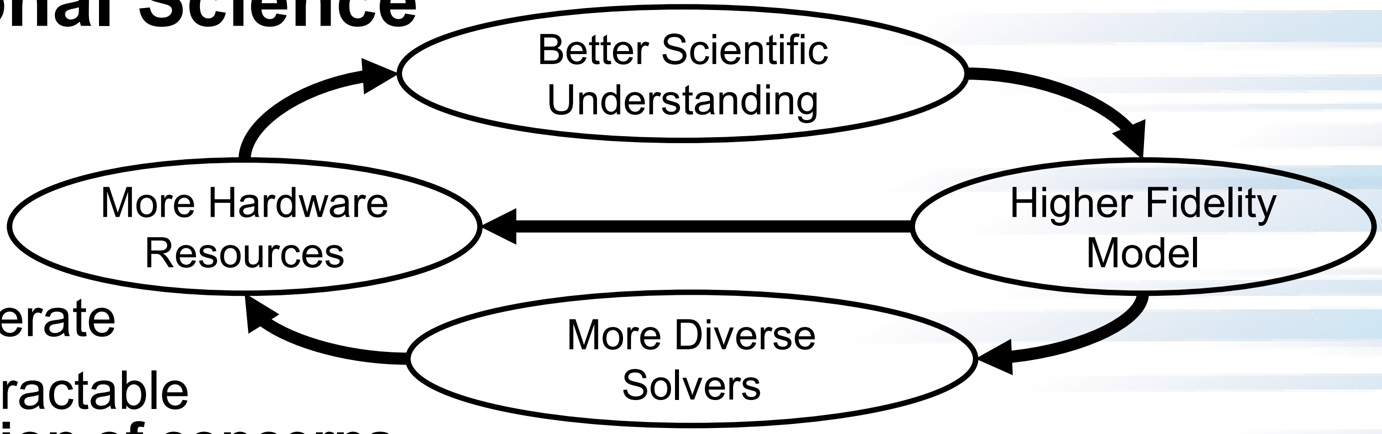
License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Michael A. Heroux, and Jared O'Neal, Better Scientific Software tutorial, in SC '19: International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, Colorado, 2019. DOI: 10.6084/m9.figshare.10114880**
- Individual modules may be cited as *Module Authors, Module Title*, in Better Scientific Software Tutorial…

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

The Success of Computational Science Creates the Challenges of Computational Science



- Positive feedback loop
  - More complex codes, simulations and analysis
  - More moving parts that need to interoperate
  - Variety of expertise needed – the only tractable development model is through **separation of concerns**
  - **It is more difficult to work on the same software in different roles without a software engineering process**
- Onset of higher platform heterogeneity
  - Requirements are unfolding, not known *a priori*
  - **The only safeguard is investing in flexible design and robust software engineering process**

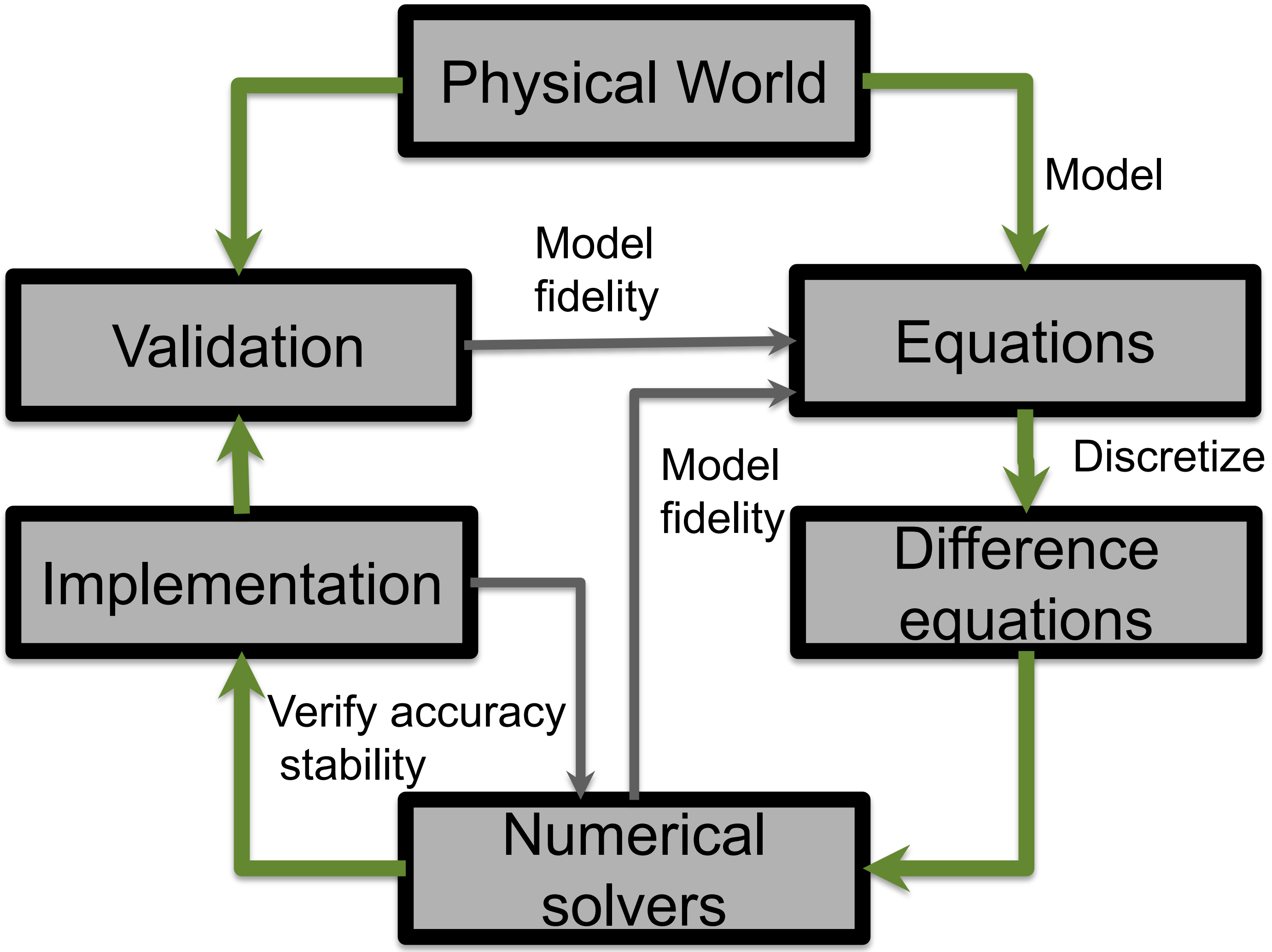
Supercomputers change fast
Especially now!

IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

Challenges Developing a Scientific Application

Technical	Sociological
• All parts of the cycle can be under research • Requirements change throughout the lifecycle as knowledge grows • Verification complicated by floating point representation • Real world is messy, so is the software	• Competing priorities and incentives • Limited resources • Perception of overhead without benefit • Need for interdisciplinary interactions

Lifecycle of a Scientific Application



- Modeling
  - Approximations
  - Discretizations
  - Numerics
    - Convergence
    - Stability
- Implementation
  - Verification
    - Expected behavior
  - Validation
    - Experiment/observation

Heroic Programming

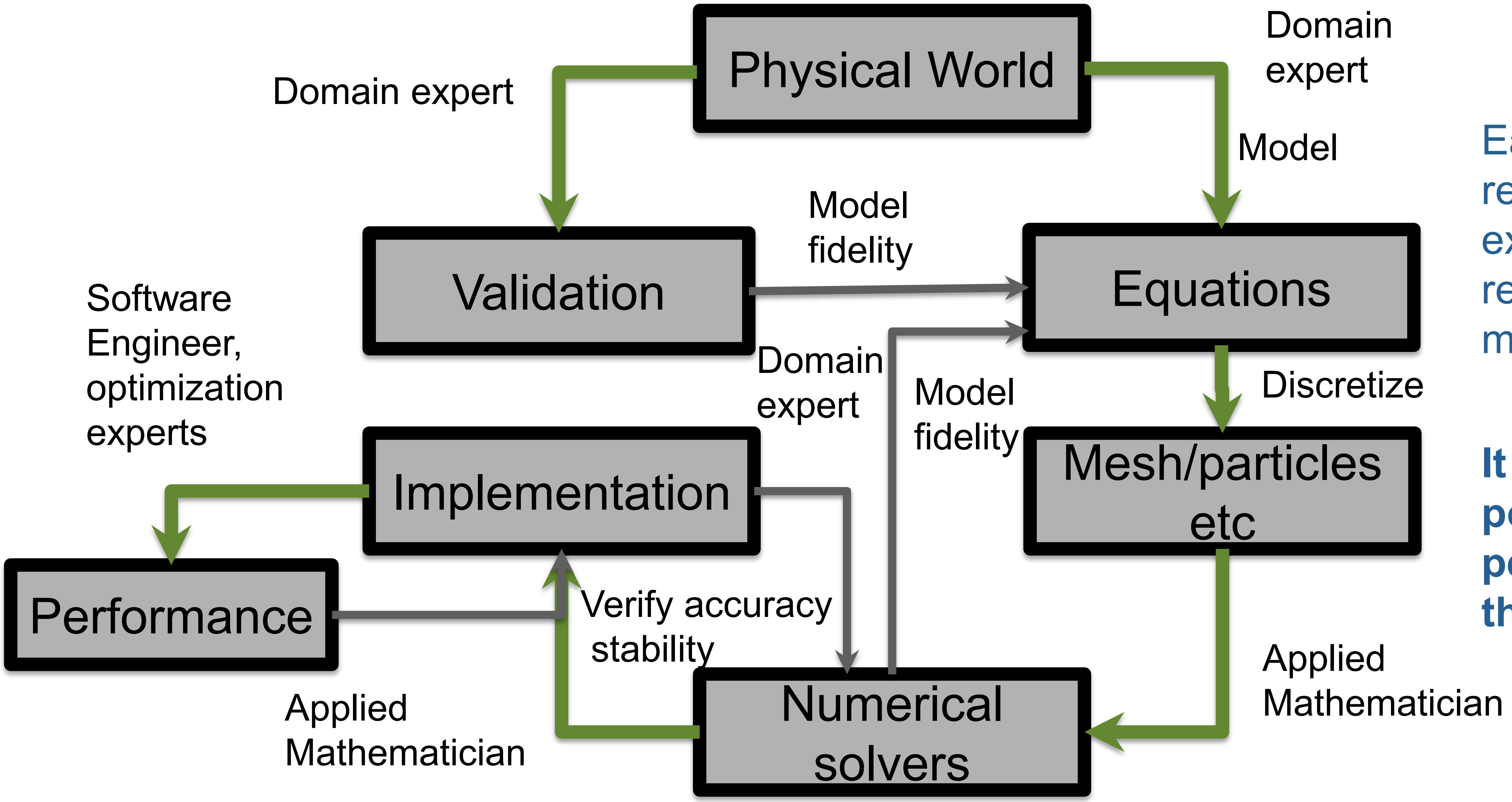
Usually a pejorative term, is used to describe the expenditure of huge amounts of (coding) effort by talented people to overcome shortcomings in process, project management, scheduling, architecture or any other shortfalls in the execution of a software development project in order to complete it. Heroic Programming is often the only course of action left when poor planning, insufficient funds, and impractical schedules leave a project stranded and unlikely to complete successfully.

From http://c2.com/cgi/wiki?HeroicProgramming

Science teams often employ heroic programming

Many do not see anything wrong with that approach

Expertise Map



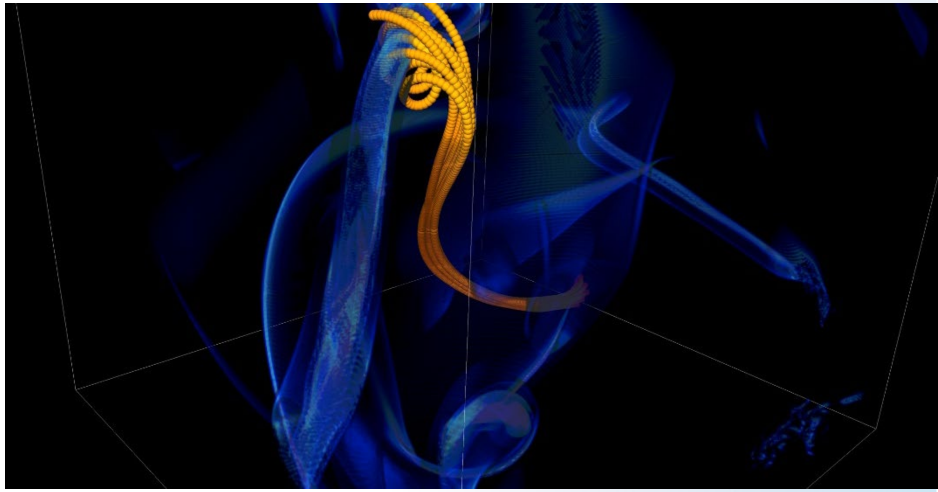
Each of these roles require deeper expertise as scientific requirements grow more complex.

It is no longer possible for a single person to take on all these roles

Good scientific process
requires
good software practices

Good software practices
increase
scientific productivity

You Can Mitigate Risk, But It Is Never Zero



- Short notice availability of one of the biggest machines of it’s time
  - **< 1month to get ready, run was 1.5 weeks**
- Quick and dirty development of particle capability in code
- Error in tracking particles resulted in duplicated tags from round-off
- Had to develop post-processing tools to correctly identify trajectories
  - **6 months to process results**

FLASH had a software process in place. It was tested regularly. This was one instance when the full process could not be applied because of time constraints.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Why Be Concerned with Software Engineering?

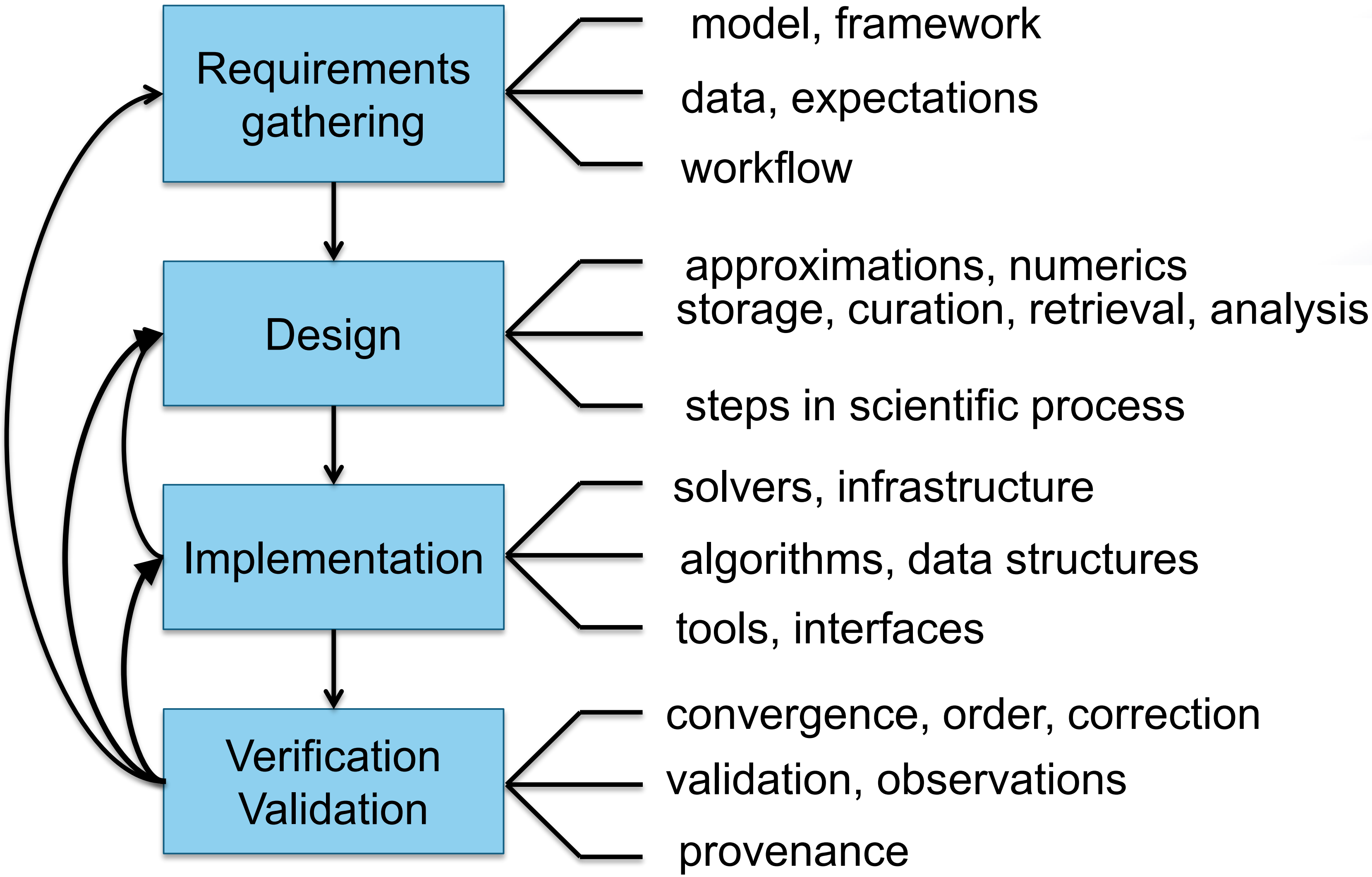
Accretion leads to unmanageable software

- Increases cost of maintenance
- Parts of software may become unusable over time
- Inadequately verified software produces questionable results
- Increases ramp-on time for new developers
- Reduces software and science productivity due to **technical debt**

Consequences of Choices

"Quick and dirty" collects technical debt, which means more effort required to add features.

Lifecycle: Software Engineering View



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Taking Stock of *Your* Situation

- Software architecture and process design is an overhead
  - Value lies in avoiding technical debt (future saving)
  - Worthwhile to understand the trade-off

- The goals of the software
  - Proof-of-concept
  - Verification
  - Exploration of some phenomenon
  - Experiment design
  - Analysis
  - Other …

Cognizant of resource constraints

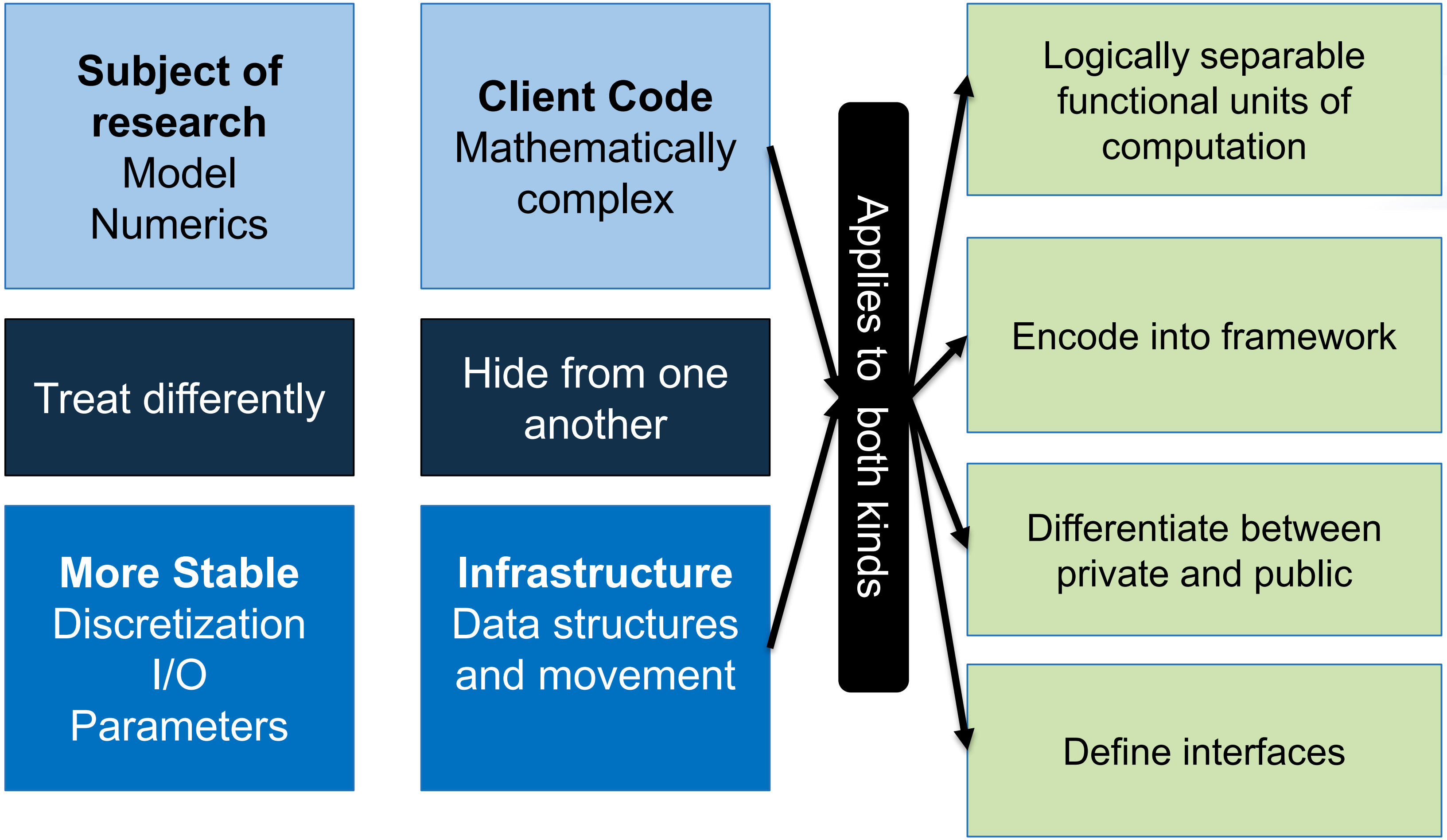
Dictate the rigor of the design and software process

Reconcile Conflicting Requirements

- Separation of concerns
  - Encapsulation of functionalities where possible
  - Abstractions for encapsulations
    - Offload complexity where possible
- Hard-nosed trade-offs
  - Flexibility and composability vs raw performance
  - Extensibility and developer productivity

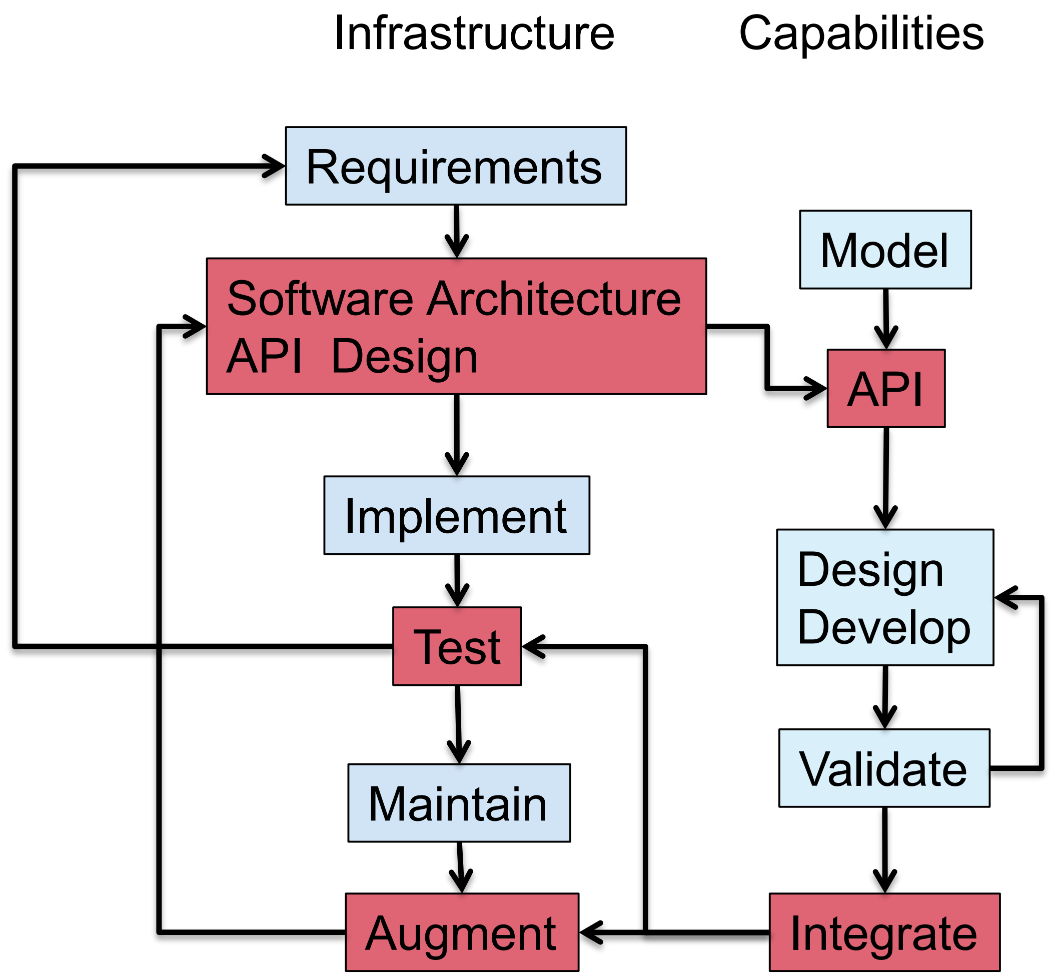
Architecting Scientific Codes

Taming the Complexity: Separation of Concerns



IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

A Successful Model

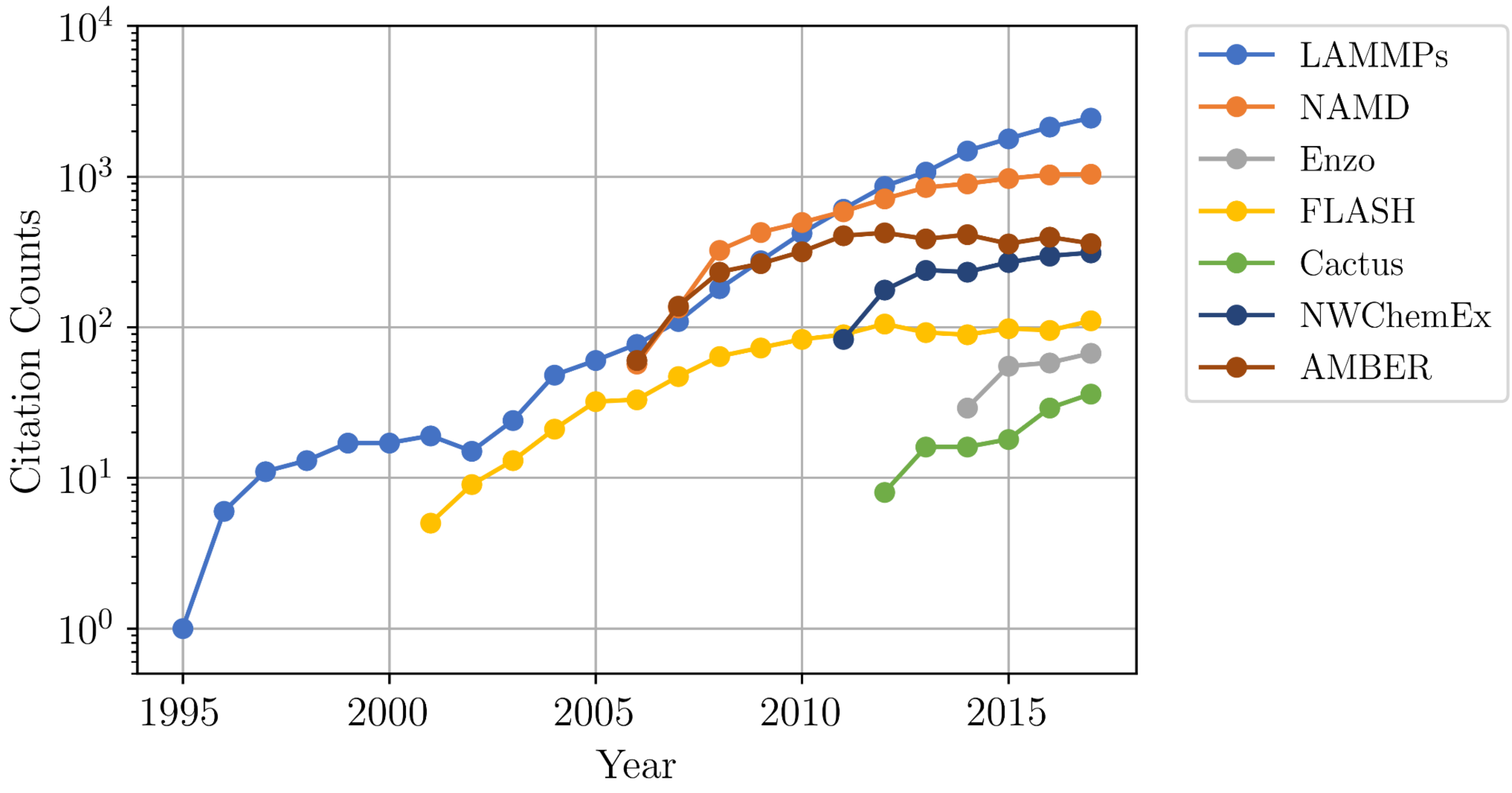


Design Investment Impact – FLASH Example

capabilities	categories	new community reached	year
base	all	thermonuclear astrophysics	2000
MHD	physics	reconnection, solar plasma	2002
particles	physics and infrastructure	cosmology	2003
multigrid	infrastructure	CFD	2008
Lagrangian Markers	infrastructure	FSI	2009
PIC	pl		
nuclear EOS, neutrino source terms and leakage	pl		
3-T, conductivity Radiation, laser	pl in		
sink particles	pl		

	astro-physics	cosmo-logy	CFD/ FSI	HEDP	solar physics	recon-nection	star fo-rmation	combus-tion
compress-ible hydro	1998	*		*	*			*
burn	1999							*
MHD	2002	*		*	*	*	*	
elliptic solver	*	2001 2001	*				*	
particles	*	2002	*	*		*	*	*
bittree	*	*	2012	*				
HYPRE interface			*	2011				
radiation	*	*		2011				

Community Impact of Well Done Software



Software Process Best Practices

Baseline

- Invest in extensible code design
- Use version control and automated testing
- Institute a rigorous verification and validation regime
- Define coding and testing standards
- Clear and well defined policies for
  - Auditing and maintenance
  - Distribution and contribution
  - Documentation

Desirable

- Provenance and reproducibility
- Lifecycle management
- Open development and frequent releases

A Useful Resource

https://ideas-productivity.org/resources/howtos/

- **‘What Is’ docs**: 2-page characterizations of important topics for SW projects in computational science & engineering (CSE)
- **‘How To’ docs**: brief sketch of best practices
  - Emphasis on ``bite-sized" topics enables CSE software teams to consider improvements at a small but impactful scale
- We welcome feedback from the community to help make these documents more useful

Other Resources

http://www.software.ac.uk/

http://software-carpentry.org/

http://flash.uchicago.edu/cc2012/

http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=4375255

http://www.orau.gov/swproductivity2014/SoftwareProductivityWorkshopReport2014.pdf

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6171147

Summary

- Good software practices are needed for scientific productivity
- Science at extreme-scales is complex and requires multiple expertise
- Software process does need to address reality
- Open codes, community contribution, are a powerful tool

Science through computing is
at best
as credible as the software that produces it

IDEAS productivity ECP EXASCALE COMPUTING PROJECT