Scientific Software Design

Anshu Dubey
Argonne National Laboratory

Better Scientific Software Tutorial, ISS, March 2021

License, Citation and Acknowledgements

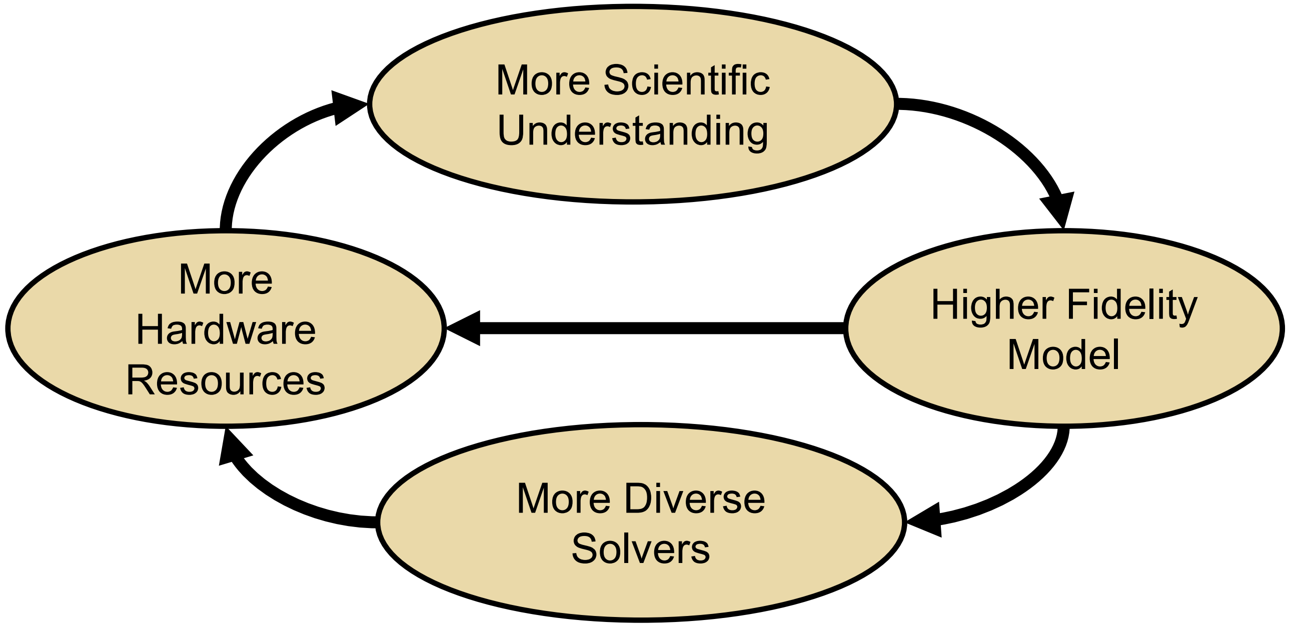
License and Citation

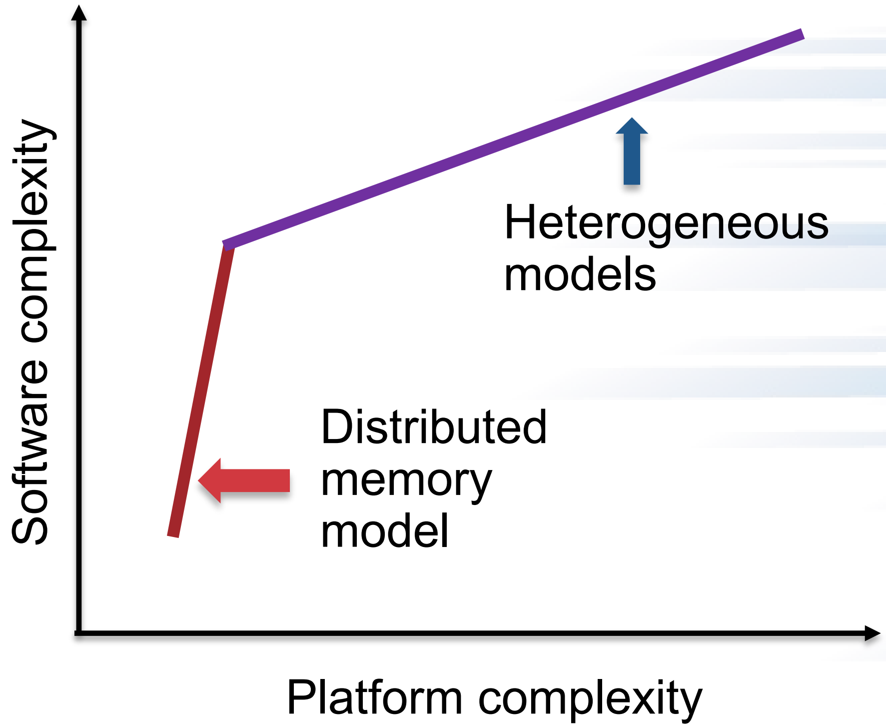
- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Rinku K. Gupta, and David M. Rogers, Better Scientific Software tutorial, in Improving Scientific Software conference, online, 2021. DOI: 10.6084/m9.figshare.14256257**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software tutorial…

Acknowledgements

- Additional contributors include: Mike Heroux, Alicia Klinvex, Mark Miller, Jared O'Neal, Katherine Riley, David Rogers, Deborah Stevens, James Willenbring
- 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 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.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at the Los Alamos National Laboratory, which is managed by Triad National Security, LLC for the U.S. Department of Energy under Contract No.89233218CNA000001
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

HPC Computational Science Use-case





General Design Principles for HPC Scientific Software

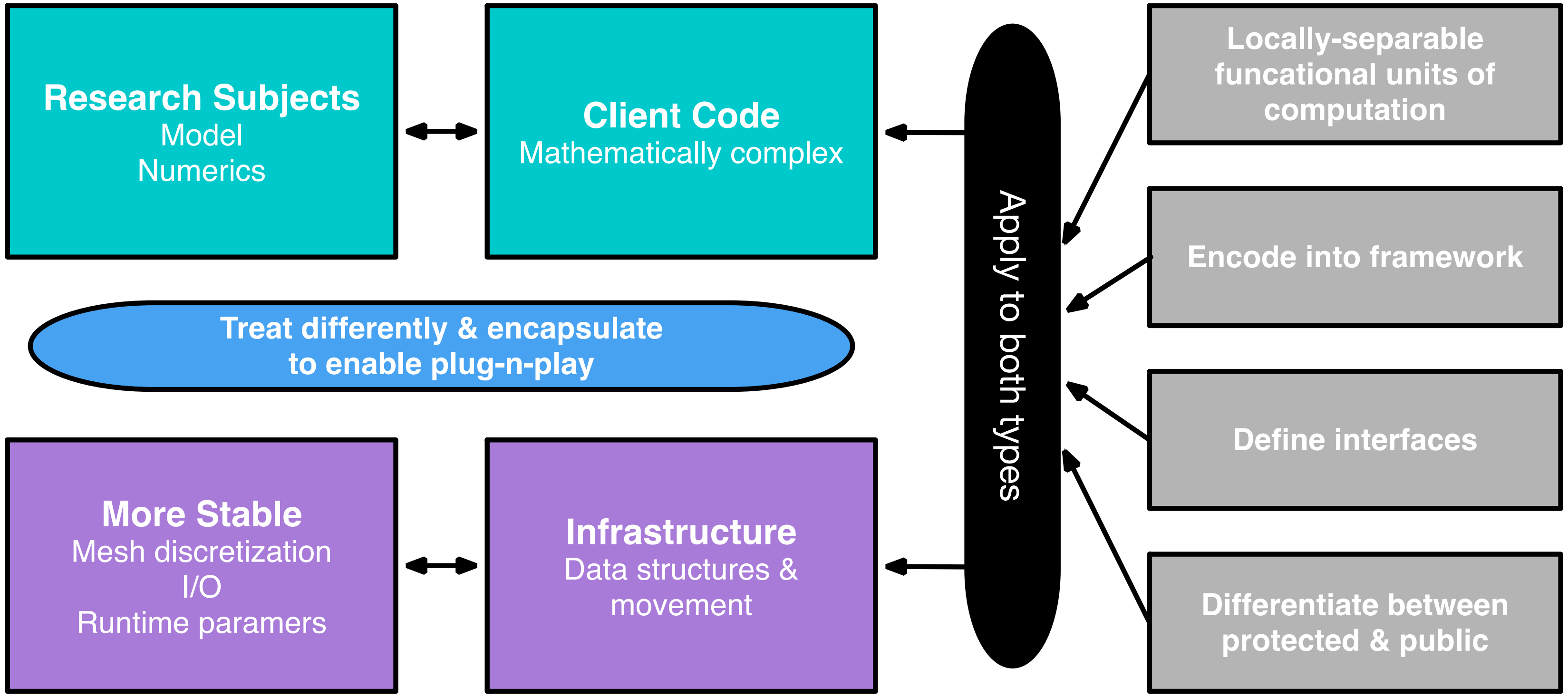
Considerations

- ❑ Multidisciplinary teams
  - ❑ Many facets of knowledge
  - ❑ To know everything is not feasible
- ❑ Two types of code components
  - ❑ Infrastructure (mesh/IO/runtime …)
  - ❑ Science models (numerical methods)
- ❑ Codes grow
  - ❑ New ideas => new features
  - ❑ Code reuse by others

Design Implications

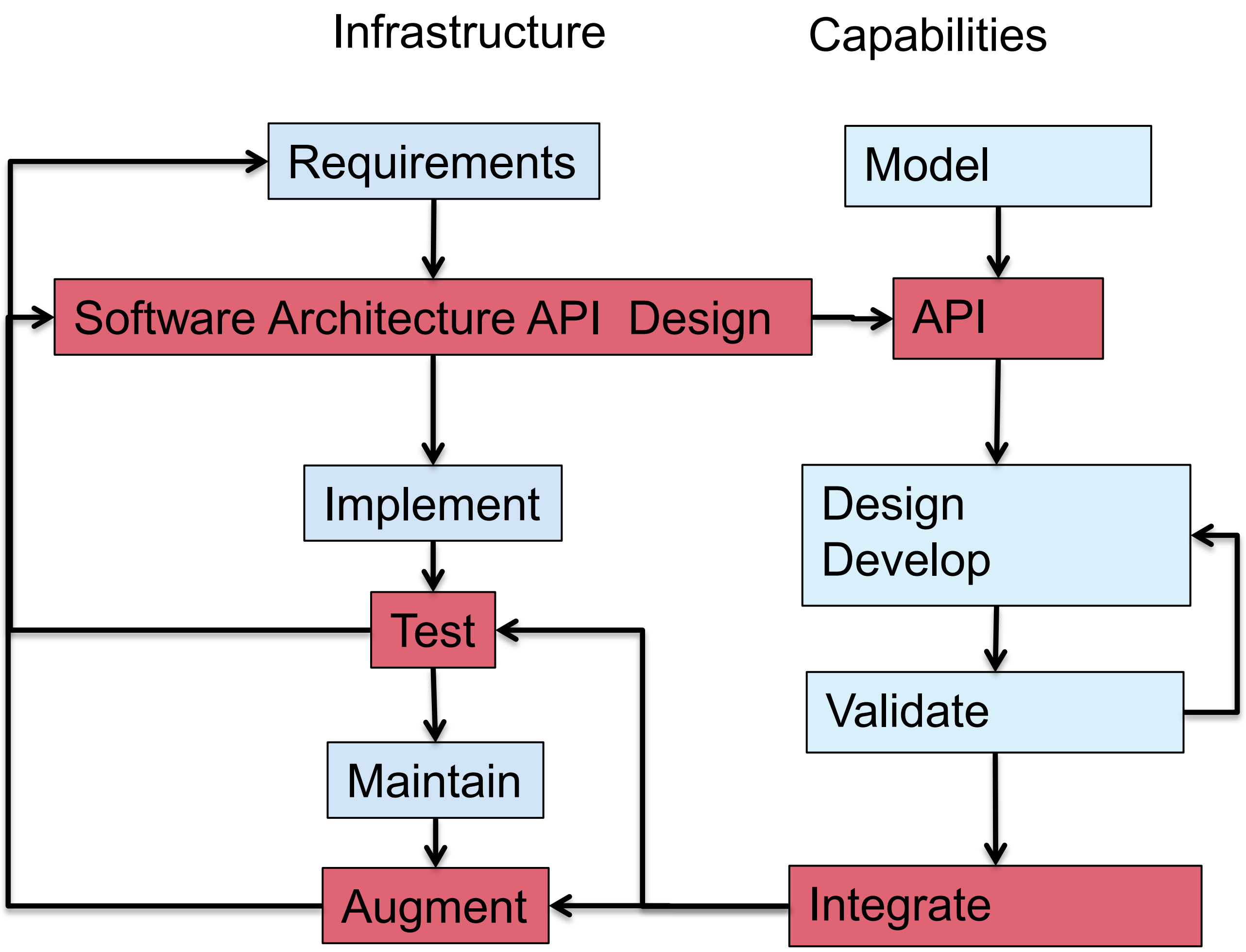
- ❑ Separation of Concerns
  - ❑ Shield developers from unnecessary complexities
- ❑ Work with different lifecycles
  - ❑ Long-lasting vs quick changing
  - ❑ Logically vs mathematically complex
- ❑ Extensibility built in
  - ❑ Ease of adding new capabilities
  - ❑ Customizing existing capabilities

General Design Principles for HPC Scientific Software



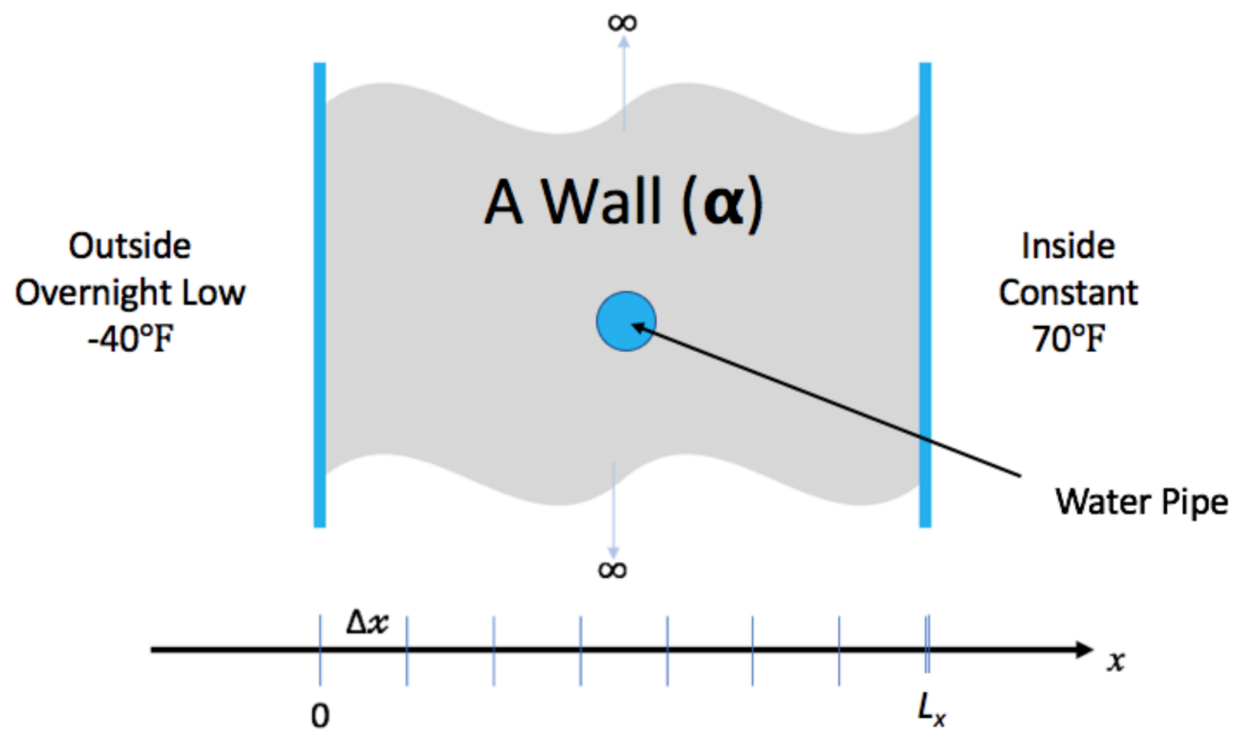
Design first, then apply programming model to the design instead of taking a programming model and fitting your design to it.

A Design Model for Separation of Concerns



The Running Example

Lets say you live in a house with exterior walls made of a single material of thickness, $$L_x$$. Inside the walls are some water pipes as pictured below.



You keep the inside temperature of the house always at 70 degrees F. But, there is an overnight storm coming. The outside temperature is expected to drop to -40 degrees F for 15.5 hours. Will your pipes freeze before the storm is over?

Problem Specification - Design Considerations

- Specification
  - Solve heat equation with some initial and boundary conditions
  - Apply different integration methods

- What is infrastructure here?
  - Discretization/ State
  - Verification
  - I/O
  - Application of initial conditions
  - Runtime parameters
  - Comparison

- What is model here?
  - Initial conditions
  - Boundary conditions
  - Integration

Infrastructure API

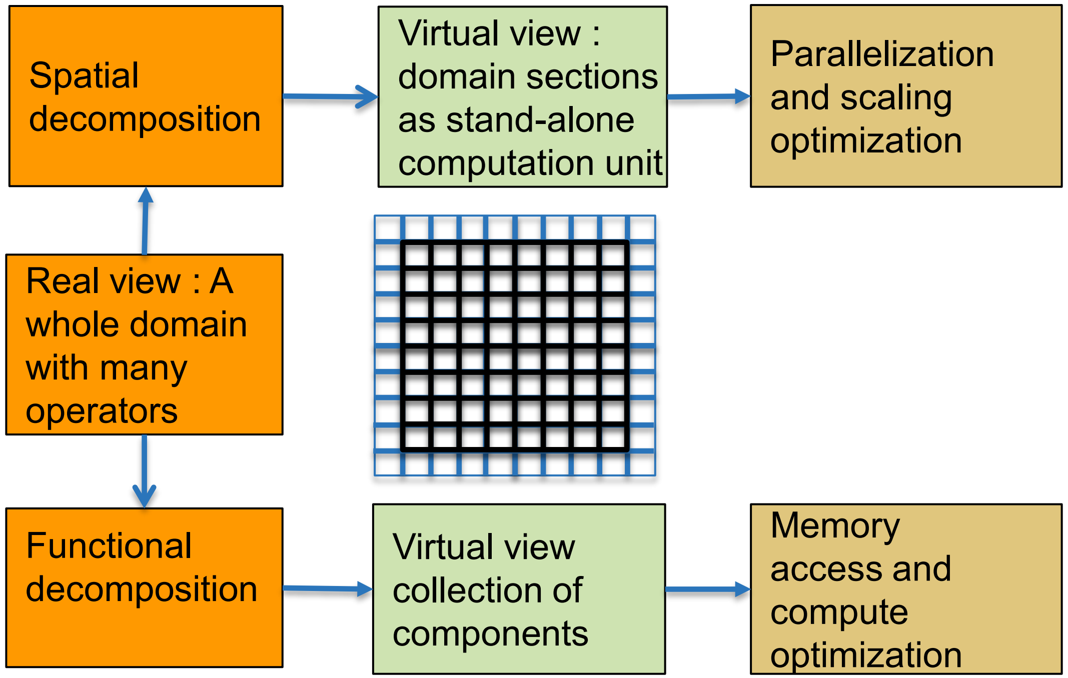
- process_args(int argc, char **argv)
- static void initialize(void)
- void copy(int n, double *dst, double const *src)
- void write_array(int t, int n, double dx, double const *a)
- void set_initial_condition(int n, double *a, double dx, char const *ic)

Numerics API

- double l2_norm(int n, double const *a, double const *b)
- static void r83_np_fa(int n, double *a)
- static void r83_np_sl (int n, double const *a_lu, double const *b, double *x)
- bool update_solution_crankn(int n, double *curr, double const *last, double const *cn_Amat, double bc_0, double bc_1)
- bool update_solution_upwind15(int n, double *curr, double const *last, double alpha, double dx, double dt, double bc_0, double bc_1)
- void compute_exact_solution(int n, double *a, double dx, char const *ic, double alpha, double t, double bc0, double bc1)
- bool update_solution_ftcs(int n, double *uk1, double const *uk0, double alpha, double dx, double dt, double bc0, double bc1)

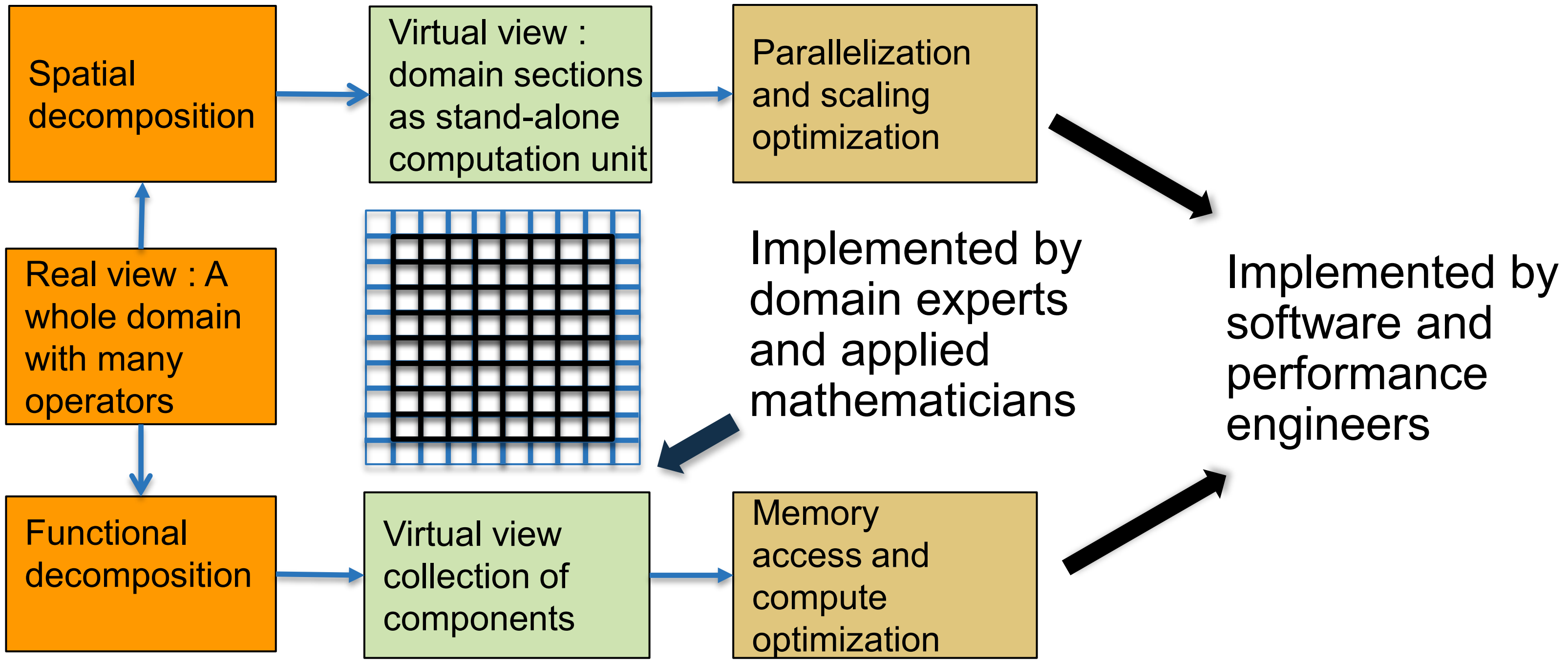
Example: Architecting Multiphysics PDEs

- Virtual view of functionalities
- Decomposition into units and definition of interfaces



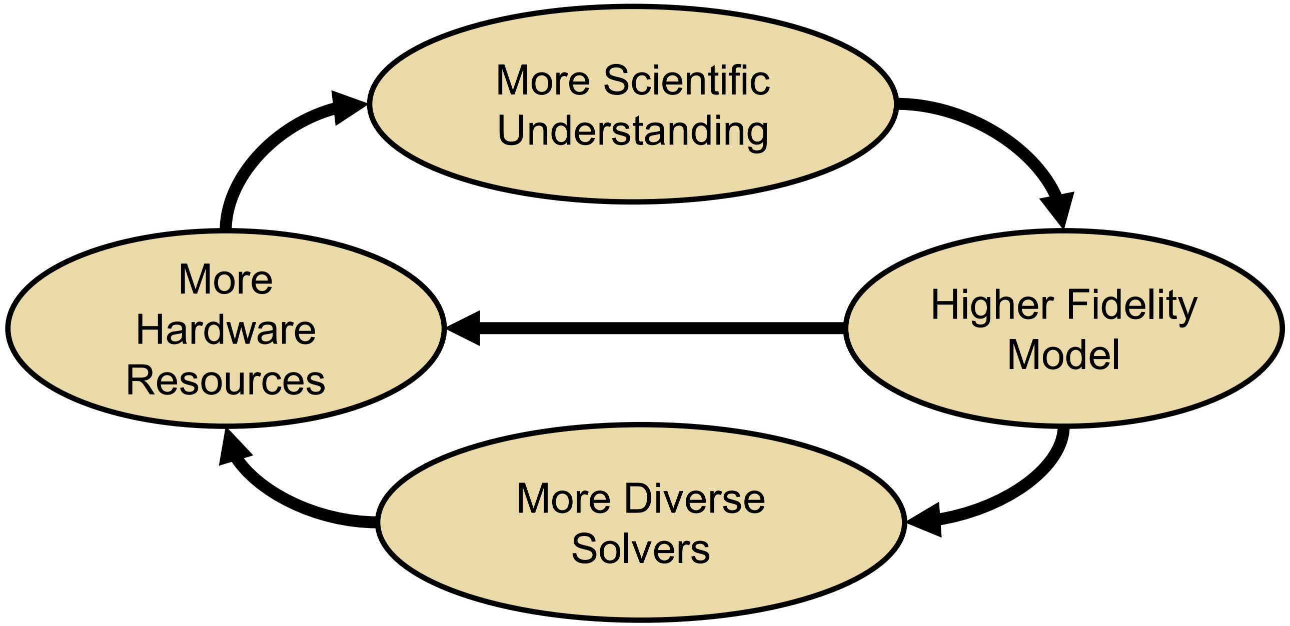
Example: Multiphysics PDEs for Distributed Memory Parallelism

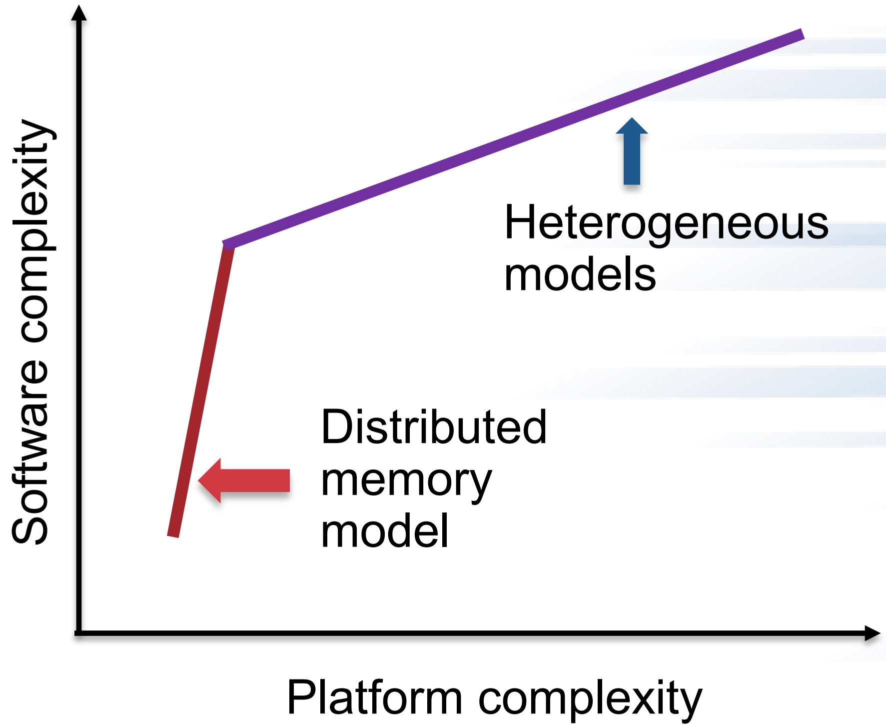
- Virtual view of functionalities
- Decomposition into units and definition of interfaces



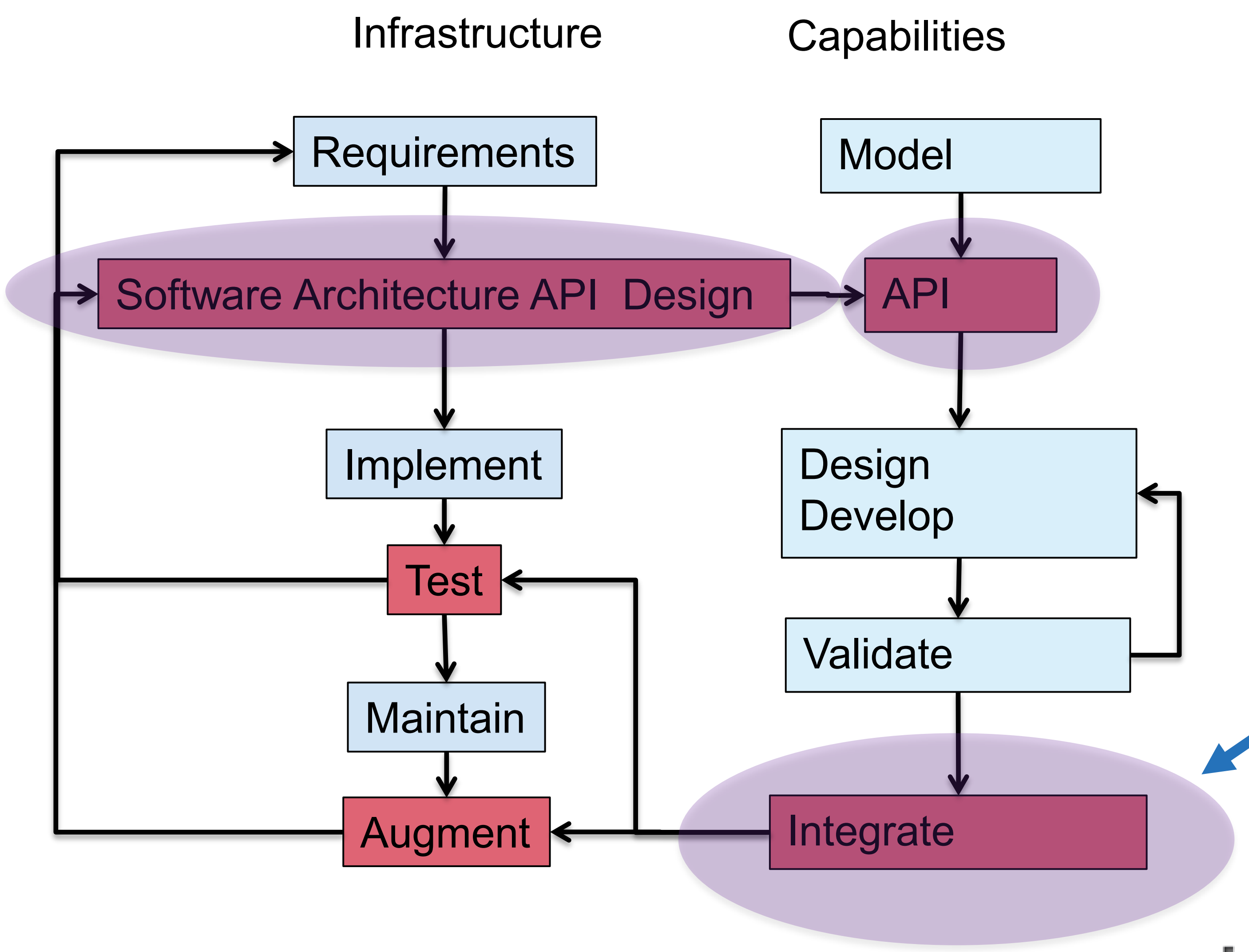
IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

HPC Computational Science Use-case



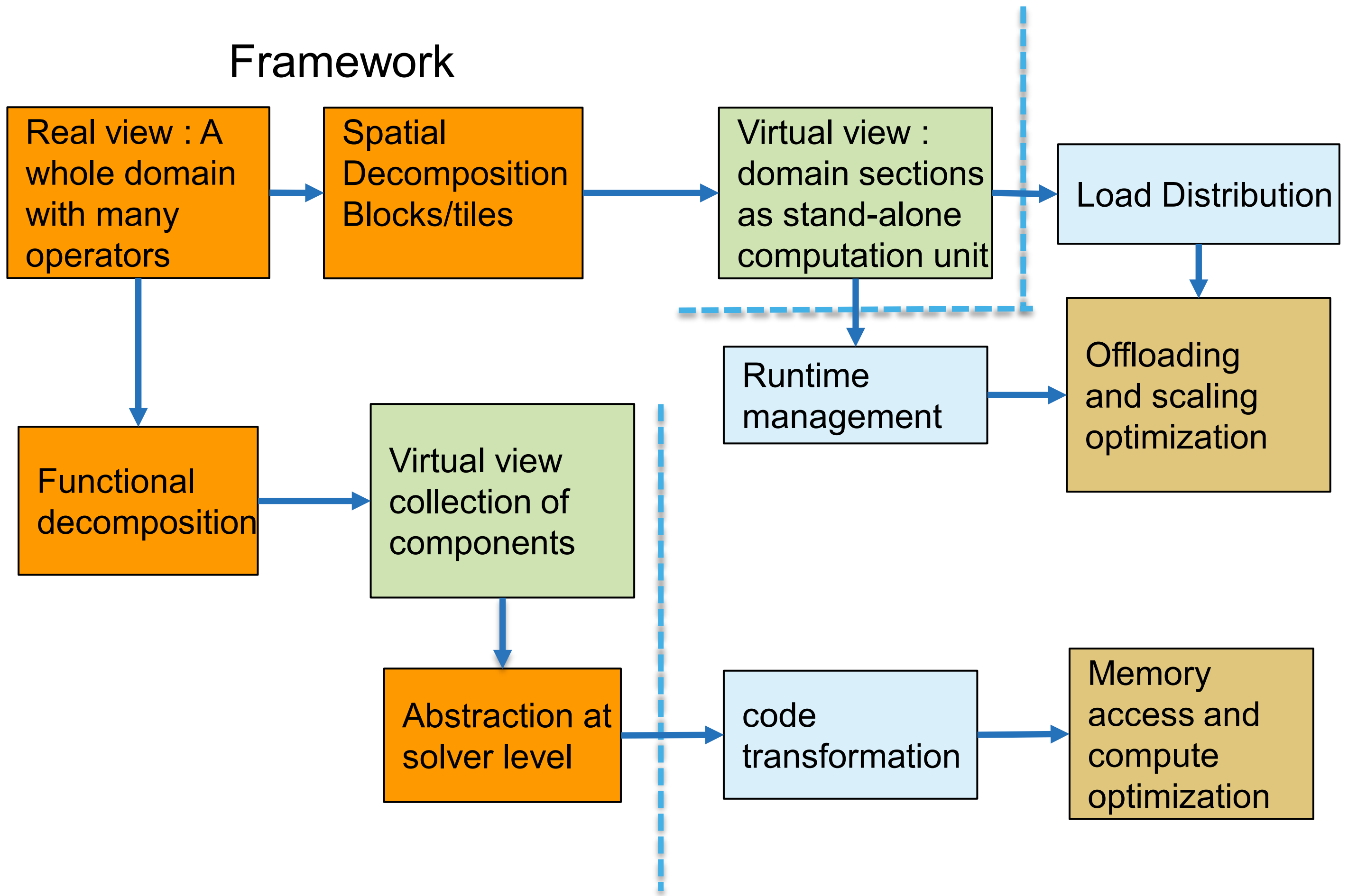


A Design Model for Separation of Concerns



This is where maximum change is likely

Features and Abstractions that must Come in



TAKEAWAYS

- DIFFERENTIATE BETWEEN SLOW CHANGING AND FAST CHANGING COMPONENTS OF YOUR CODE
- TAKE YOUR TIME TO UNDERSTAND THE REQUIREMENTS OF YOUR INFRASTRUCTURE
- IMPLEMENT SEPARATION OF CONCERNS
- DESIGN WITH PORTABILITY, EXTENSIBILITY, REPRODUCIBILITY AND MAINTAINABILITY IN MIND
- LEVERAGE EXISTING CAPABILITIES WHERE POSSIBLE

…….QUESTIONS ?

Agenda

Time (MDT)	Module	Topic	Speaker
1:00pm-1:05pm	00	Introduction	David E. Bernholdt, ORNL
1:05pm-1:15pm	01	Motivation and Overview of Best Practices in HPC Software Development	David E. Bernholdt, ORNL
1:15pm-1:45pm	02	Agile Methodologies	Rinku K. Gupta, ANL
1:45pm-2:00pm	03	Git Workflows	Rinku K. Gupta, ANL
2:00pm-2:20pm	04	Software Testing 1	David M. Rogers, ORNL
2:20pm-2:40pm		*Break (optional Q&A)*	*All*
2:40pm-3:00pm	05	Software Design	Anshu Dubey, ANL
3:00pm-3:15pm	06	Software Testing 2	David M. Rogers
3:15pm-3:40pm	07	Refactoring	Anshu Dubey, ANL
3:40pm-3:55pm	08	Reproducibility	David E. Bernholdt, ORNL
3:55pm-4:00pm	09	Summary	David E. Bernholdt, ORNL