

Refactoring Scientific Software

Presented by



<https://pesoproject.org>

Member of



*Consortium for the Advancement
of Scientific Software*

<https://cass.community>

With prior support from



David E. Bernholdt (he/him)
Oak Ridge National Laboratory

Software Sustainability track @ Argonne Training Program on Extreme-Scale Computing summer school

Contributors: David E. Bernholdt (ORNL), Anshu Dubey (ANL), Jared O'Neal



See slide 2 for
license details

License, Citation and Acknowledgements

License and Citation



- This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0).
- **The requested citation the overall tutorial is:** David E. Bernholdt, Software Sustainability track, in Argonne Training Program on Extreme-Scale Computing, St. Charles, Illinois, 2026. DOI: [10.6084/m9.figshare.33139910](https://doi.org/10.6084/m9.figshare.33139910).
- Individual modules may be cited as *Speaker, Module Title, in Tutorial Title, ...*

Acknowledgements

- This work was supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Next-Generation Scientific Software Technologies (NGSST) and Scientific Discovery through Advanced Computing (SciDAC) programs.
- 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 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 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 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.

What is Refactoring

Definition: Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

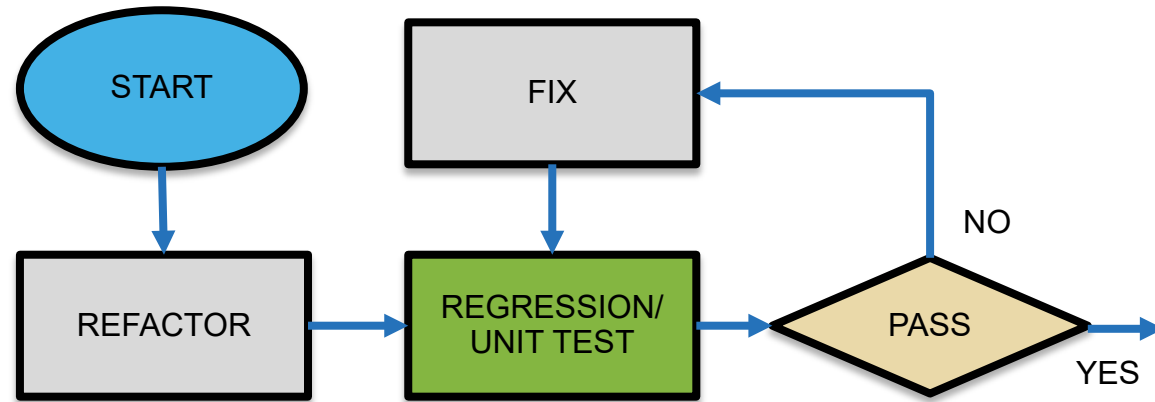
- Different from development
 - You have a working code
 - You know and understand the behavior
 - You have a baseline that you can use for comparison

What is Refactoring

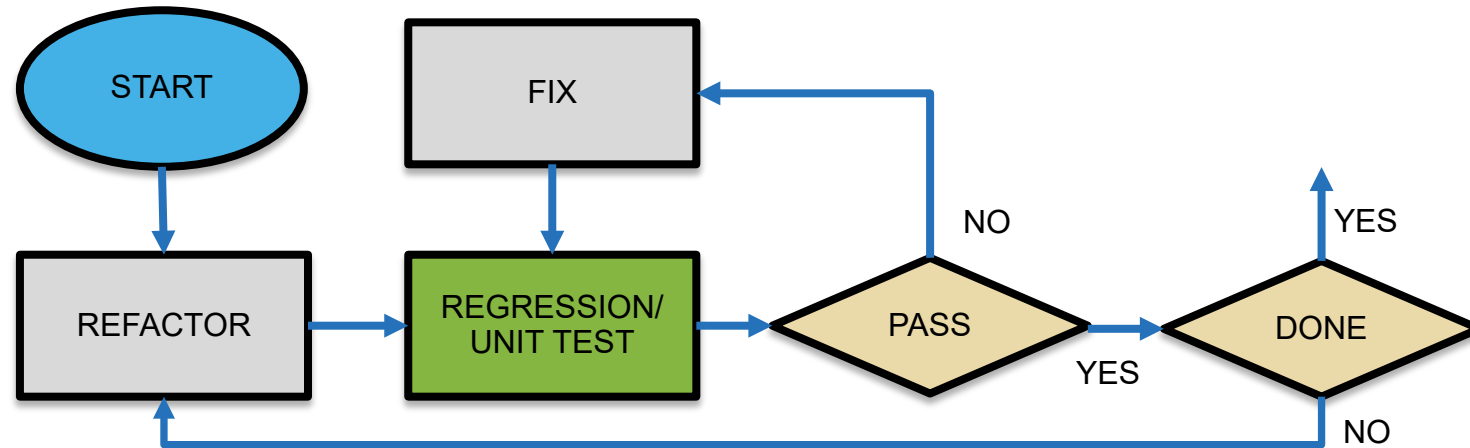
Definition: Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

- Different from development
 - You have a working code
 - You know and understand the behavior
 - You have a baseline that you can use for comparison
- General motivations
 - Modularity enhancement
 - Improve sustainability
 - Release to outside users
 - Easier to use and understand
 - Port to new platforms
 - Performance portability
 - Expand capabilities
 - Structural flexibility

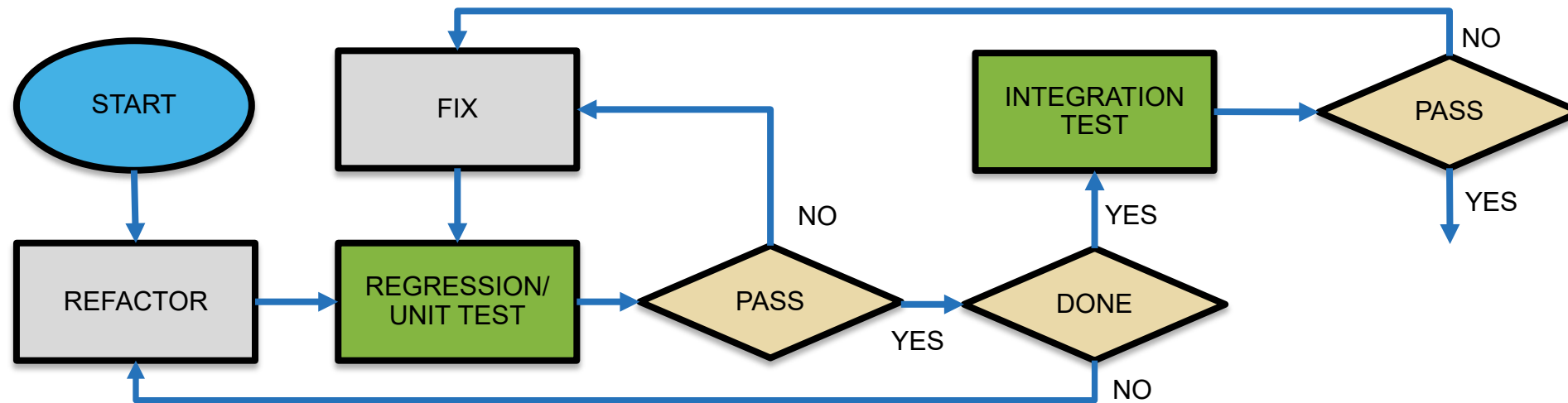
An Example Workflow



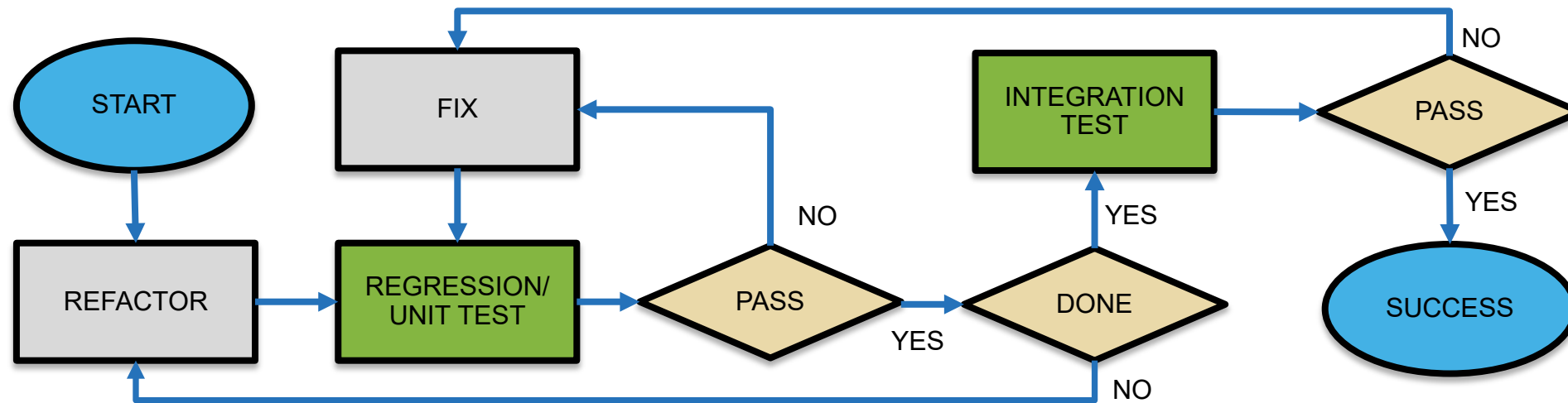
An Example Workflow



An Example Workflow

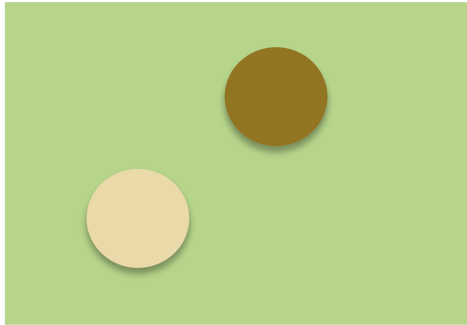


An Example Workflow

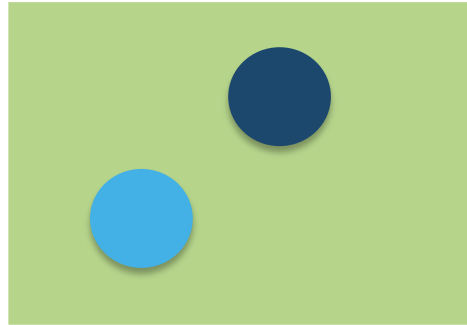


Map from Here to There: On ramp plan

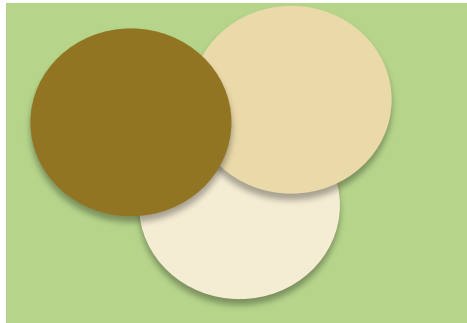
Proportionate to the scope



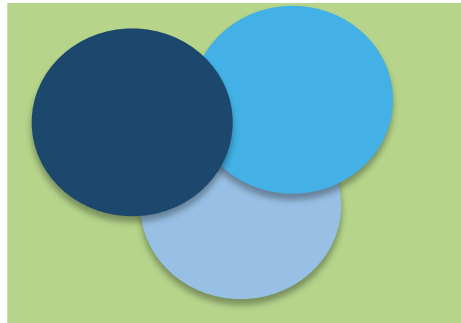
All at once



Scattered independent changes - May be OK

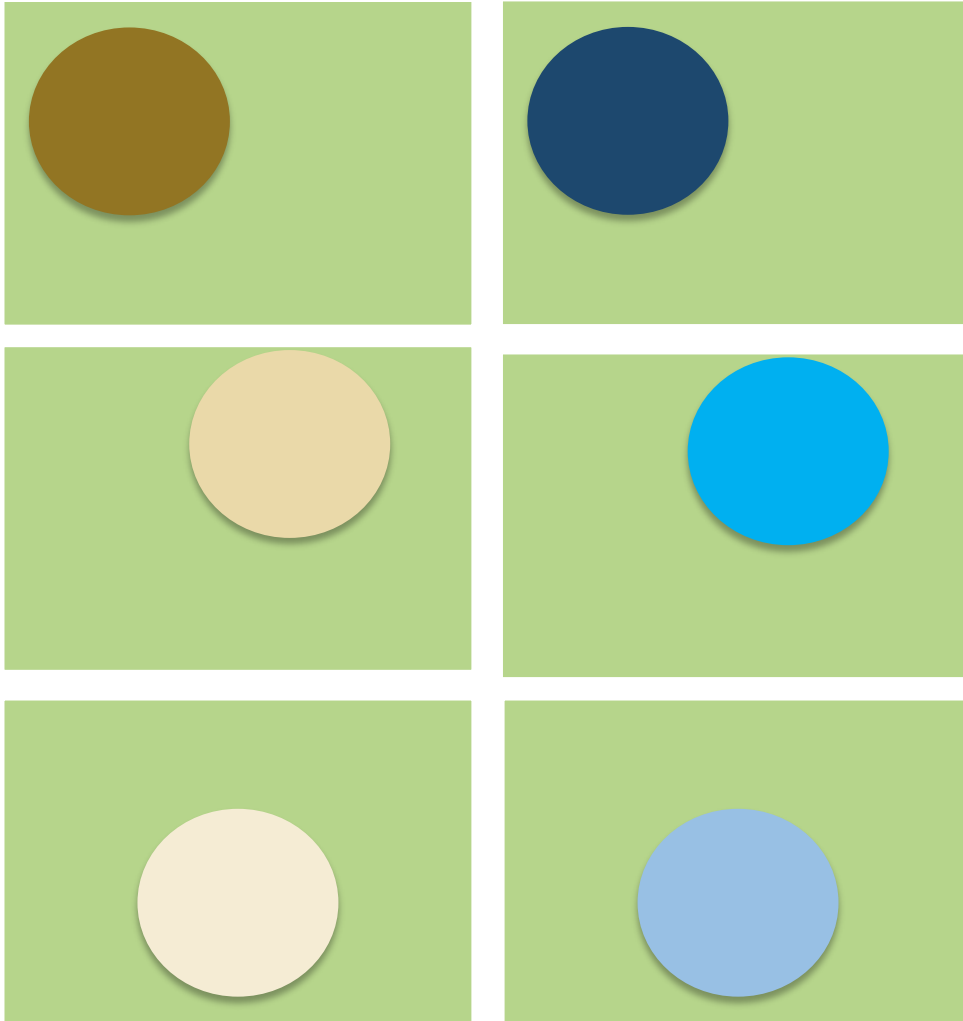


All at once



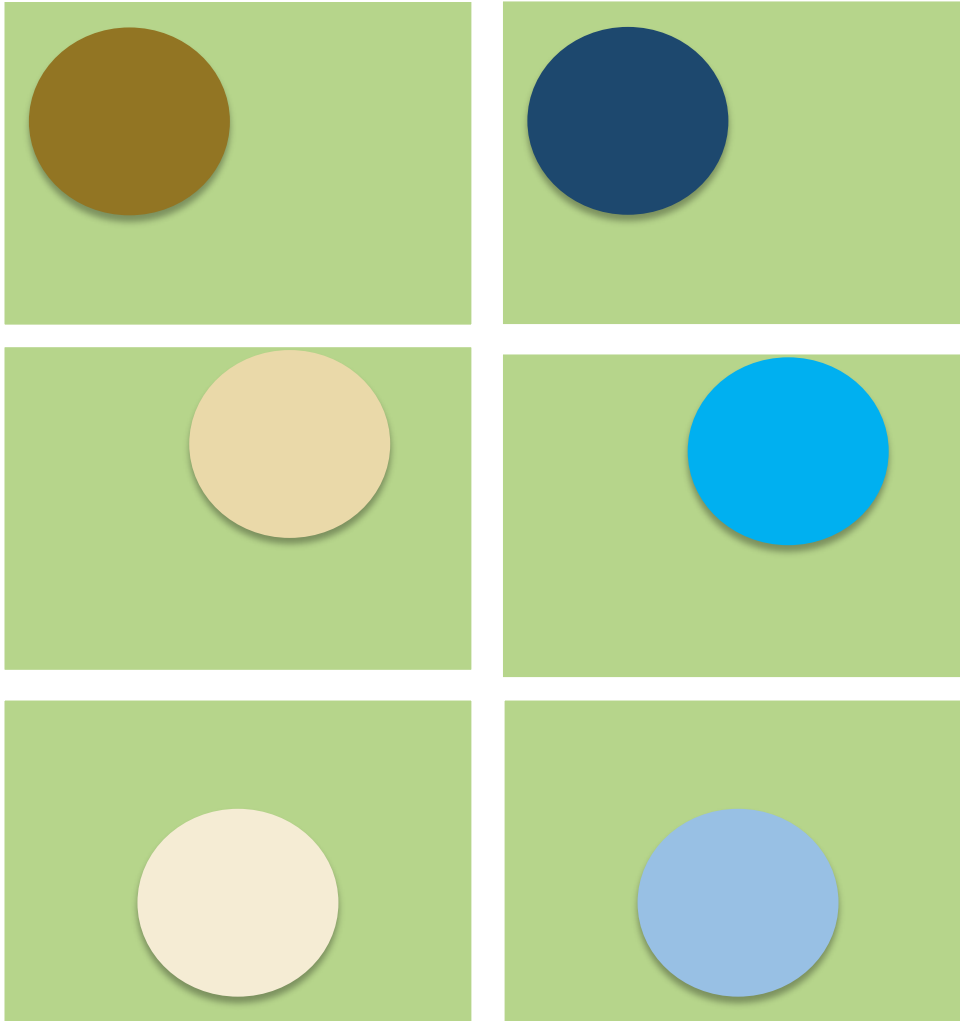
Invasive large-scale change in the code - Bad idea

Map from Here to There: On ramp plan1

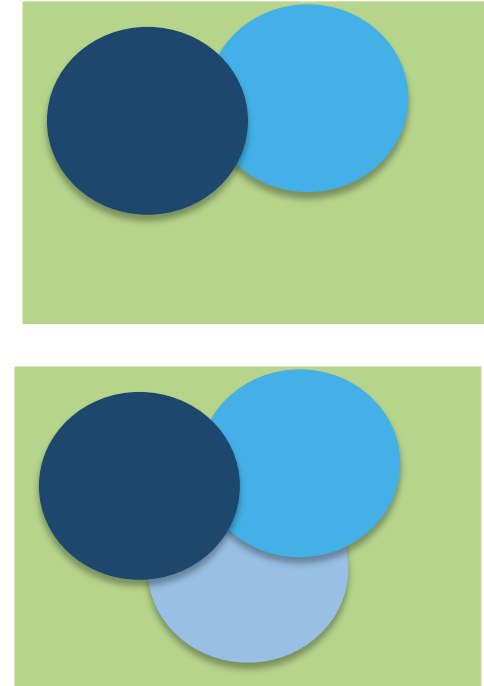


- Turn off all modules except for the one being refactored.
- Have a way of testing in intermediate stages
- Do this for all modules that need refactoring independently

Map from Here to There: On ramp plan1

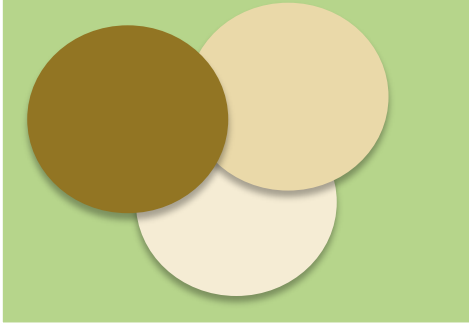


- Turn off all modules except for the one being refactored.
- Have a way of testing in intermediate stages
- Do this for all modules that need refactoring independently



- One by one turn on more than one refactored module

Map from Here to There: On ramp plan2

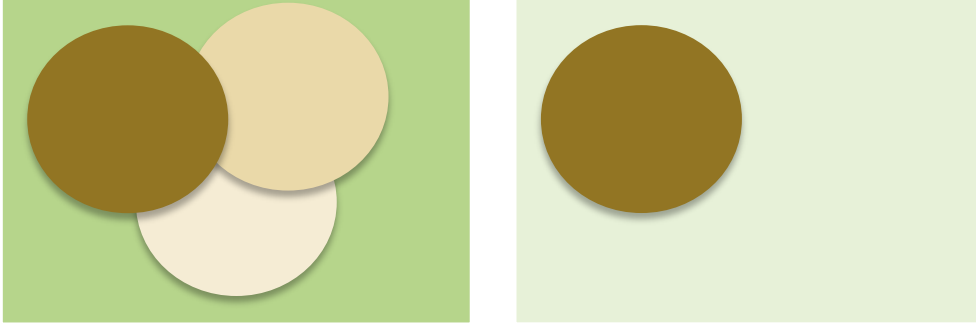


Map from Here to There: On ramp plan2



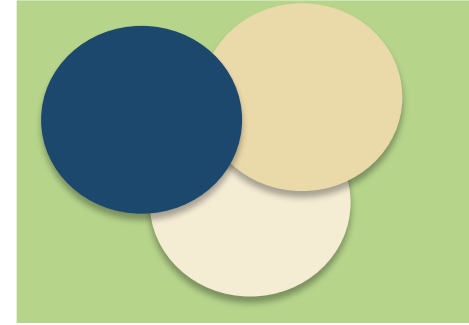
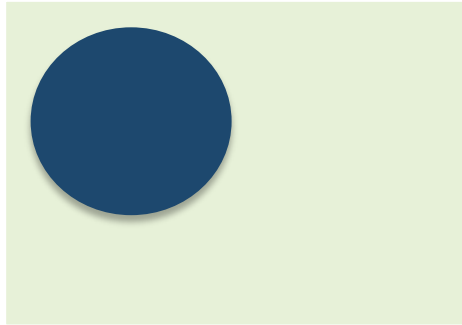
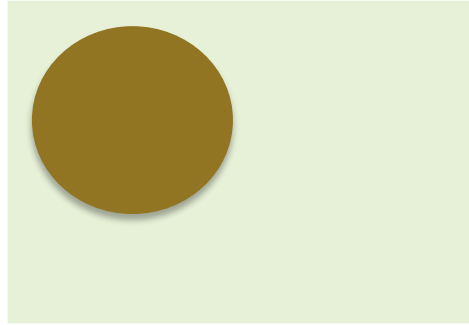
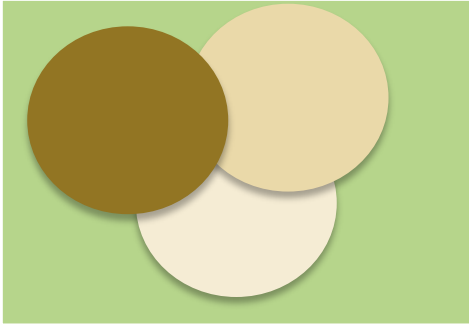
- Build a separate environment for testing refactored module

Map from Here to There: On ramp plan2



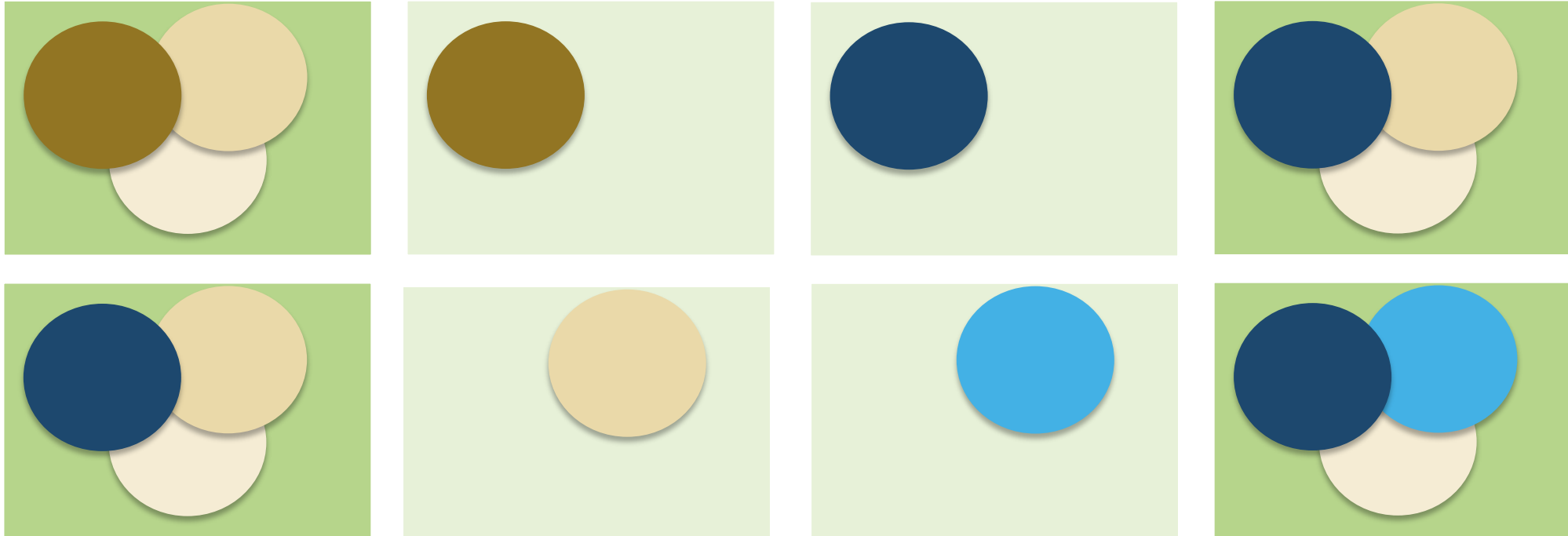
- Build a separate environment for testing refactored module
- Copy over the module in this isolated environment

Map from Here to There: On ramp plan2



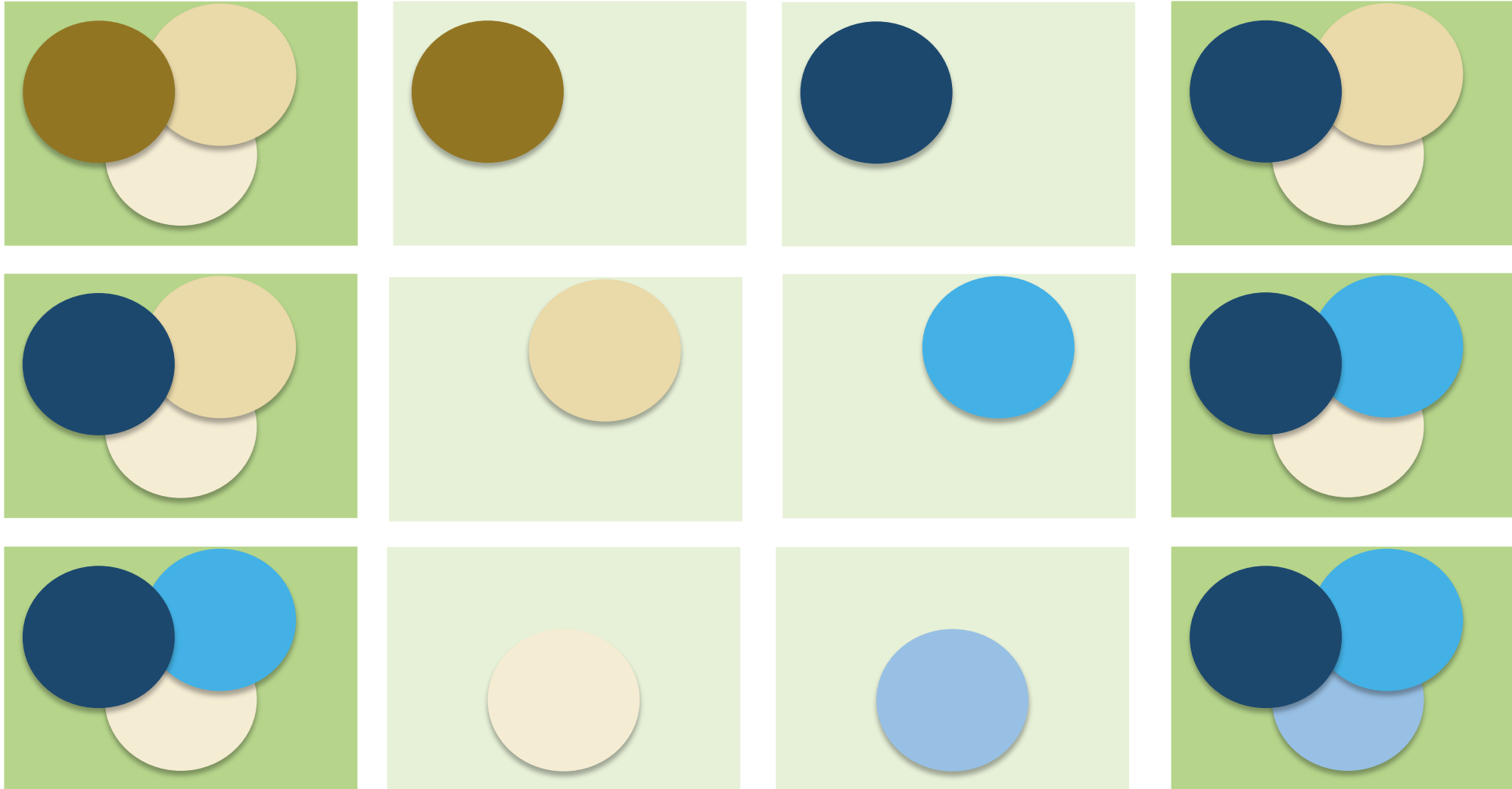
- Build a separate environment for testing refactored module
- Copy over the module in this isolated environment
- Put back refactored module

Map from Here to There: On ramp plan2



- Build a separate environment for testing refactored module
- Copy over the module in this isolated environment
- Put back refactored module

Map from Here to There: On ramp plan2



- Build a separate environment for testing refactored module
- Copy over the module in this isolated environment
- Put back refactored module

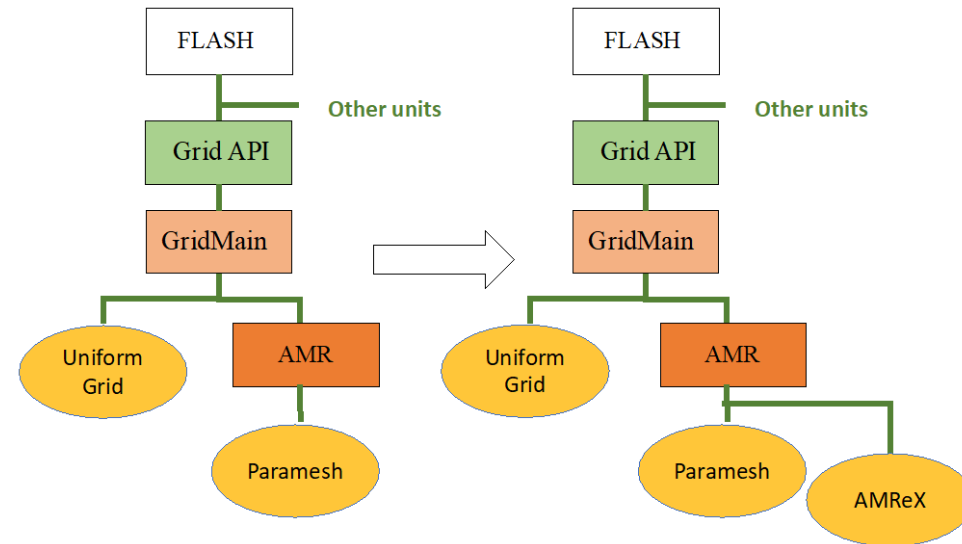
A Real-World Example: FLASH to Flash-X

Refactoring to supporting a different AMR library

Goal: Replace Paramesh with AMReX

Plan: Getting there from here

- On ramping
- Design
- Intermediate steps
- Realizing the goal



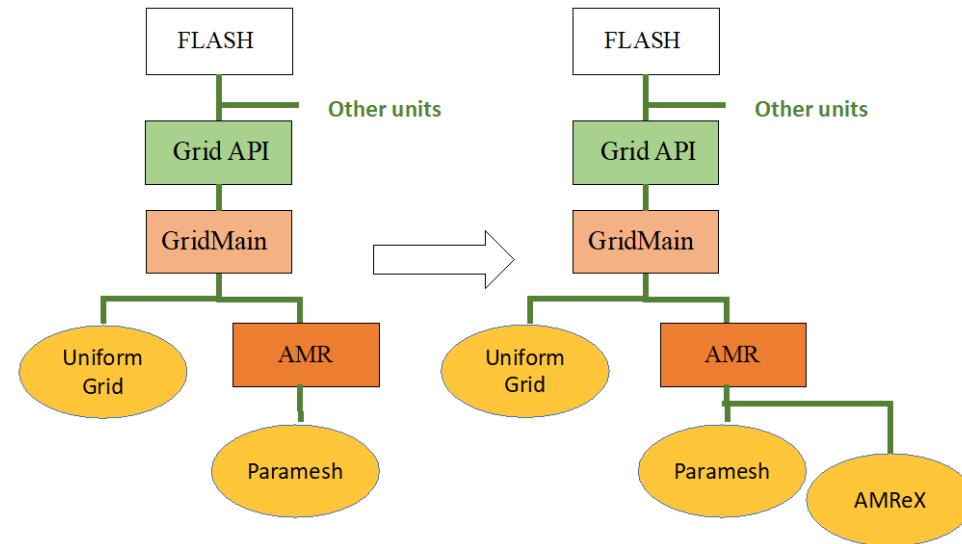
A Real-World Example: FLASH to Flash-X

Refactoring to supporting a different AMR library

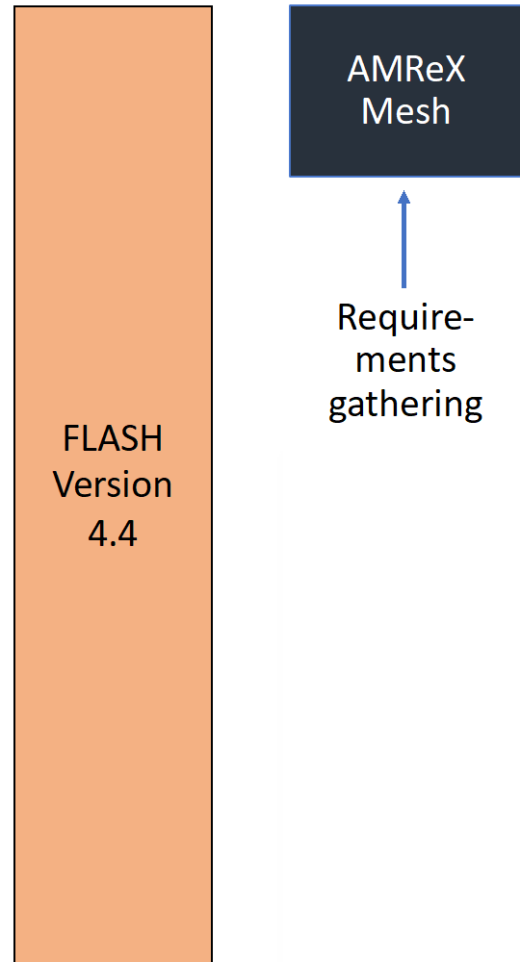
Goal: Replace Paramesh with AMReX

Plan: Getting there from here

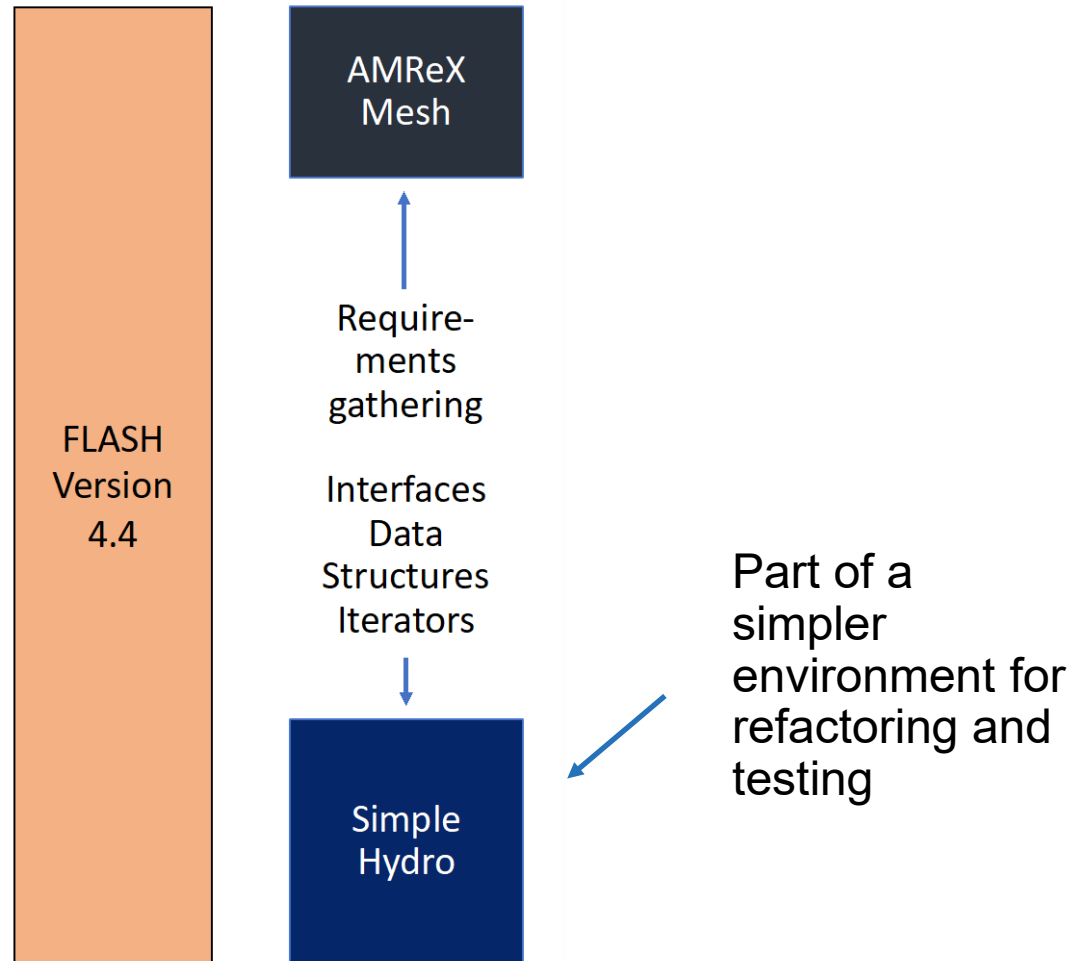
- On ramping
 - Design
 - Intermediate steps
 - Realizing the goal
-
- Cost estimation
 - Expected developer time
 - Extent of disruption in production schedules
 - Get a buy-in from the stakeholders
 - That includes the users
 - For both development time and disruption



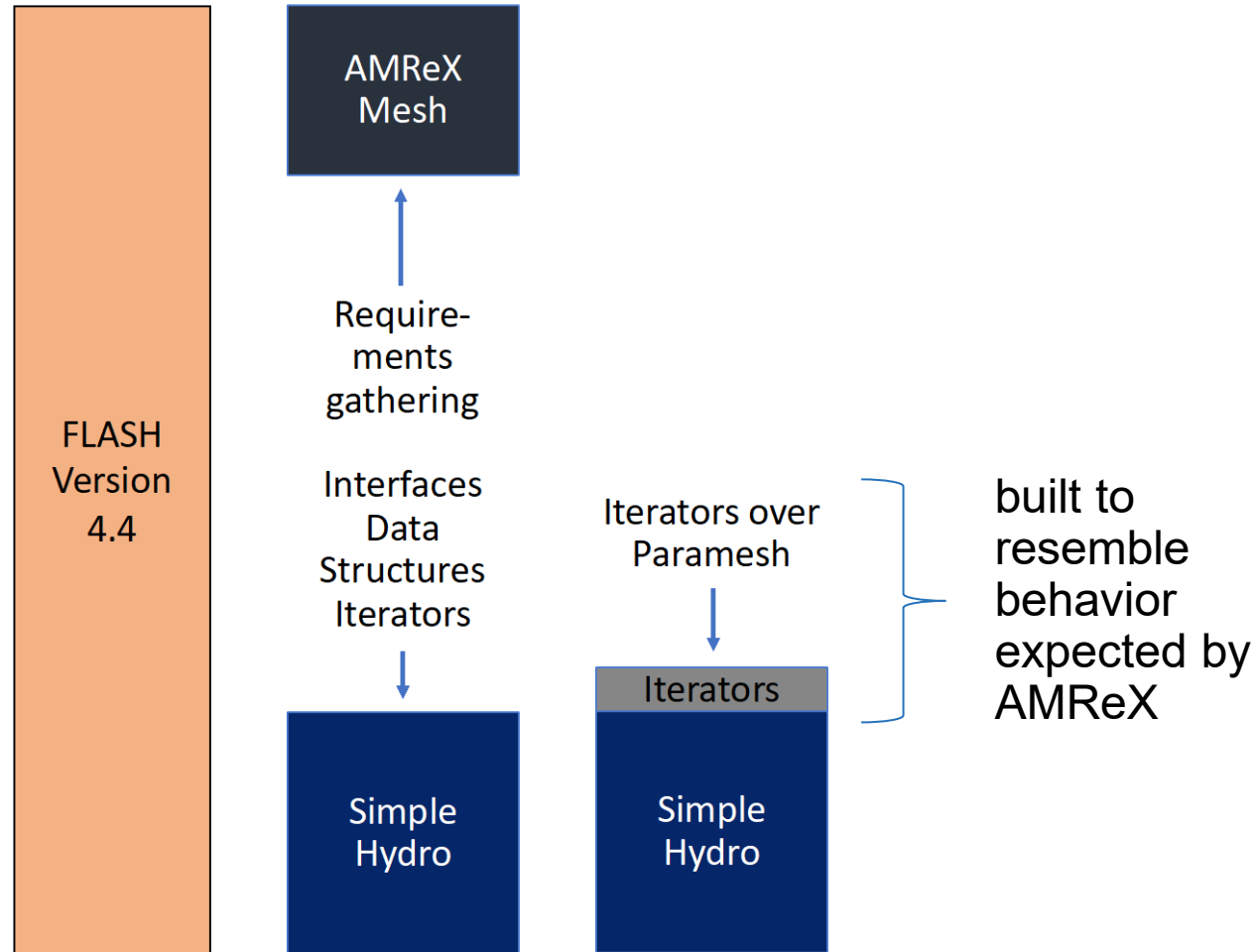
Steps in the Flash-X Refactoring : a mix of strategies



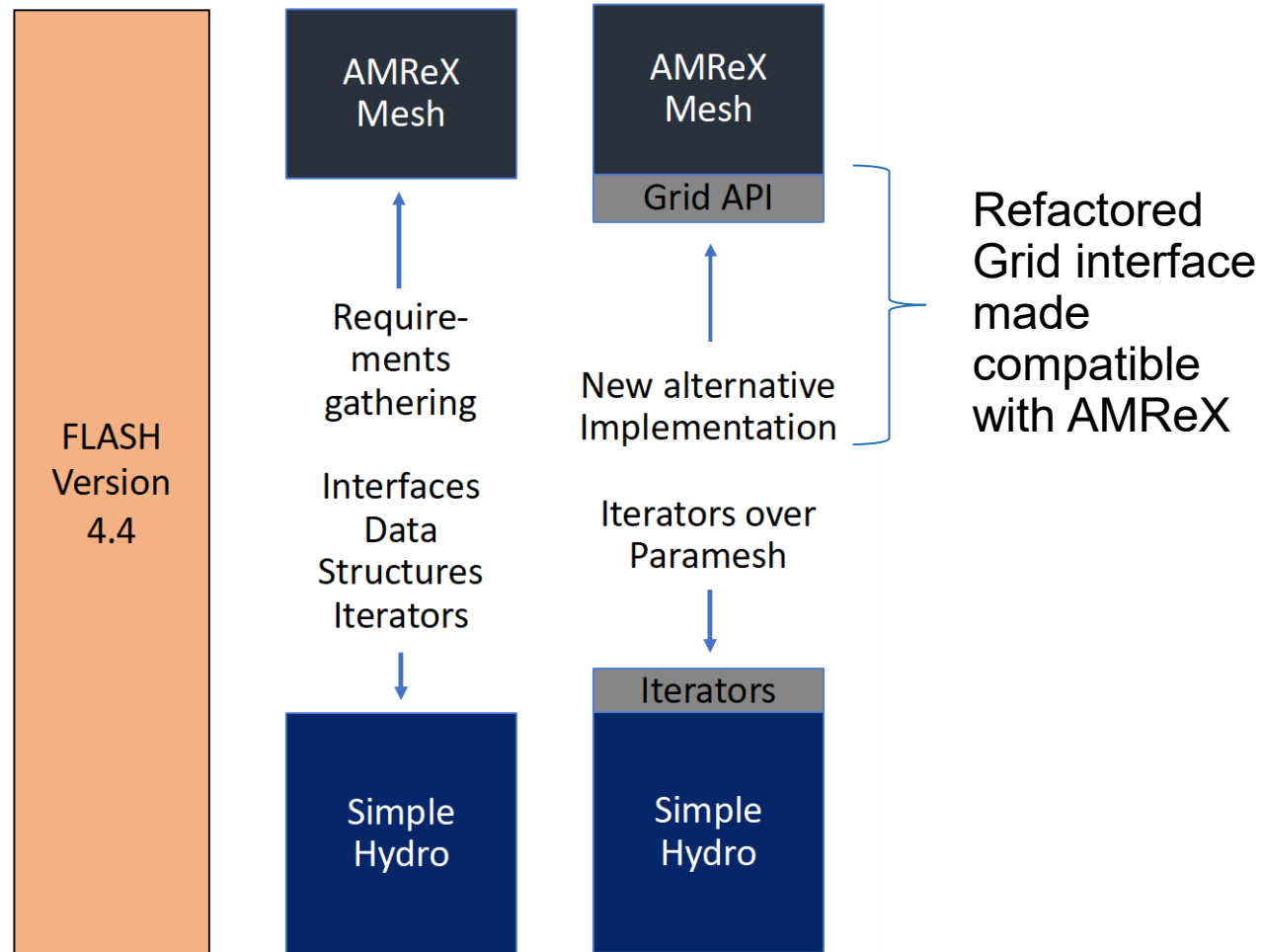
Steps in the Flash-X Refactoring : a mix of strategies



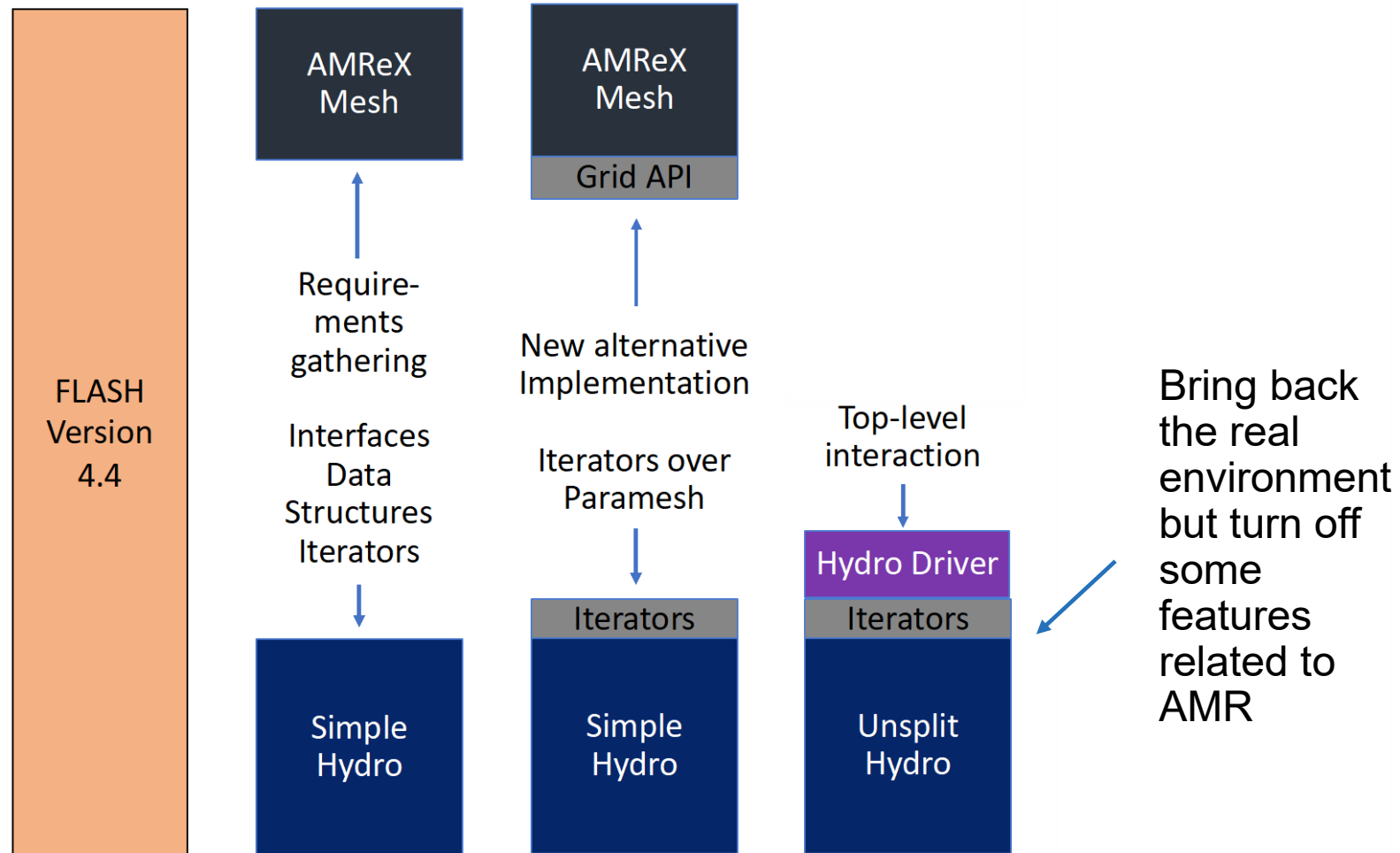
Steps in the Process



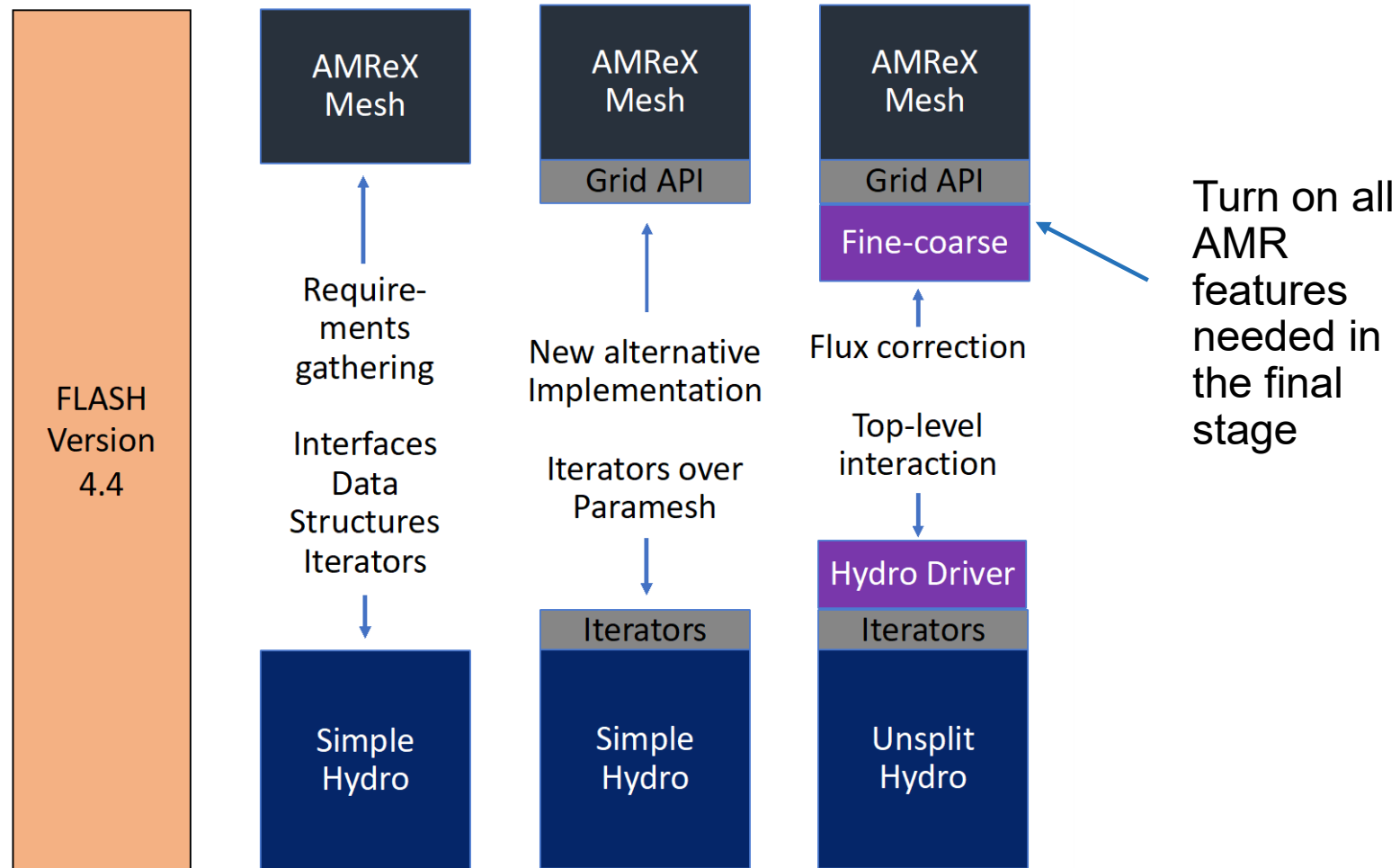
Steps in the Process



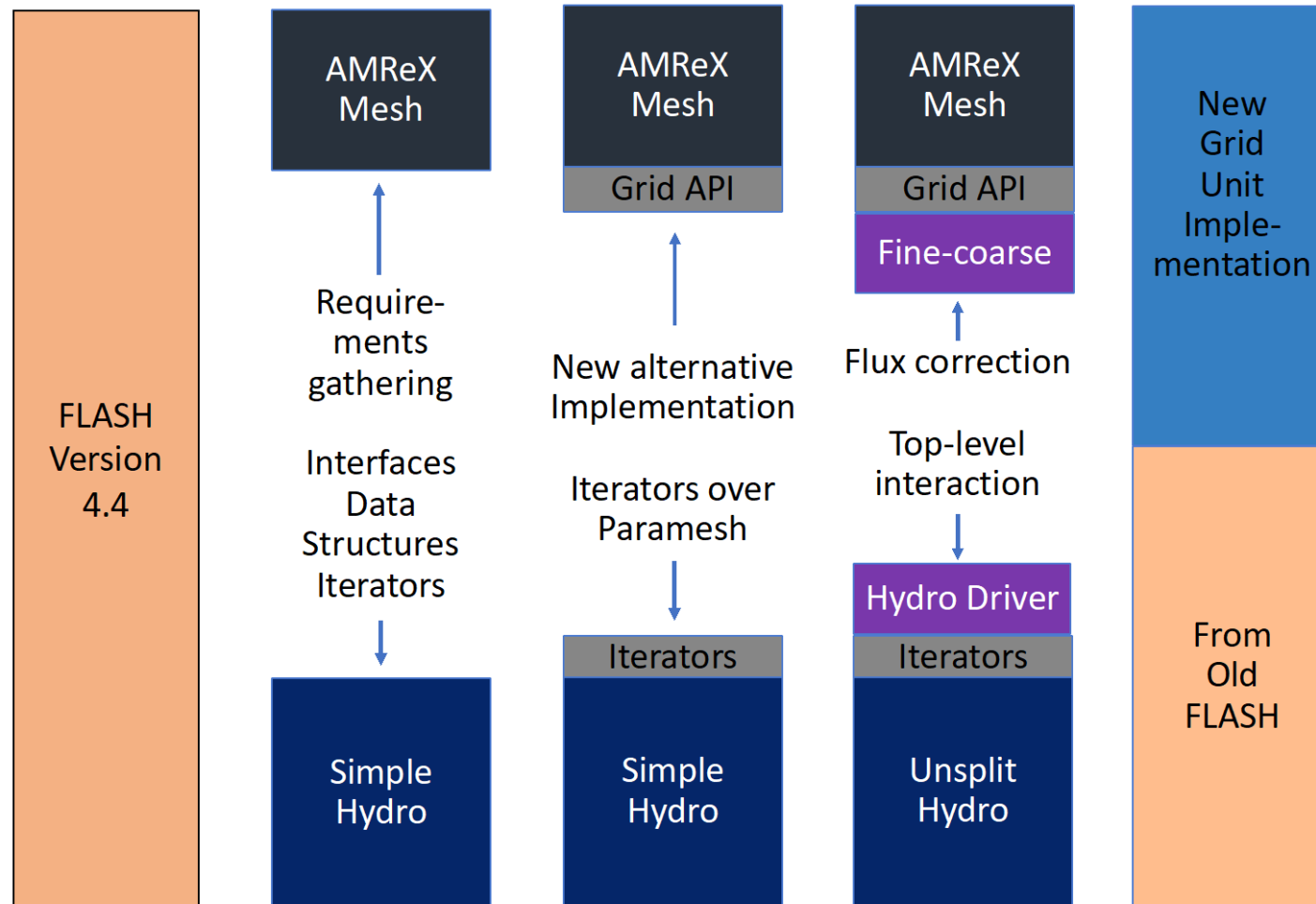
Steps in the Process



Steps in the Process



Steps in the Process



To Have a Good Outcome from Refactoring

1. Know why
2. Know how much
3. Know the cost
4. Plan
5. Have strong testing and verification
6. Get buy-in from stakeholders