Mission Driven Innovation:
A Case Study in Mutliphysics Code Development

Robert N. Rieben

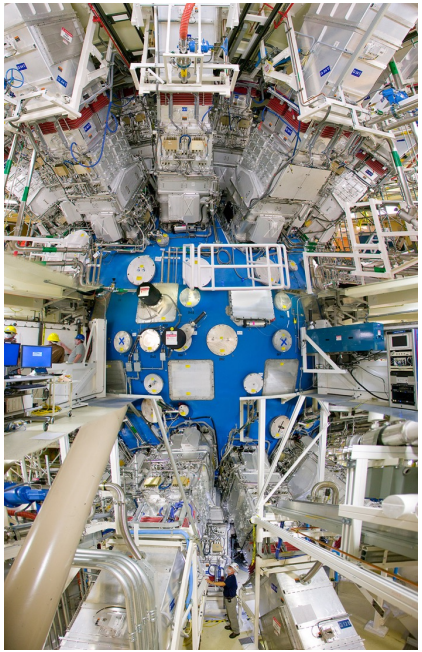
LLNL-PRES-XXXXXX

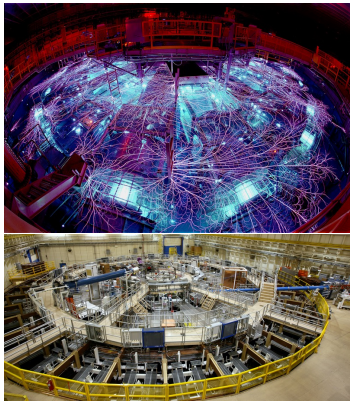
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

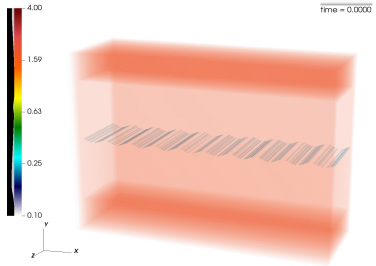
High energy density physics experiments require complex multi-physics simulations

- **Radiation-magneto-hydrodynamics:**
  - Multi-material, all speed “hydrodynamics”
    - Solid, liquid, gas, plasma phases
    - Compressible flow; complex material properties
    - Explicit time integration
  - Radiation/magnetic diffusion/transport
    - Fluid coupling via force/energy exchange
    - 3T plasma physics and TN burn
    - Implicit/explicit (IMEX) time integration
- **Multiple diverse algorithms:**
  - Arbitrary Lagrangian-Eulerian (ALE), direct Eulerian, Particle and Monte Carlo methods
  - Unstructured AMR, high-order discretizations
- **Applications:**
  - Inertial confinement fusion (ICF)
  - Pulsed power experiments
  - Equation of state/material strength experiments





National Ignition Facility (*left*), Z-machine at Sandia (*top*), ablator-gas atomic mix experiments at NIF, (*top-right*), and MARBL simulation of turbulent KH shear layer experiment at OMEGA laser facility (*right*).

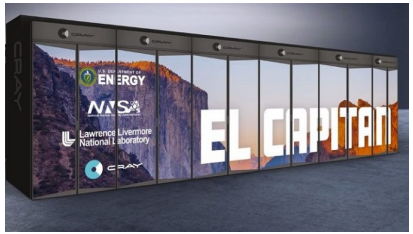


Our goal is to model and ultimately predict the behavior of complex physical systems

High performance computing is essential for predictive science

- Our multiphysics simulation codes must:
  - Perform at scale on massively parallel computing architectures
    - This now includes node level parallelism (GPUs) and exa-scale!
  - Be adaptable and extendable
  - Be sustainable across multiple generations of hardware
    - Code lifetimes measured in ***decades***
  - Work on general (unstructured) 2D and 3D meshes

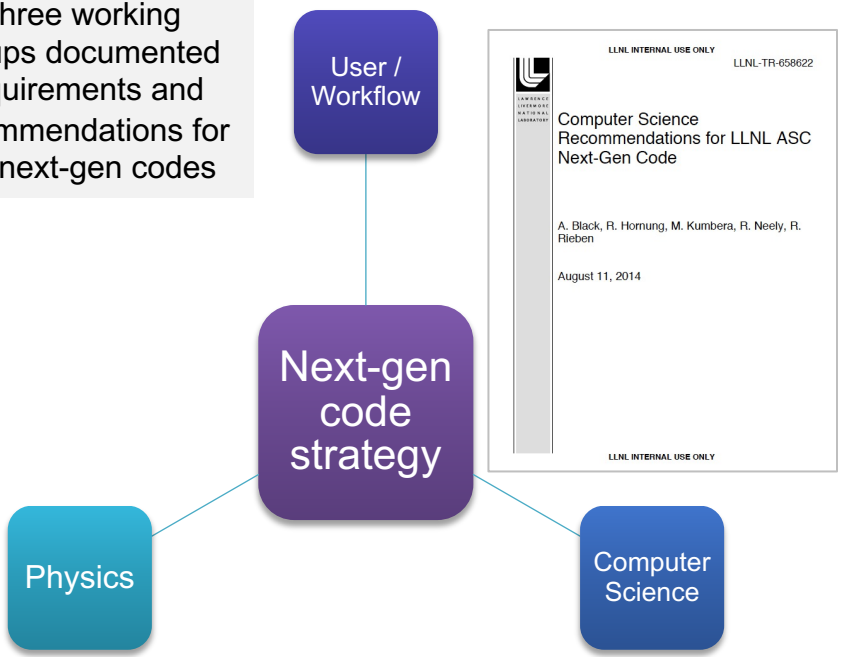
- In addition, we need to anticipate future technologies like
  - AI/ML in the simulation loop
  - Optimization driven workflows



The NNSA labs have a long history of developing successful codes to utilize the ever-increasing power in HPC

The Multiphysics on Advanced Platforms Project (MAPP) was started in 2015 to meet the exascale computing challenge

Three working groups documented requirements and recommendations for our next-gen codes



LLNL INTERNAL USE ONLY

LLNL-TR-658622

Computer Science Recommendations for LLNL ASC Next-Gen Code

A. Black, R. Hornung, M. Kumbera, R. Neely, R. Rieben

August 11, 2014

LLNL INTERNAL USE ONLY

Key recommendations:

- Develop modular CS infrastructure up front
- Modular physics components
  - ALE hydro as main option
  - Include direct Eulerian option
- Mesh abstraction layer to enable multiple views into data
- Checkpoint/Restart should depend only on inline database
- C++ as primary language, allow C and Fortran20xx kernels
- Support external productivity tools for mesh generation
- Mesh setup and management improvements for user workflow

The lessons learned from the ASCI program* were closely studied and followed

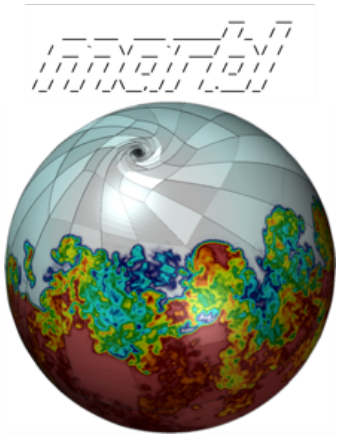
- Build on successful code development history and prototypes
- Risk identification, management and mitigation
- Determine the schedule and resources from the requirements
- Customer focus
- Use of modern but proven computer science techniques
- Better physics and computational mathematics is more important than better computer science

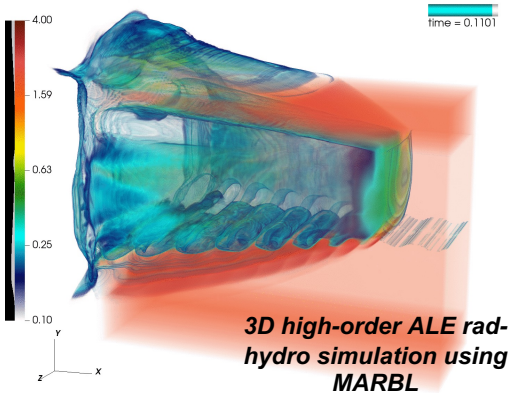
*D. Post, R. Kendall, “Lessons Learned from ASCI,”, 2004

We believe these lessons hold true in the ATDM / exascale era

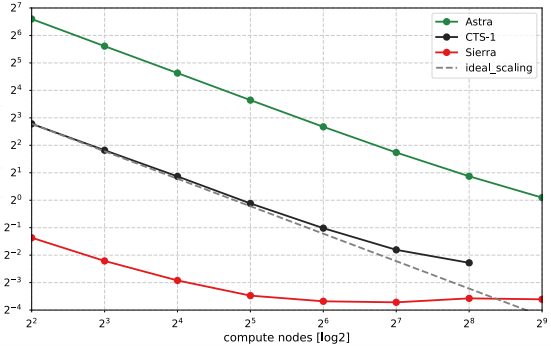
MARBL is a NextGen multiphysics code based on high-order numerical algorithms and modular CS infrastructure

- **Next-gen ICF, pulsed power code**
  - Part of MAPP: Multiphysics on Advanced Platforms Project
  - Developed under ATDM
- **Modular infrastructure and physics packages**
  - High-order finite element ALE hydrodynamics
    - High-order finite difference Eulerian option
  - 3T multi-group diffusion with TN burn
  - In development:
    - Deterministic and implicit Monte Carlo radiation transport
    - Turbulence models
- **High-order numerical methods**
  - Higher resolution/accuracy per unknown
  - Higher FLOP/byte, improved GPU throughput
- **GPU performance**
  - Code scaled to ½ of Sierra and all of Astra (SNL)
  - Up to 15X GPU vs CPU node speedup





3D high-order ALE rad-hydro simulation using MARBL



MARBL strong scaling on Sierra, Astra and CTS-1

This project has many roots, going back to LDRD funded research at the TRL level 1

TRL 8 and 9: 2021 and beyond

- In the post ATDM era, we need a sustained effort as we transition into full production and TRL 9 status

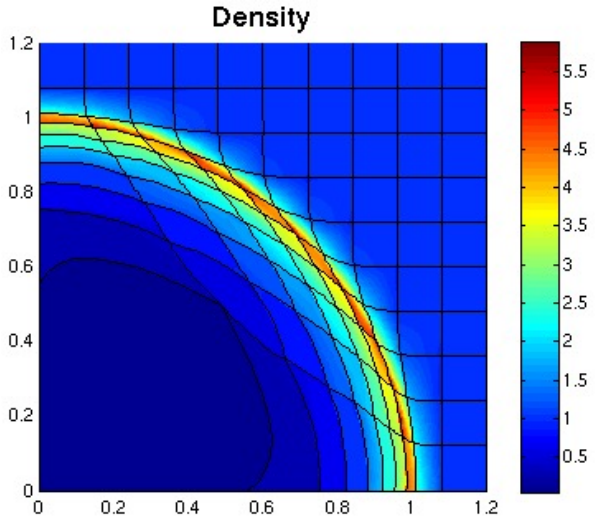
TRL 4 to 7: 2015 to 2020 (ATDM Program)

- Build on LDRD successes
- Focused development, ST integration and HPC performance/scaling

TRL 1 to 3: prior to 2015

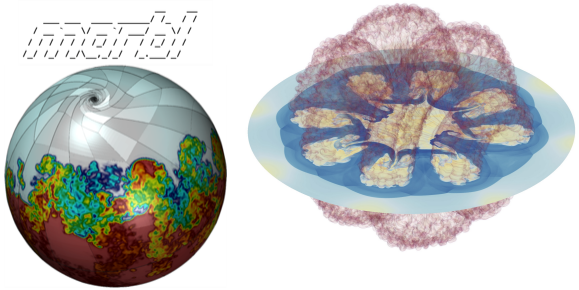
- Novel approaches investigated via LDRD/Strategic Initiatives
- Promote successful ideas, learn and move-on from failures

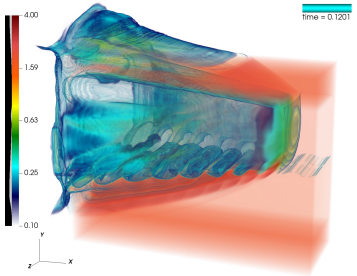
There is no formal process for standing up a new code of this scale, but there are plenty of lessons learned

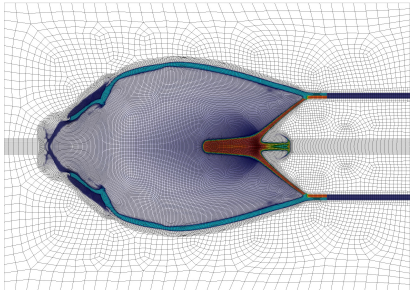


TRL1 to TRL7
in ~12 years

How did we
do it?







July 16th, 2009
First ever high-order Lagrangian computation of Sedov problem (***MatLab code***!)
by summer student (T. Ellis) at LLNL

Today
MARBL multiphysics code: high-order, 1D/2D/3D unstructured mesh, GPU enabled, massively parallel, AMR, 40+ TPLs, >1M SLOC, ~40 contributors

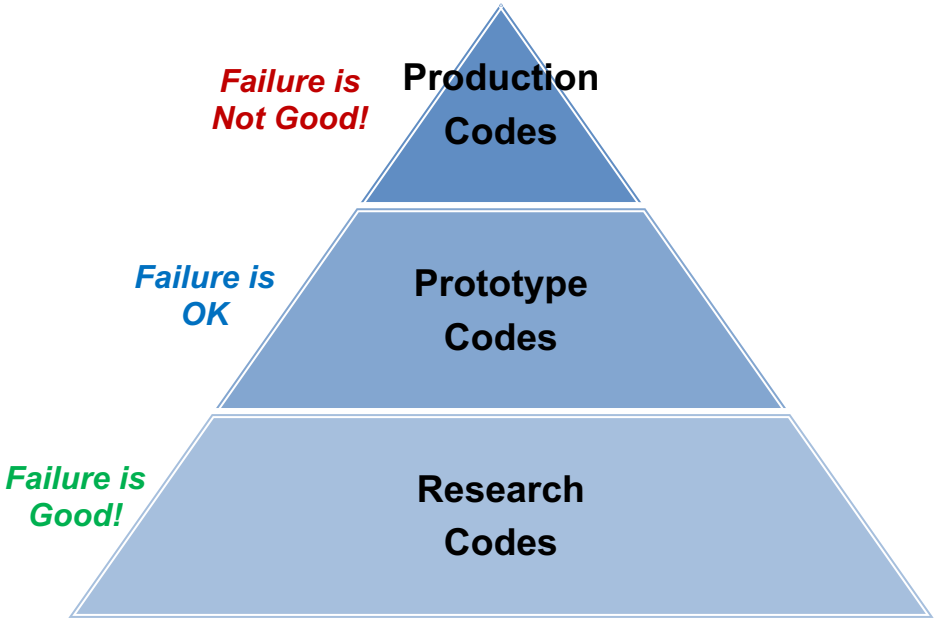
Managing complexity is key to bridging the research to development gap

Gall's Law:

"A complex system that works is invariably found to have evolved from a simple system that worked. The inverse proposition also appears to be true: A complex system designed from scratch never works and cannot be made to work. You have to start over, beginning with a working simple system."

– J. Gall, from *Systemantics: How Systems Really Work and How They Fail*

Gall's law aligns closely with our intuition about the need for failure in research to promote innovation

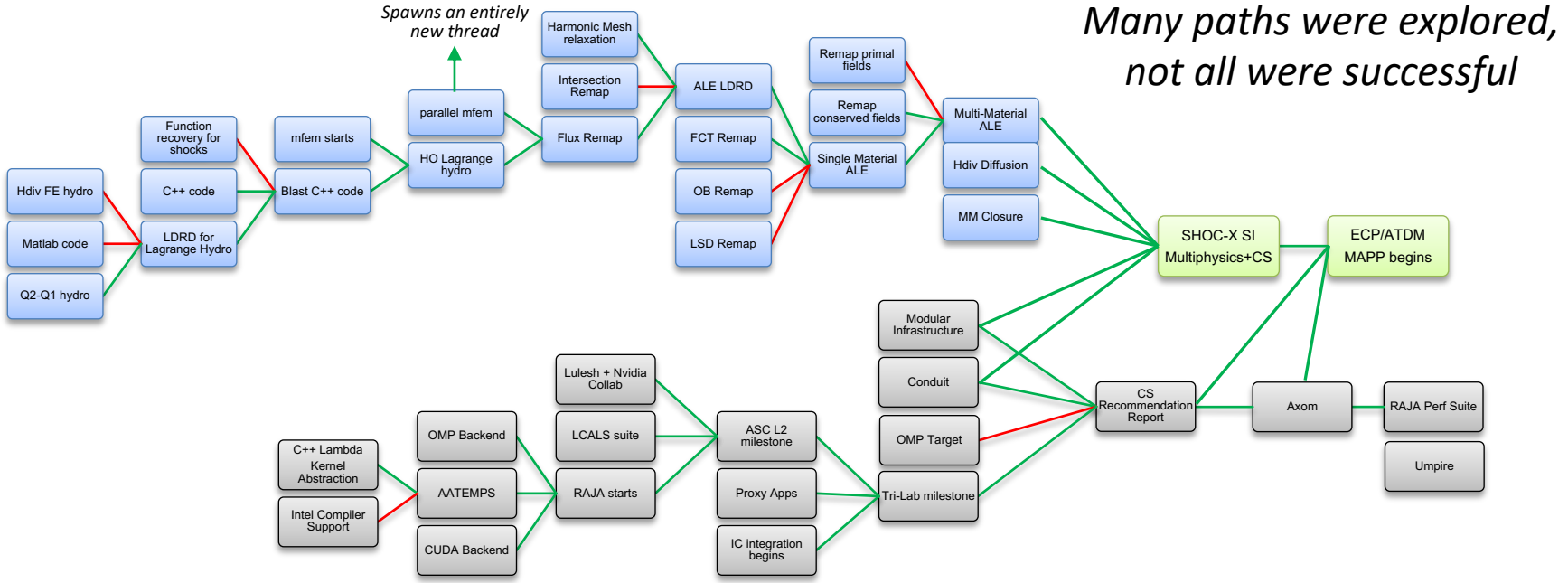


The "Tipton Model" for Physics Code Development

- Robert Tipton proposed a model* to foster innovation in code development while preventing expensive failures
- Innovation is encouraged at the **research** level
  - R+D efforts, 1-3 person teams
  - Expect many failures
    - Failures are fine! We learn a lot from them
  - Success means ability to solve a new problem
    - Or an existing problem in a better way
- Successful research codes are promoted to **prototype**
  - Teams of 3-6
  - User uptake on a focused application is metric for success
- Successful prototypes are promoted to **production** level
  - Full sized teams of 10-12
  - Mission impact is key focus
  - Users get final say on success/failure

R Tipton, LLNL, unpublished notes

Ultimately, complex codes are evolved over time, integrating features from multiple R+D efforts



This is just a simplified subset, there are many parallel strands that merged to/from this timeline

Infrastructure and a supporting software ecosystem is key to transforming research into production

- We need to collaboratively develop and share modular software building blocks across applications

- Why is this important?
  - Applications have many support needs in common (in-memory data management, file I/O, spatial queries and indexing schemes, visualization and analysis, etc.)
  - Replicating code-specific capabilities has high development costs for application teams and are are hard to vet, maintain, and share
  - HPC platform diversity requires more CS expertise that must be shared

- How are we doing this?
  - We are building general open source capabilities that broaden our base of contributors beyond traditional WSC code teams
  - Productized toolkits of components provide foundational building blocks that all codes (research, prototype and production) can leverage and customize their use

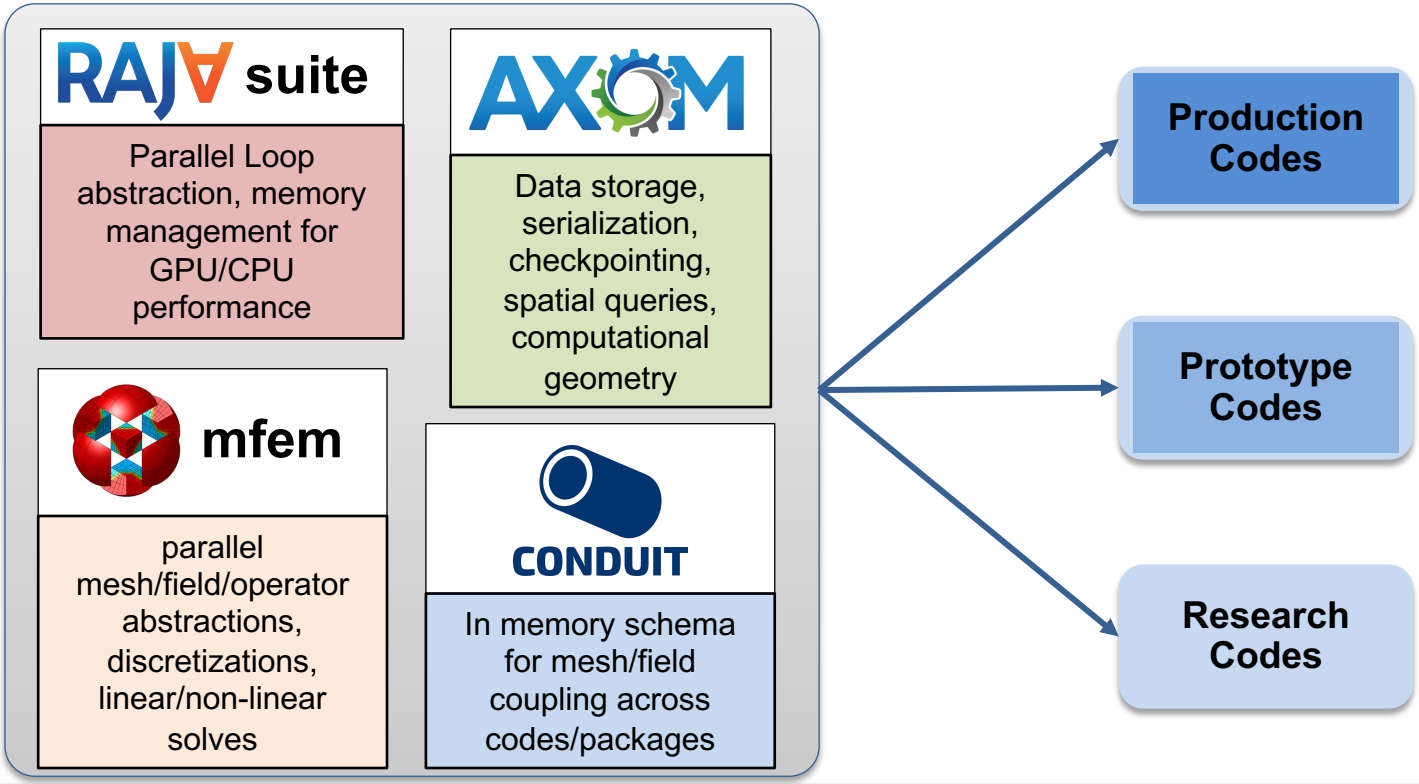
Courtesy R. Hornung (LLNL)

Toolkits of software building blocks enable applications to leverage a wide range of CS domain expertise

Shared Toolkits

The standard for multiphysics simulation capabilities on HPC platforms continues to rise.

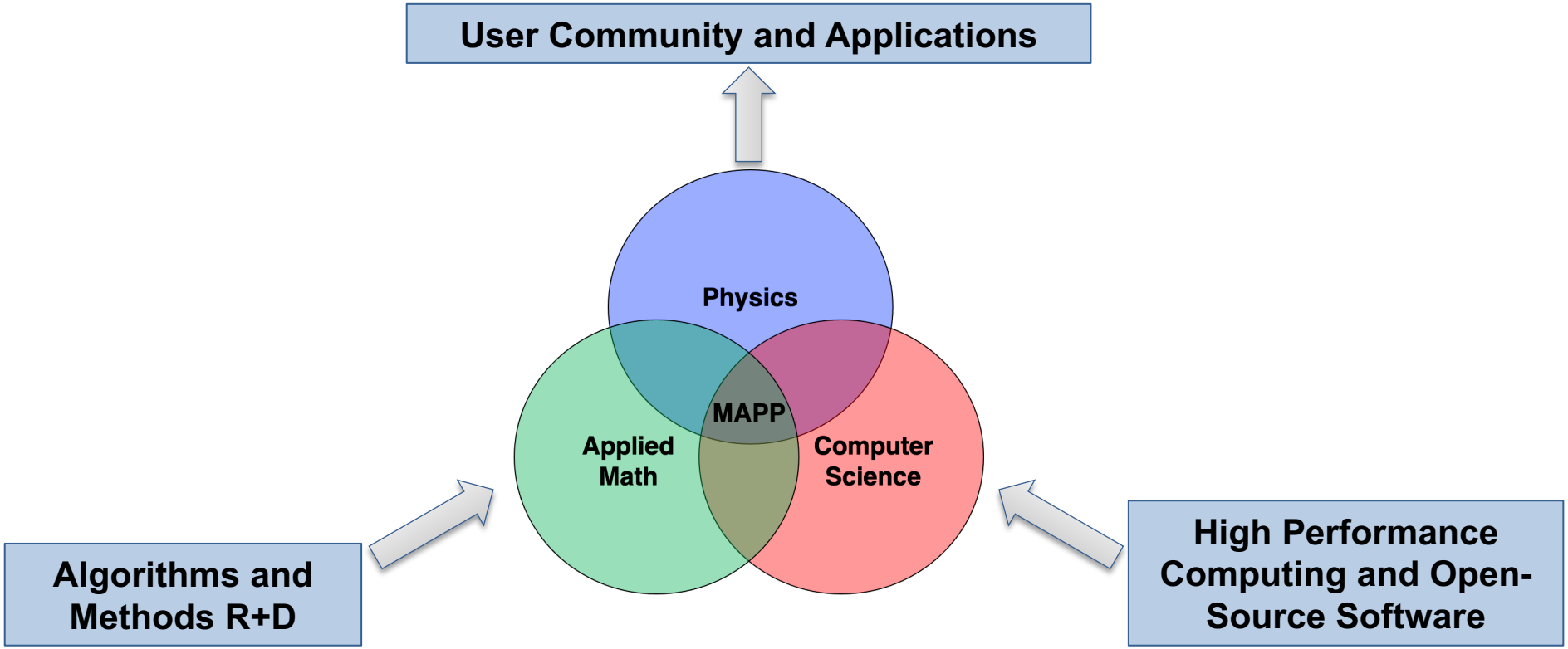
We need shared infrastructure to enable future innovations to feed into the multiphysics codes of today and tomorrow



Co-design and inter-disciplinary collaboration is critical for success: *Modularity requires communication*

- Updates are not a simple process which can be done on short notice
  - often needed in a production environment
- Careful communication, coordination and testing across the multiple projects involved is needed
- Maintaining constant and multiple paths of communication and coordination is essential to realize the many benefits of modularity
- Individual libraries must view their success as dependent on the success of the integrated applications

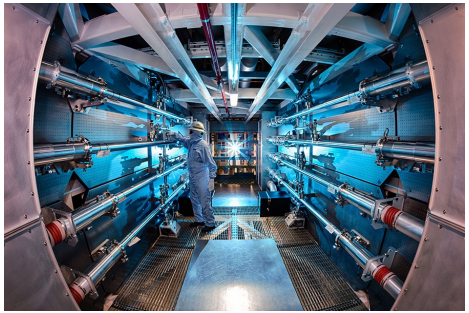
Multiphysics codes are a careful balance between the disciplines of physics, applied mathematics and computer science



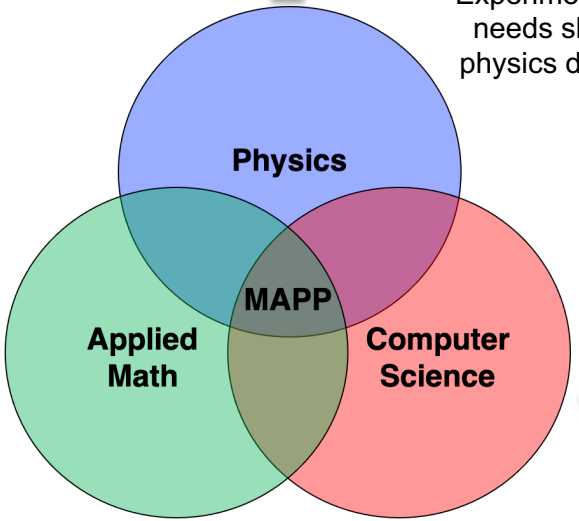
There are important feedback mechanisms that must be in place to ensure success

User Community and Applications

Physics Experiments



Experiment modelling needs should drive physics development



New research in algorithms and methods should be driven by needs from the code

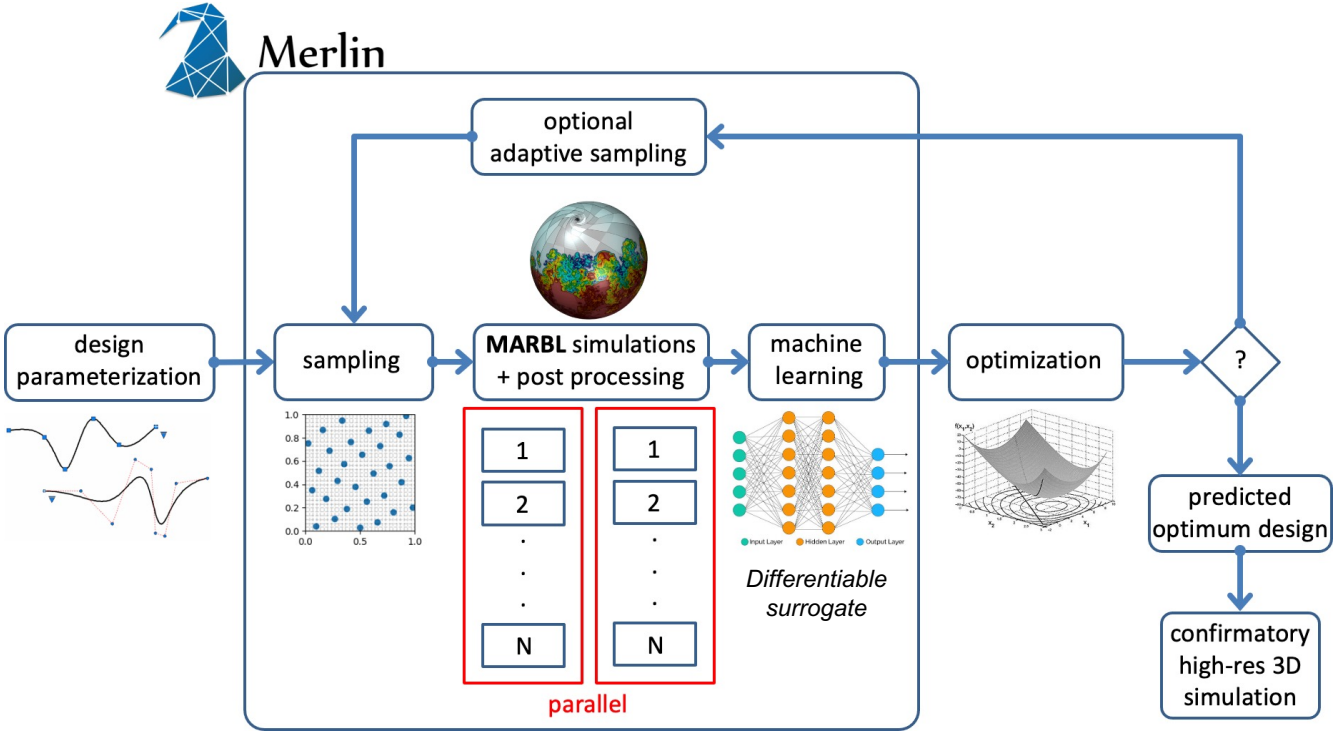
Code teams should provide input into hardware procurements

Algorithms and Methods R+D

High Performance Computing and Open-Source Software

Our investments in modular infrastructure and data services will feed into future code development

- Design optimization for non-linear multiphysics is an exciting area of interest to LLNL
- Today's design process is experience/intuition driven
- This process prohibits rapid response to new design challenges
- Today's HPC can perform 100-1000's simulations per day, tomorrow's HPC will exceed that. We need a new design paradigm to take full advantage of HPC



Concluding Remarks

- The lessons learned from ASCI hold true in the ATDM / exascale era
- Managing complexity is key to bridging the research to development gap
- Gall's law applies to Multiphysics code development
- Research codes should be encouraged to promote innovation
- The Tipton model of code development can be used to promote innovation while minimizing expensive failures
- Codes need to be plugged into user application space early and often
- Infrastructure and a supporting software ecosystem is key to transforming research into production