Putting it All Together

Using Numerical Packages In Practice

Presented to

ATPESC 2020 Participants

Ann Almgren

Deputy Director, ECP Block-Structured AMR Co-Design Center

Date 08/04/2020

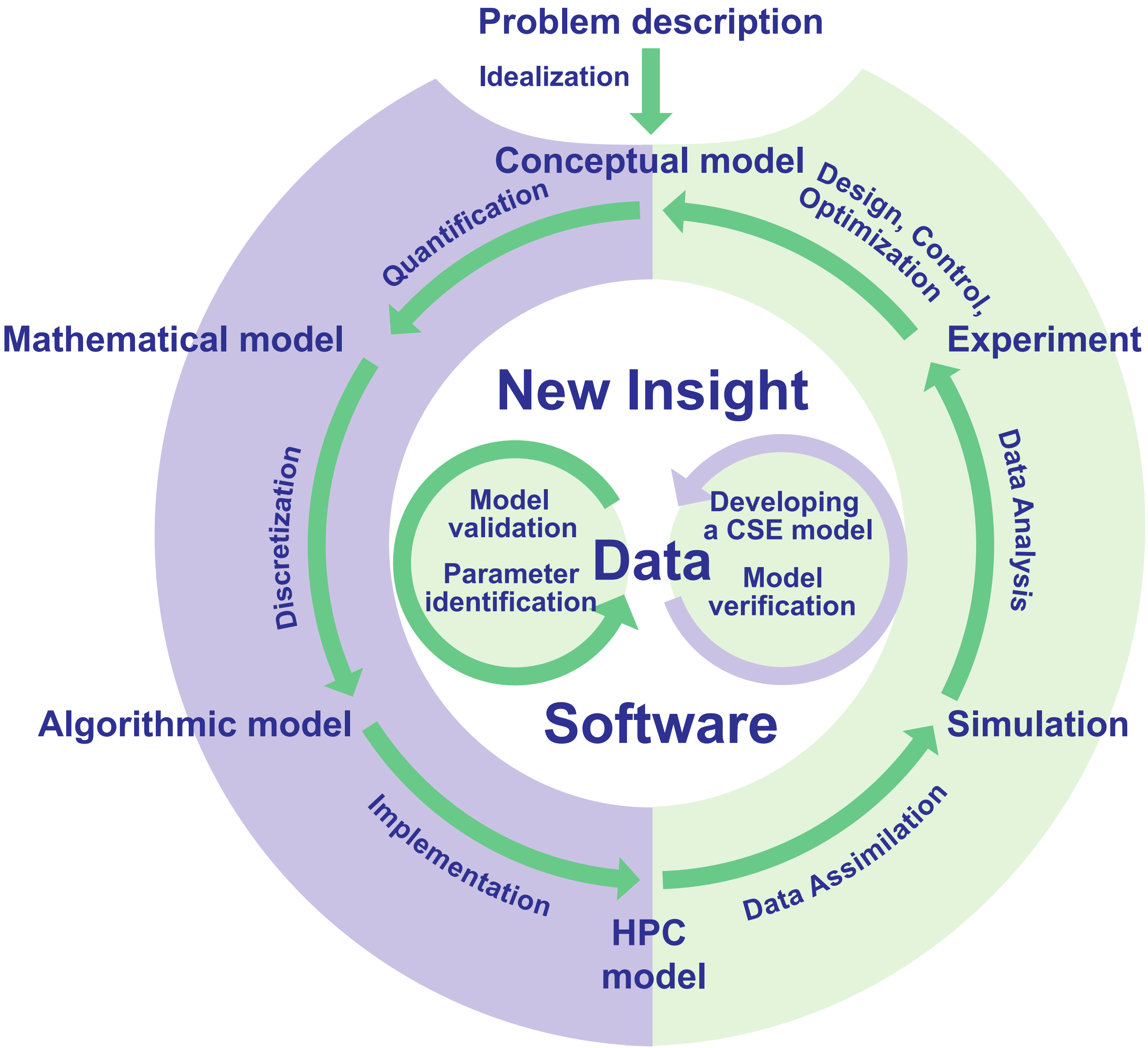
U.S. DEPARTMENT OF ENERGY | Office of Science

ATPESC Numerical Software Track

EXASCALE COMPUTING PROJECT

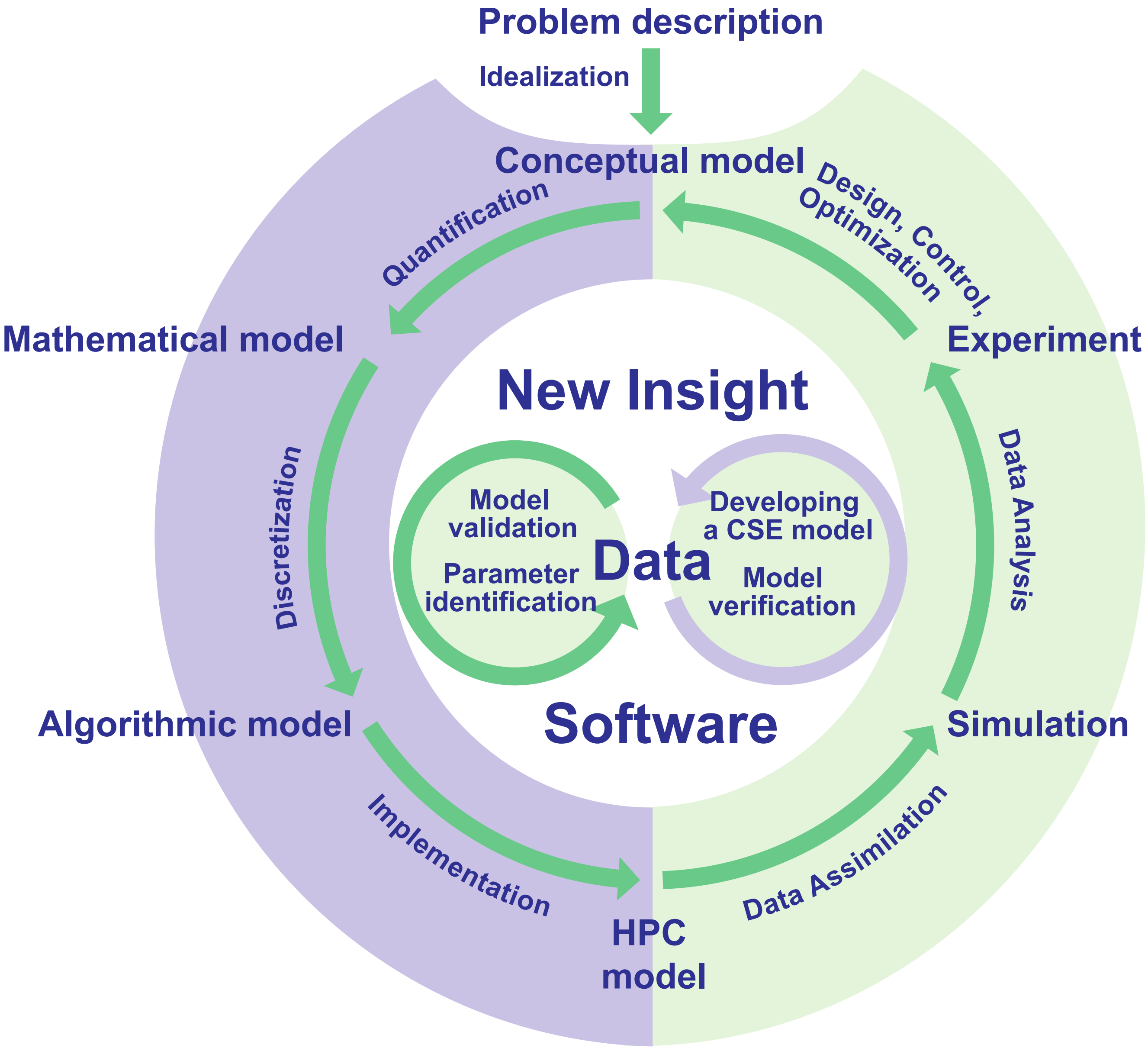
Argonne NATIONAL LABORATORY   BERKELEY LAB   Sandia National Laboratories   Rensselaer   SMU

Recall from this morning's introduction:



Recall from this morning's introduction:

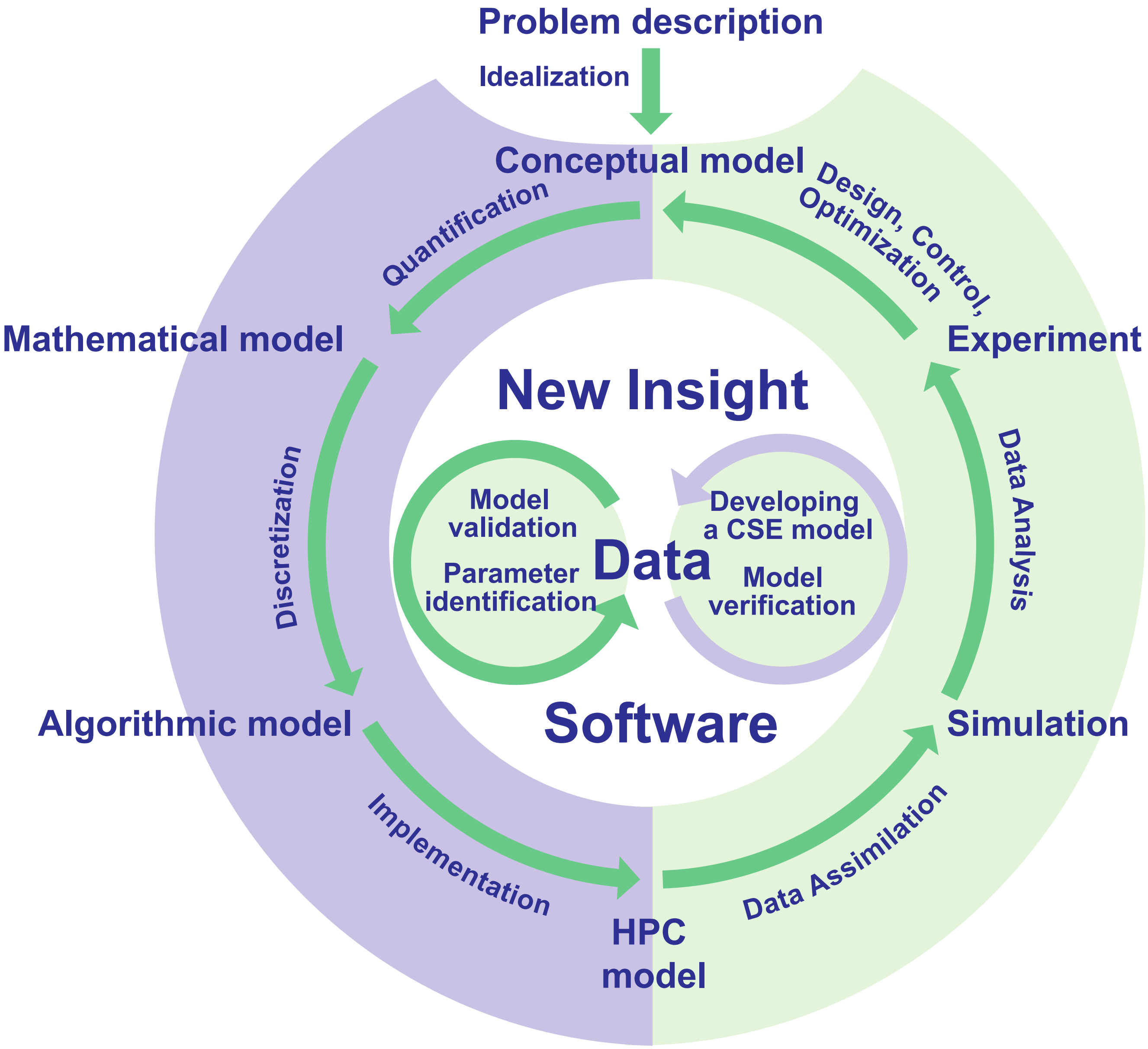
None of us can actually do it all



Recall from this morning’s introduction:

None of us can actually do it all

So where do you want to spend your time?



EXASCALE COMPUTING PROJECT

Key steps of simulation science application development

- Physical model
  - Expertise may be very domain-specific

- Mathematical model
  - Expertise may require detailed mathematical knowledge

- Discretization and algorithm development
  - Expertise includes knowing regimes of applicability, stability, approximation, error bounds

- Parallel implementation
  - Expertise in hardware, software stack and parallel programming models

That’s a lot of expertise!

Very few of us are experts in all of these areas. So how do we optimize the insight/impact of our computational science?

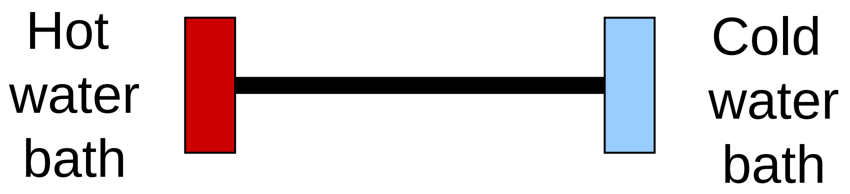
- Team science – in an ideal world we could work in teams that have all the relevant expertise within one team
- That’s not always possible – so one way to broadly share expertise is through software libraries
  - Expertise in discretization and algorithm development
  - Expertise in hardware, software stack and parallel programming models

In the short-term we often prefer to do things ourselves

$$\nabla^2 T = 0 \in \Omega$$

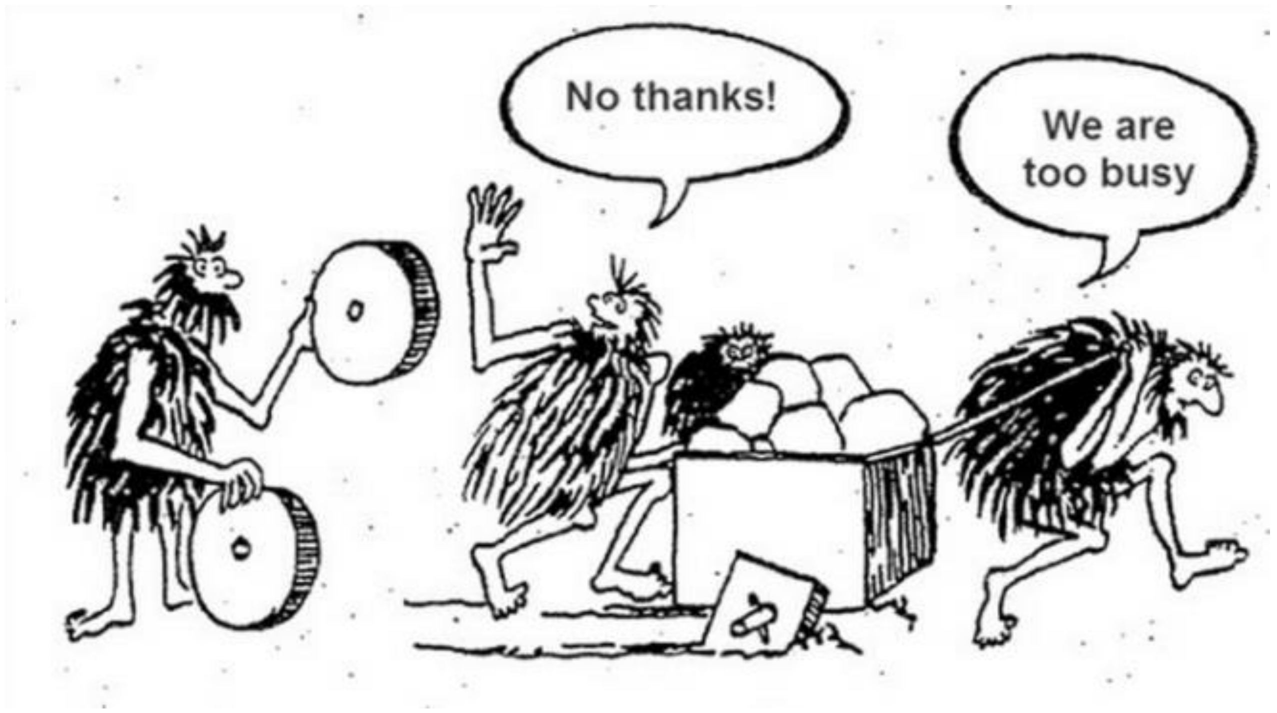
$$T(0) = 180^o$$

$$T(1) = 0^o$$



Hot water bath

Cold water bath



For the 1-D heat equation why bother learning a software package?

Sometimes simple is good

We can prototype in matlab, build simple serial implementations, and demonstrate proof-of-concept.

This can be good:

- New algorithms are often designed and validated in this mode.
- Sometimes writing your own version of a known technology (e.g. multigrid solver) is worth it -- “learning by doing”

This can be bad:

- Our own implementations are more likely to lack generality, be inefficient or even buggy.
- How much time do we spend “reinventing the wheel?”

The “supply” side of software libraries

Software libraries/frameworks/tools are made by real people.

The people aspect matters

- Software developers know a lot about their product
- But they don’t necessarily know exactly what you need

Communication/collaboration is an important part of the process; it’s good for the developer as well as the user!

Why don’t people “just” use software libraries

Lack of knowledge – how do you know whether the right tool even exists?

And if it exists: Where do you find it? How do you use it? Will it work with your other tools?

Why don’t people “just” use software libraries

Frustration!   It can be really frustrating to not have the tool do what you want as well as you want. And how do you tell whether it’s you or the tool?

So how can you find the right tool – if it exists - and how do you learn how to use it correctly?

Ideal solution: a "toolbox" of compatible (interoperable) tools that "just work"

- This is exactly what the software developers are working towards

- But it takes time and resources

- The developer/user interaction can be a win-win

On a practical level, there are trade-offs

Challenges

- Something new to learn
- Hard to predict show-stoppers
- Not always plug-n-play
- Trusting the work of others
- Overhead of collaborating
- Funding priorities

Advantages

- Key challenges addressed well
  - Portable, Performant, Scalable, Interoperable
- Numerics are well tested/vetted
- Functionality is often more general than you would have made yourself
- More science, more impact; less time writing/debugging software
- Become part of a community – for collaboration and help

How do we tip the balance?

Challenge		Mitigation
Something new to learn	↔	Many examples and documentation
Hard to predict show-stoppers	↔	Engage package developers early
Not always plug-n-play	↔	Submit build issues
Trusting others	↔	Identify or develop tests
Overhead of collaborating	↔	Builds relationships
Funding priorities	↔	Add to the package yourself

The point of open source is to encourage use

Package teams want users to make progress.

If package is missing a crucial feature, ask.

And … many contributors started as users!

If you like to code … and you like to contribute … think about joining the “supply side”!

It’s not that big a leap – and it’s a win-win.

See also Track 7:

Software Productivity & Sustainability (Aug 6)

Auspices and Disclaimer

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program and the Exascale Computing Project funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research

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

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.