SAND2014-19756C

Exceptional service in the national interest

Sandia National Laboratories

Toward the Next Generation of Portable, Scalable HPC Applications

IDEAS productivity

Michael A. Heroux
Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Trilinos

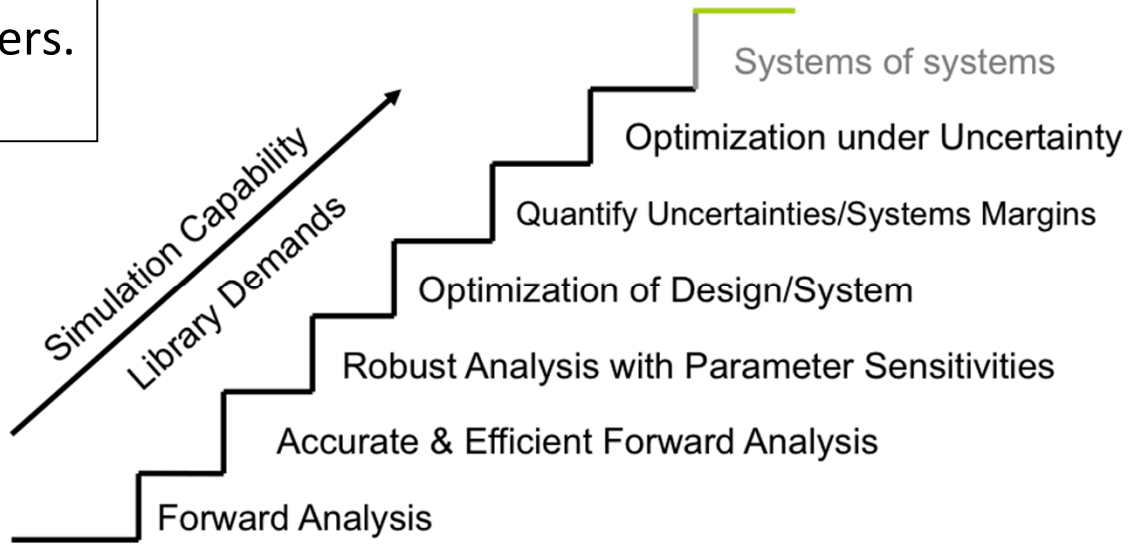
R&D 100

Laptops to Leadership systems

Optimal Kernels to Optimal Solutions:

- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

Transforming Computational Analysis To Support High Consequence Decisions



- 60 Packages.
- Other distributions:
  - Cray LIBSCI
  - Public repo.

Each stage requires *greater performance* and *error control* of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.

Interoperable Design of Extreme-scale Application Software (IDEAS)

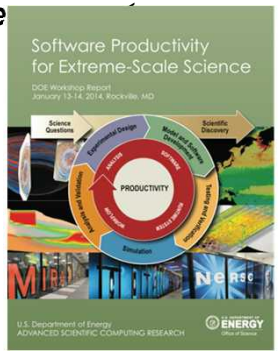
Motivation

Enable ***increased scientific productivity,*** realizing the potential of extreme- scale computing, through ***a new interdisciplinary and agile approach to the scientific software***

Objectives

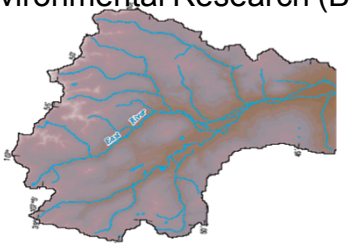
Address confluence of trends in hardware and increasing demands for predictive multiscale, multiphysics simulations.

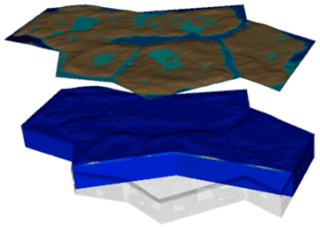
Respond to trend of continuous refactoring with efficient agile software engineering methodologies and improved software design.

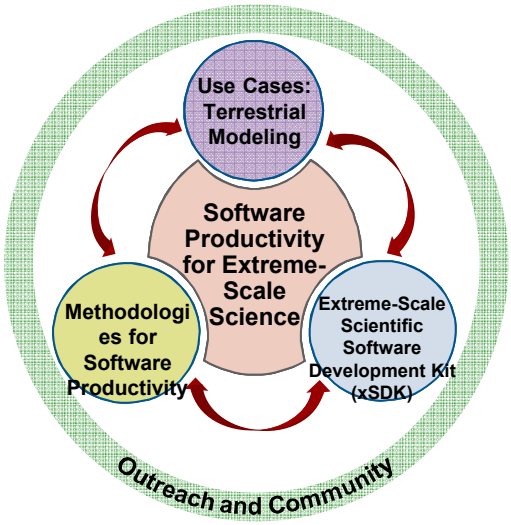


Impact on Applications & Programs

Terrestrial ecosystem ***use cases tie IDEAS to modeling and simulation goals*** in two Science Focus Area (SFA) programs and both Next Generation Ecosystem Experiment (NGEE) programs in DOE Biologic and Environmental Research (BER).







Approach

ASCR/BER partnership ensures delivery of both crosscutting methodologies and metrics with impact on real application and programs.

Interdisciplinary multi-lab team (ANL, LANL, LBNL, LLNL, ORNL, PNNL, SNL)

ASCR Co-Leads: Mike Heroux (SNL) and Lois Curfman McInnes (ANL)

BER Lead: David Moulton (LANL)

Topic Leads: David Bernholdt (ORNL) and Hans Johansen (LBNL)

Integration and synergistic advances in three communities deliver scientific productivity; outreach establishes a new holistic perspective for the broader scientific community.

Sandia National Laboratories

BACKGROUND & MOTIVATION

A Confluence of Trends

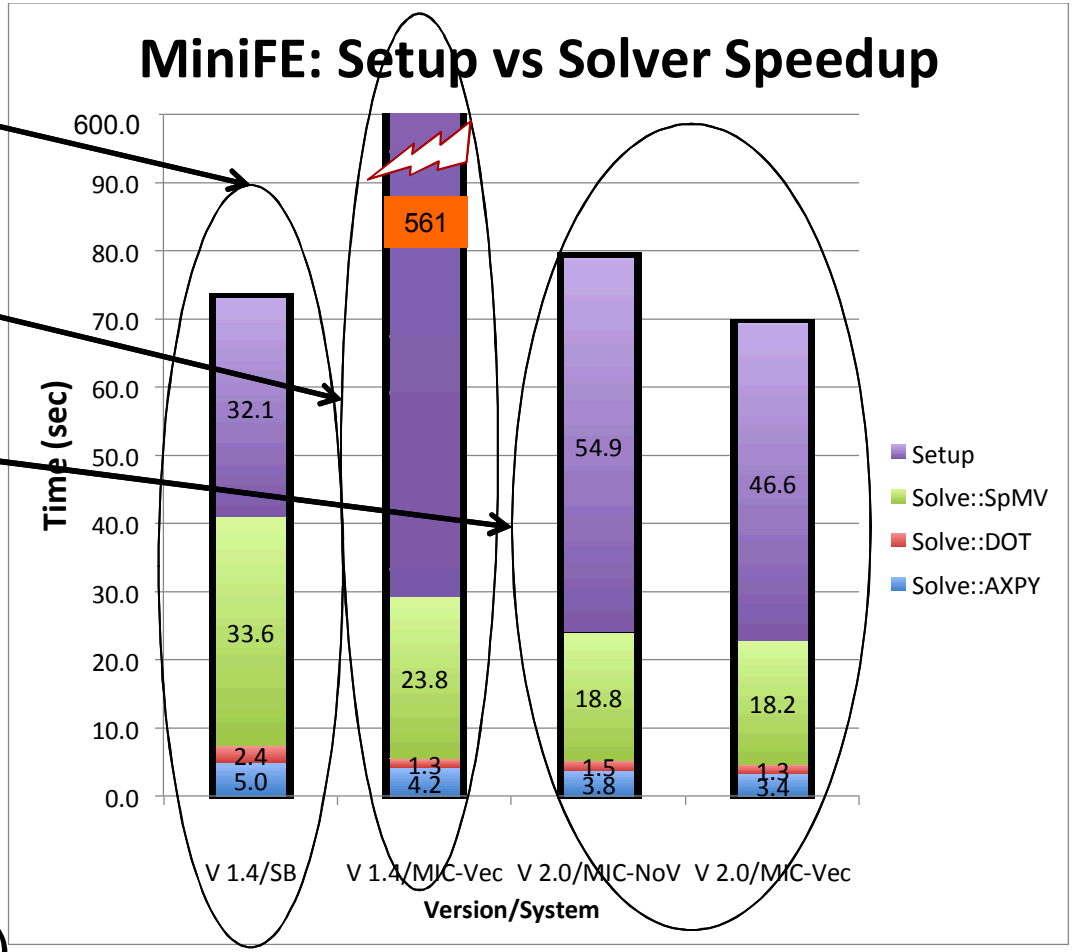
Sandia National Laboratories

- Fundamental trends:
  - Disruptive HW changes: Requires thorough alg/code refactoring.
  - Demands for coupling: Multiphysics, multiscale.
- Challenges:
  - Need 2 refactorings: 1+ε, not 2-ε. Really: Continuous change.
  - Modest app development funding: No monolithic apps.
  - Requirements are unfolding, evolving, not fully known *a priori*.
- Opportunities:
  - Better design and SW practices & tools are available.
  - Better SW architectures: Toolkits, libraries, frameworks.
- Basic strategy: Focus on productivity.

The work ahead of us: Threads and vectors

MiniFE 1.4 vs 2.0 as Harbingers

- Typical MPI-only run:
  - Balanced setup vs solve
- First MIC run:
  - Thread/vector solver
  - No-thread setup
- V 2.0: Thread/vector
  - Lots of work:
    - Data placement, const /restrict declarations, avoid shared writes, find race conditions, …
  - Unique to each app
- Opportunity: Look for new crosscutting patterns, libraries (e.g., libs of data containers)



Sandia National Laboratories

If I had eight hours to chop down a tree, I would spend six sharpening my axe.

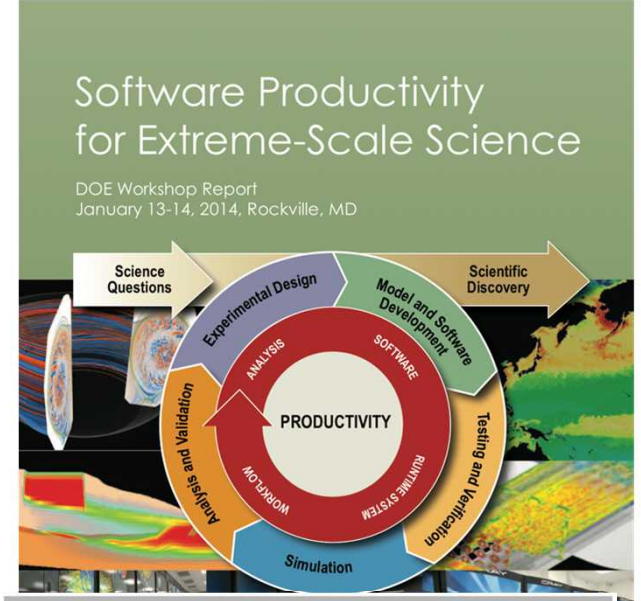
- Abraham Lincoln

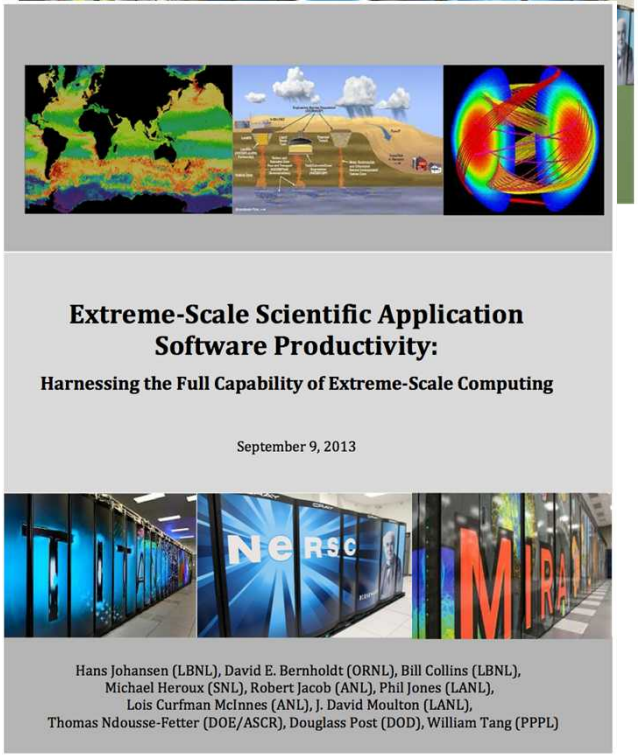
PRODUCTIVITY

BETTER, FASTER, CHEAPER: PICK ALL THREE

Productivity Emphasis

- *Scientific* Productivity.
- Many design choices ahead.
- Productivity emphasis:
  - Metrics.
  - Design choice process.
- Software ecosystems: Rational option
  - Not enough time to build monolithic.
  - Too many requirements.
  - Not enough funding.
- Focus on actionable productivity metrics.
  - Optometrist model: which is better?
  - Global model: For “paradigm shifts”.





Sandia National Laboratories

PARALLEL PROGRAMMING & PRODUCTIVITY

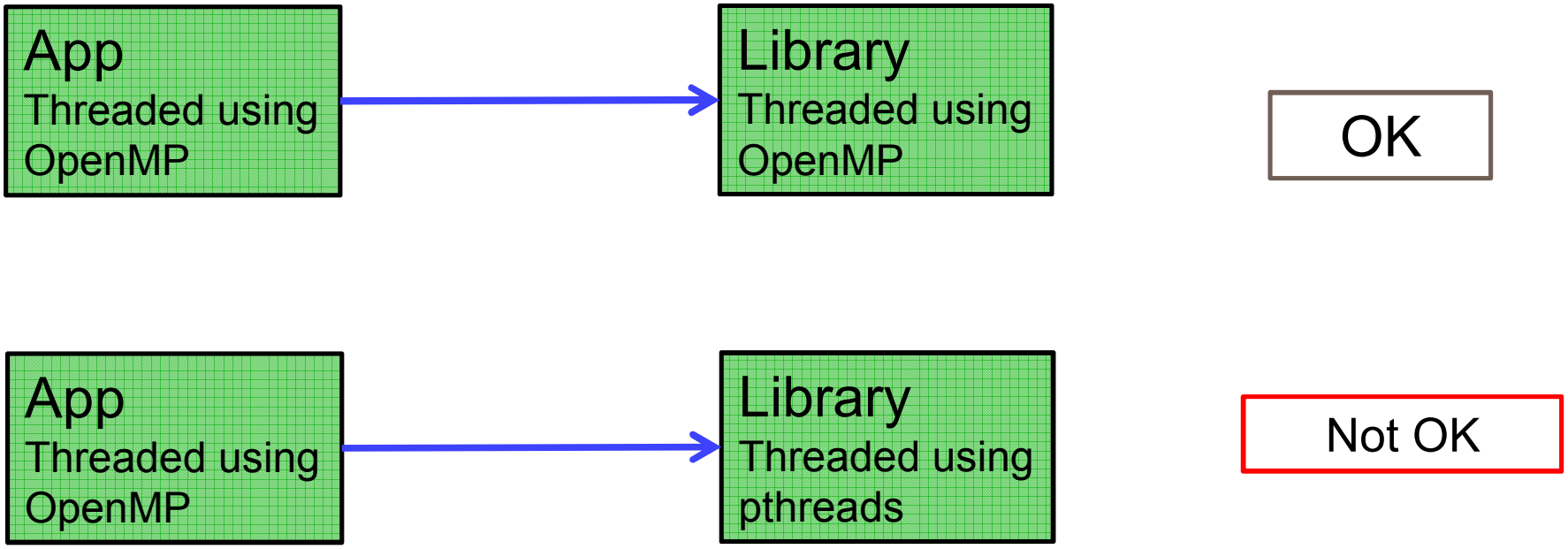
General Reality of Multicore Parallelism

Sandia National Laboratories

- Best single shared memory parallel programming environment:
  - MPI.
- But:
  - Two level parallelism (MPI+X) is generally more effective.
- But, the best option for X (if explored at all) is:
  - MPI.
- Furthermore, for an (N_1 x N_2) MPI+X decomposition:

  "For a given number of core counts, the best performance is achieved with the smallest possible N_2 for both hybrid [MPI+OpenMP] and MPI [MPI+MPI] versions. As N_2 increases, the runtime also increases."

Threading Multi-API Usage: Needs to work



- Problem: App uses all threads in one phase, library in another phase.
- Intel Sandy Bridge: Not OK **1.16 to 2.5 X slower than** OK **.**
- Intel Phi: Not OK **1.33 to 1.8 X slower than** OK **.**
- Implication:
  - Libraries must pick an API.
  - Or support many. Possible, but complicated.

Data Placement & Alignment

Sandia National Laboratories

- First Touch is not sufficient.
  - Happens at initialization. Hard to locate, control.
- Really need placement as first class concept.
  - Create object with specific layout.
  - Create objects compatible with existing object.
- Lack of support limits MPI+OpenMP.
  - OpenMP restricted to single UMA core set.

Nvidia GPUs

Sandia National Laboratories

- Supports mixed environments: All that it handles.
- Has good performance model support.
- Has flexible data placement model.
- C++ support is good, waiting for lambdas.
- Severe environment, but results in general goodness.

OpenMP 4.0, OpenACC

Sandia National Laboratories

- Active, addressing highest priority requirements.
- Incompatible, even conceptually.
- Best hope: Compiler recognizes both.

Scalable Multicore/Manycore Execution Very Challenging

- Features are coming, but slowly.
- Performance models coming too.
- Happy to be a C++ developer.
  - Fortran support always lags.
  - Fortran features arrive a decade late.
- Missing piece: Restructured application (task-centric).

New Programming Models, Environments, Languages

Sandia National Laboratories

- Can we use new:
  - Environments: Yes.
  - Models: Yes.
  - Languages: NO.*

* Other app areas may be different.

Yes: New Environments

Sandia National Laboratories

- Clang/llvm – Great new stack.
  - Enables innovations desired for decades.
  - Switch in tools: e.g., gdb to lldb, trivial.
  - Architected for new layers:
    - Minor C++ syntax extensions.
    - Additional compiler passes.
- Really, really need “flang compiler”.
  - Google search: Did you mean: *clang* compiler
  - The single biggest DOE/Science productivity requirement?
  - Support for OpenMP 4.
- Or: We should plan our (slow-but-steady) Fortran exit plan.

Yes: New Models

Sandia National Laboratories

- New models not new programming languages:
  - SIMT – GPUs.
  - SIMD – Old, but new for junior community members.
  - MAP – parallel_for
  - REDUCE – parallel_reduce.
  - Manytasking – Qthreads, HPX, even OpenMP.
  - Strategy: Introduce at node level, then expand to inter-node.
- DSLs: Yes, but embedded or mini:
  - TBB, thrust: embedded in C++.
  - CUDA: Extension, OK (at least temporarily).
  - forAllNodes – Conceptual iterator, self-documenting, polymorphic.

No: New Languages

Sandia National Laboratories

- Existing landscape:
  - C++ is our programming language.
  - C is a subset of C++.
  - Fortran is still around, and an issue.
  - Python, Perl, CMake, Matlab, etc. are great aides.
  - New scripting languages OK, if broadly available.
- New HPC languages have uniformly failed: HPF, HPCS.
- Existing HPC language (Fortran) is an emaciated entity:
  - Vendor implementations behind the standard, non-portable.
  - CUDAFortran, OpenACC for Fortran: Special projects.
  - OpenMP for Fortran is only viable parallel option.

Sandia National Laboratories

TOWARD A NEW APPLICATION ARCHITECTURE

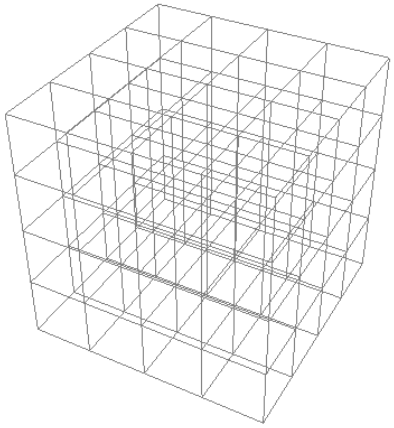
Task-centric/Dataflow:

A Productive Application Architecture

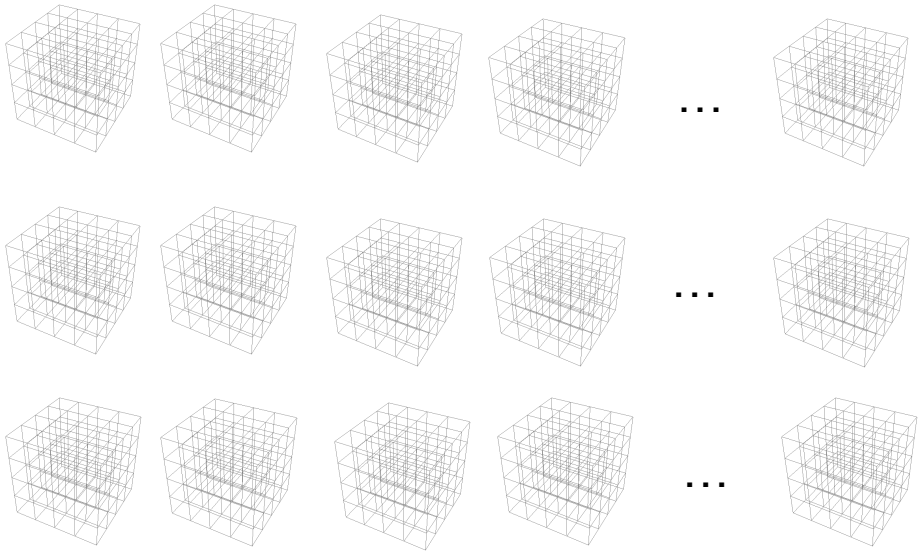
Sandia National Laboratories

- Atomic Unit: Task
  - Domain scientist writes code for a task.
  - Task execution requirements:
    - Tunable work size: Enough to efficiently use a core once scheduled.
    - Vector/SIMT capabilities.
    - Small thread-count SMP.
    - Task data dependencies.
  - Déjà vu for apps developers: Feels a lot like MPI programming.
  - Universal portability:
    - Works within node, across nodes.
    - Works across heterogeneous core types.

Task-centric/Dataflow vs. BSP/SPMD



- BSP/SPMD:
  - Halo exchange.
  - Local compute.
  - Global collective.
  - Halo exchange.

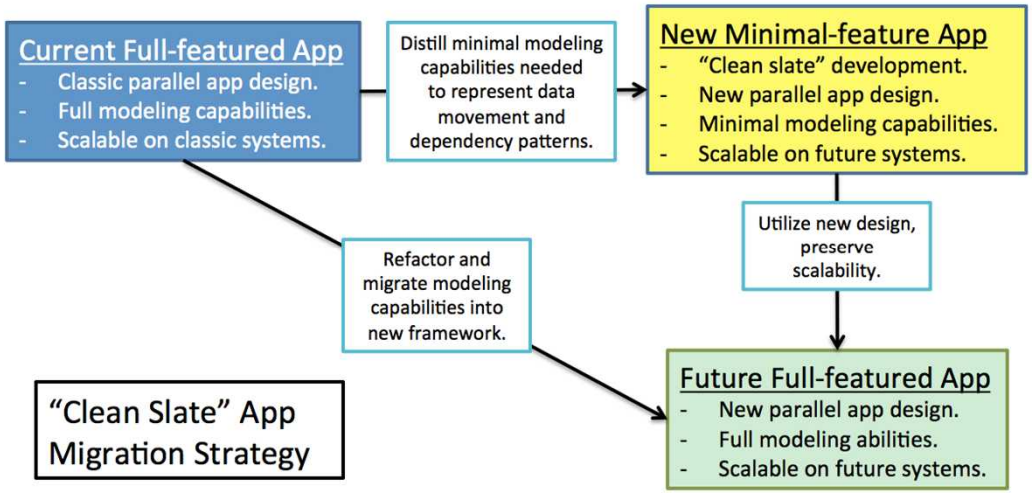


- Task-centric: Many tasks
  - Async dispatch: Many in flight.
  - Natural latency hiding.
  - Higher message injection rates.
  - Better load balancing.
  - Compatible with “classics”:
    - Fortran, C, OpenMP.
    - Used within a task.
  - Natural resilience model:
    - Every task has a parent (can regenerate).
  - Demonstrated concept:
    - Co-Design centers, PSAAP2, others.

21

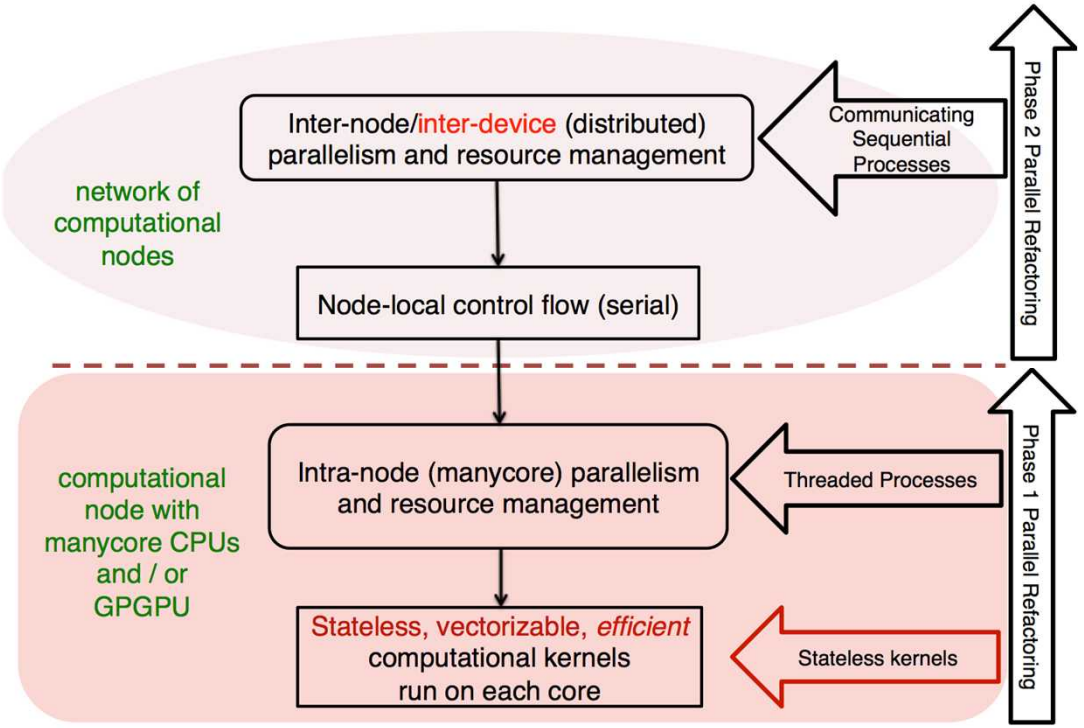
Movement to Task-centric/Dataflow is Disruptive: Use Clean-slate strategies

- Best path to task-centric/dataflow.
- Stand up new framework:
  - Minimal, *representative* functionality.
  - Make it scale.
- Mine functionality from previous app.
  - May need to refactor a bit.
  - May want to refactor substantially.
- Historical note:
  - This was the successful approach in 1990s migration from vector multiprocessors (Cray) to distributed memory clusters.
  - In-place migration approach provided early distributed memory functionality. Failed long-term scalability needs.



Phased Migration to Task-centric/Dataflow

- All Apps Looking for new Node-level programming environments.
- Exploring standards, emerging:
  - OpenMP, pthreads.
  - OpenMP 4, OpenACC.
- Exploring non-standard:
  - HPX (Parallex).
  - Legion.
- Brute force:
  - Uintah framework.
- Strategy:
  - Phase 1: On-node.
  - Phase 2: Inter-node.



23

Open Questions for Task-Centric/Dataflow Strategies

Sandia National Laboratories

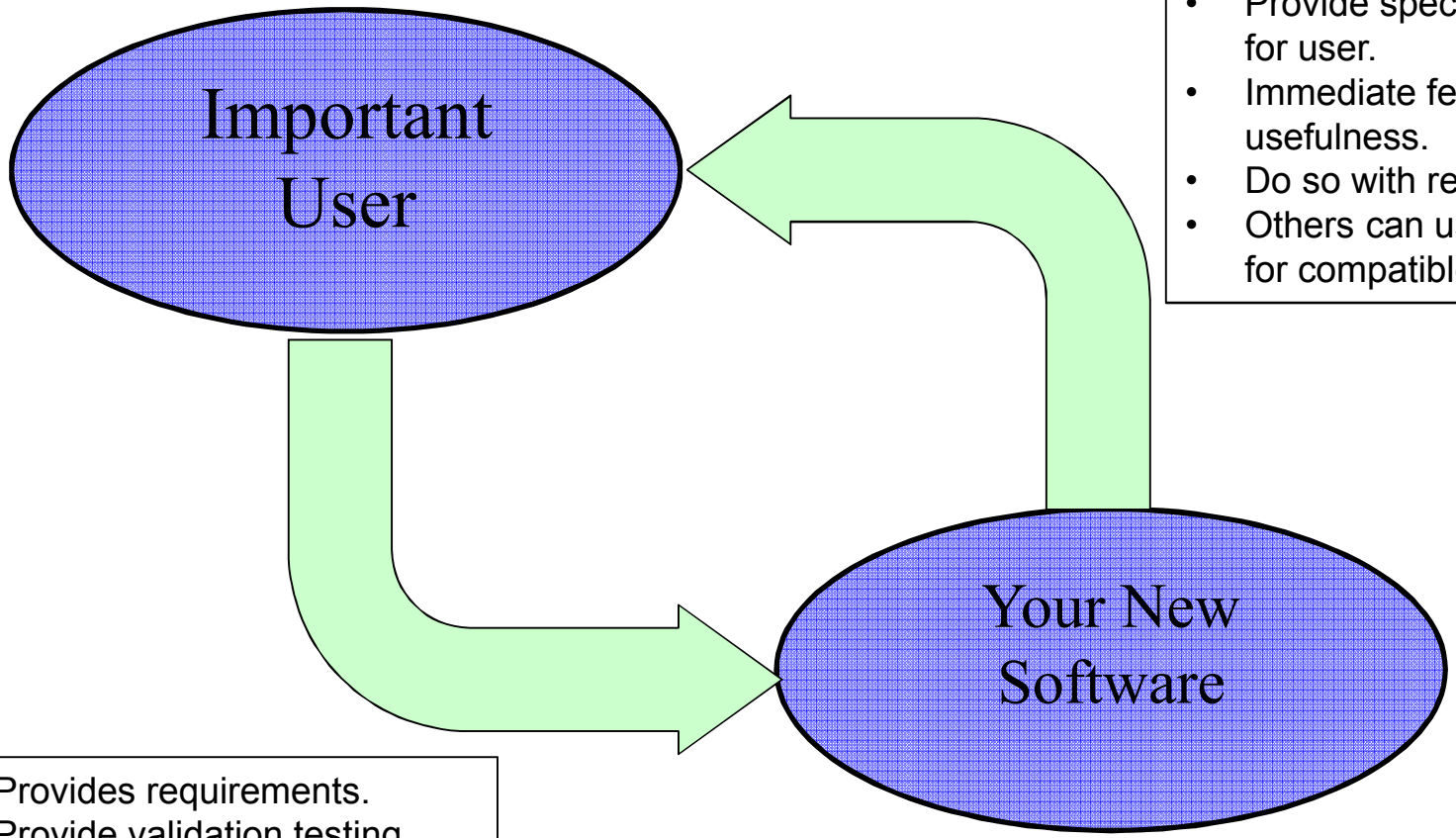
- Functional vs. Data decomposition.
  - Over-decomposition of spatial domain:
    - Clearly useful, challenging to implement.
  - Functional decomposition:
    - Easier to implement. Challenging to execute efficiently (temporal locality).
- Dependency specification mechanism.
  - How do apps specify inter-task dependencies?
  - Futures (e.g., C++, HPX), data addresses (Legion), explicit (Uintah).
- Roles & Responsibilities: App vs Libs vs Runtime vs OS.
- Interfaces between layers.
- Huge area of R&D for many years.

24

Sandia National Laboratories

SW LIFECYCLE MODELS & PRODUCTIVITY

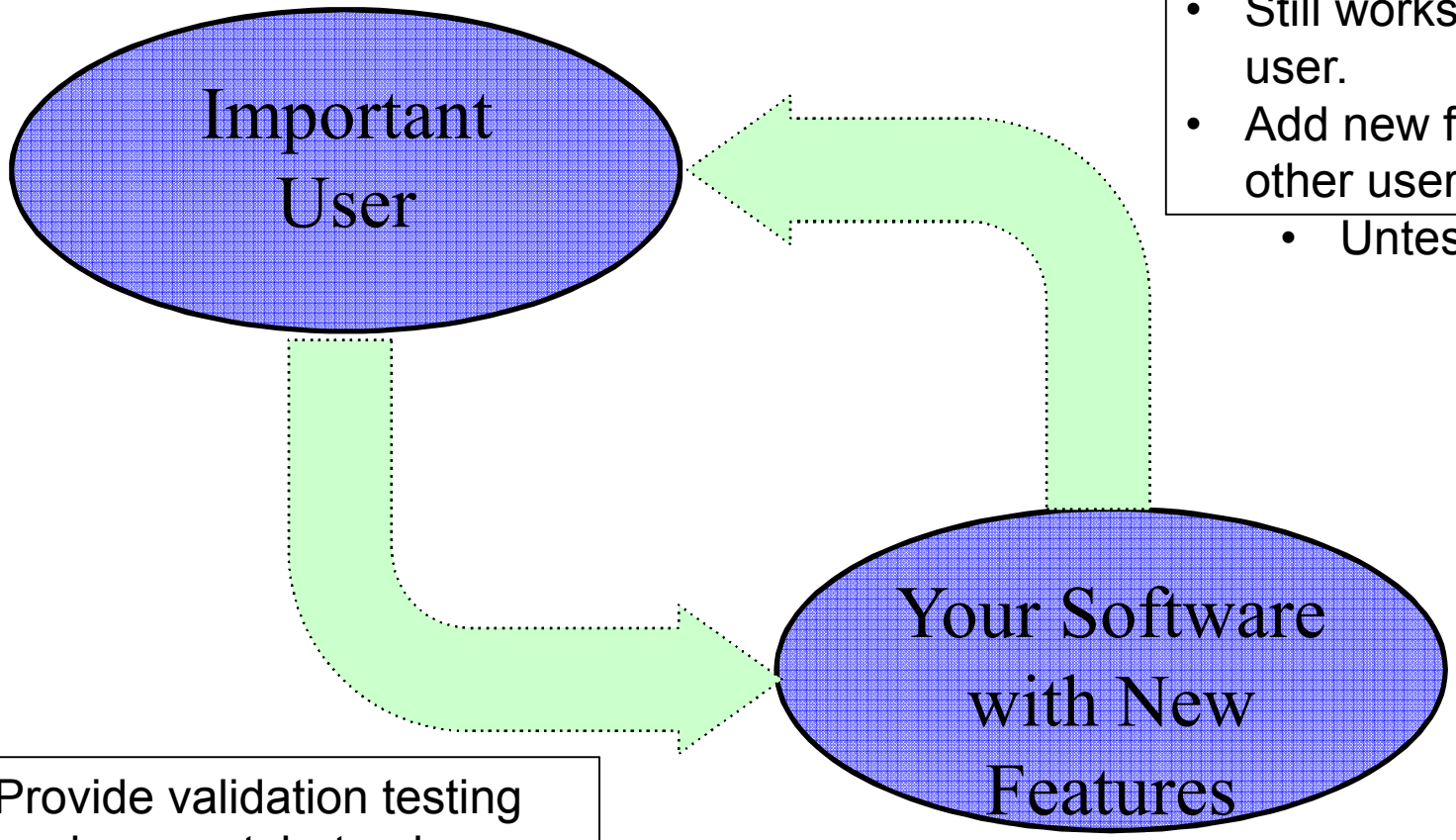
Common SW Development Scenario: Today



- Provide specific capabilities for user.
- Immediate feedback on usefulness.
- Do so with reuse in mind.
- Others can use your software for compatible needs.

- Provides requirements.
- Provide validation testing environment.
- Immediate feedback on correctness.

Common SW Development Scenario: Next Year

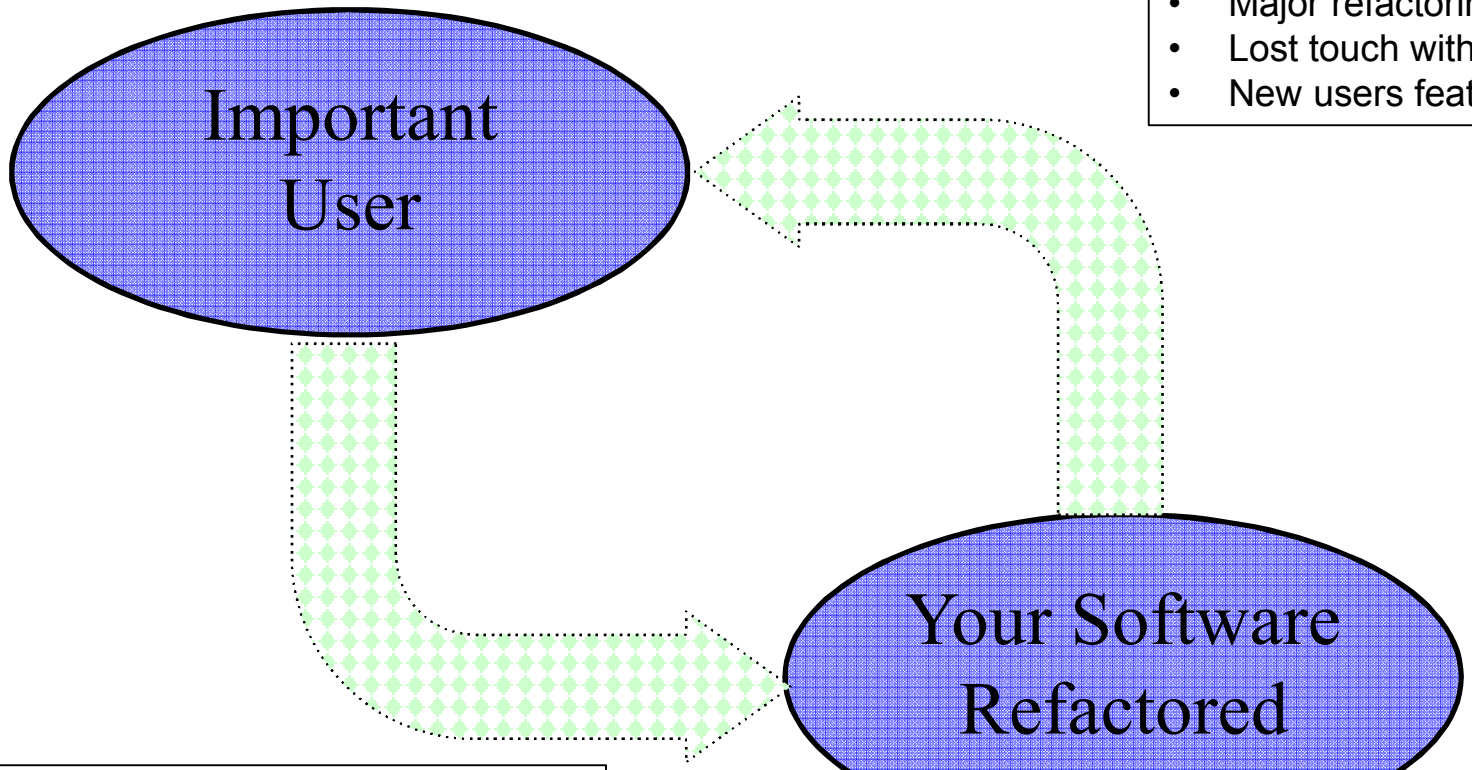


- Still works for original user.
- Add new features for other users.
  - Untested

- Provide validation testing environment, but only partial coverage.
- Other features untested.

Common SW Development Scenario: 5 Years

Sandia National Laboratories



- Major refactoring.
- Lost touch with original users.
- New users features untested.

- Use old version of code.
- Many features untested.

Result: Not enough test coverage for confident refactoring.

Validation-Centric Approach (VCA): Common Lifecycle Model for CSE Software

Sandia National Laboratories

Central elements of validation-centric approach (VCA) lifecycle model

- Develop the software by testing against real early-adopter customer applications.
- Manually verify the behavior against applications or other test cases.

Advantages of the VCA lifecycle model:

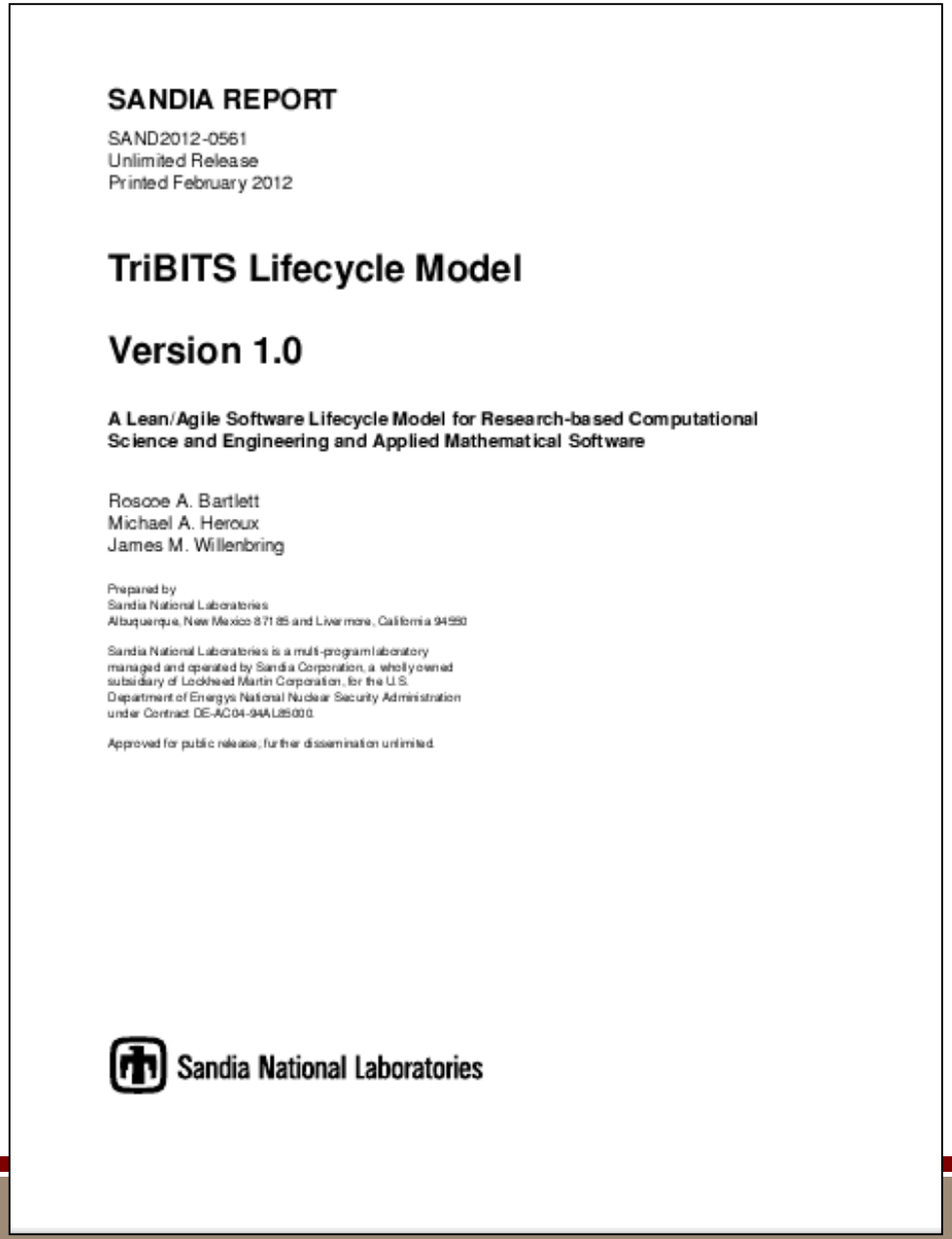
- Assuming customer validation of code is easy (i.e. linear or nonlinear algebraic equation solvers => compute the residual) …
- Can be very fast to initially create new code.
- Works for the customer's code right away.

Problems with the VCA lifecycle model:

- ***Does now work well when validation is hard*** (i.e. ODE/DAE solvers where no easy to compute global measure of error exists).
- ***Re-validating against existing customer codes is expensive or is often lost*** (i.e. the customer code becomes unavailable).
- ***Difficult and expensive to refactor***: Re-running customer validation tests is too expensive or such tests are too fragile or inflexible (e.g. binary compatibility tests).

VCA lifecycle model often leads to expensive or unmaintainable codes.

TriBITS Lifecycle Model 1.0 Document



SANDIA REPORT

SAND2012-0561
Unlimited Release
Printed February 2012

TriBITS Lifecycle Model

Version 1.0

A Lean/Agile Software Lifecycle Model for Research-based Computational Science and Engineering and Applied Mathematical Software

Roscoe A. Bartlett
Michael A. Heroux
James M. Willenbring

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Sandia National Laboratories

TriBITS: One Deliberate Approach to SE4CSE

Component-oriented SW Approach from Trilinos, CASL Projects, LifeV, …
Goal: "Self-sustaining" software

Sandia National Laboratories

TriBITS Lifecycle Maturity Levels
0: Exploratory
1: Research Stable
2: Production Growth
3: Production Maintenance
-1: Unspecified Maturity

Goals

- *Allow Exploratory Research to Remain Productive*: Minimal practices for basic research in early phases
- *Enable Reproducible Research*: Minimal software quality aspects needed for producing credible research, researchers will produce better research that will stand a better chance of being published in quality journals that require reproducible research
- *Improve Overall Development Productivity*: Focus on the right SE practices at the right times, and the right priorities for a given phase/maturity level, developers work more productively with acceptable overhead
- *Improve Production Software Quality*: Focus on foundational issues first in early-phase development, higher-quality software will be produced as other elements of software quality are added
- *Better Communicate Maturity Levels with Customers*: Clearly define maturity levels so customers and stakeholders will have the right expectations

Ultimate Goal: Produce "self-sustaining" software products.

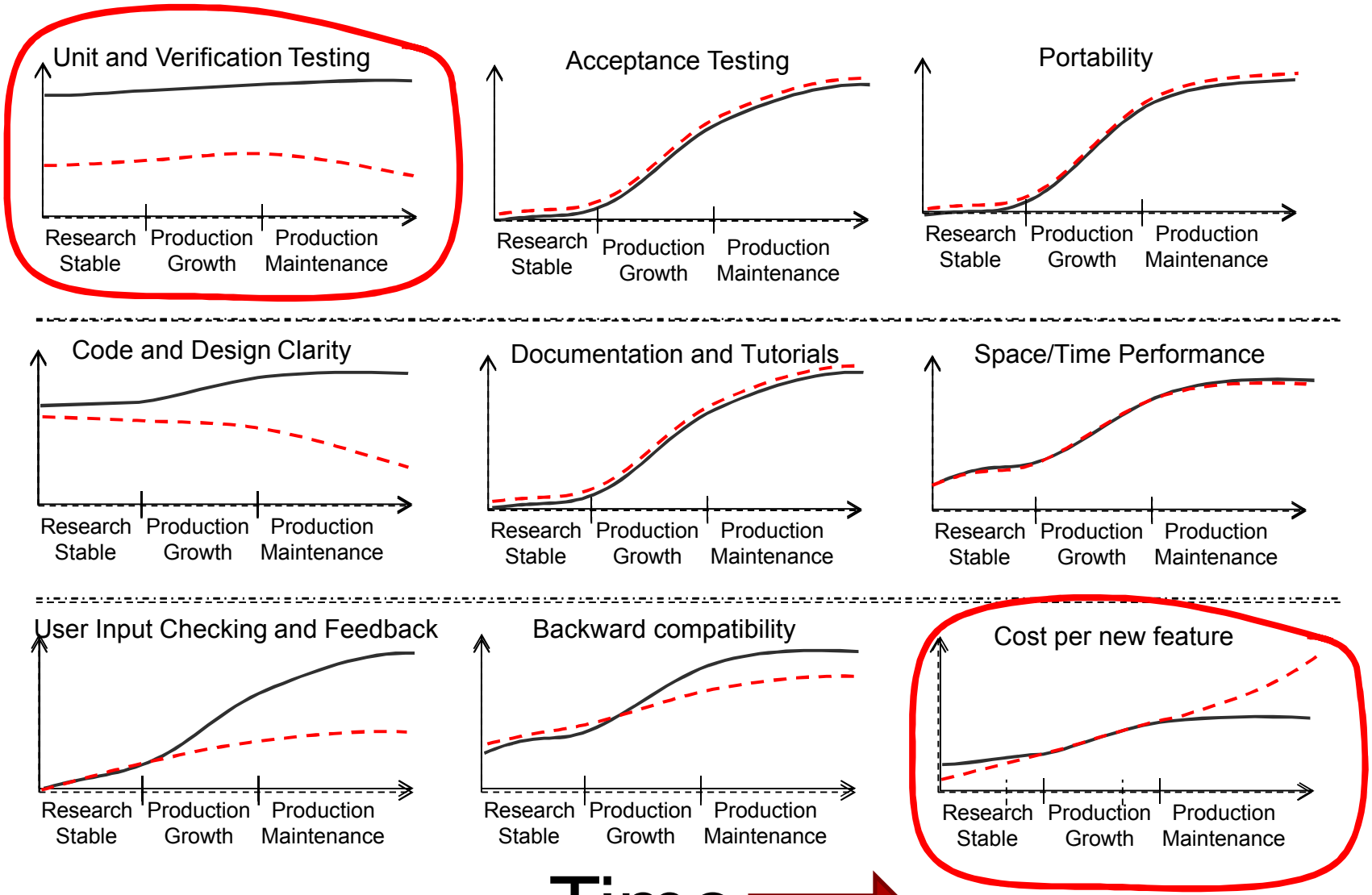
Defined: Self-Sustaining Software

Sandia National Laboratories

- ***Open-source*:** The software has a sufficiently loose open-source license allowing the source code to be arbitrarily modified and used and reused in a variety of contexts (including unrestricted usage in commercial codes).
- ***Core domain distillation document*:** The software is accompanied with a short focused high-level document describing the purpose of the software and its core domain model.
- ***Exceptionally well testing*:** The current functionality of the software and its behavior is rigorously defined and protected with strong automated unit and verification tests.
- ***Clean structure and code*:** The internal code structure and interfaces are clean and consistent.
- ***Minimal controlled internal and external dependencies*:** The software has well structured internal dependencies and minimal external upstream software dependencies and those dependencies are carefully managed.
- ***Properties apply recursively to upstream software*:** All of the dependent external upstream software are also themselves self-sustaining software.
- ***All properties are preserved under maintenance*:** All maintenance of the software preserves all of these properties of self-sustaining software (by applying Agile/Emergent Design and Continuous Refactoring and other good Lean/Agile software development practices).

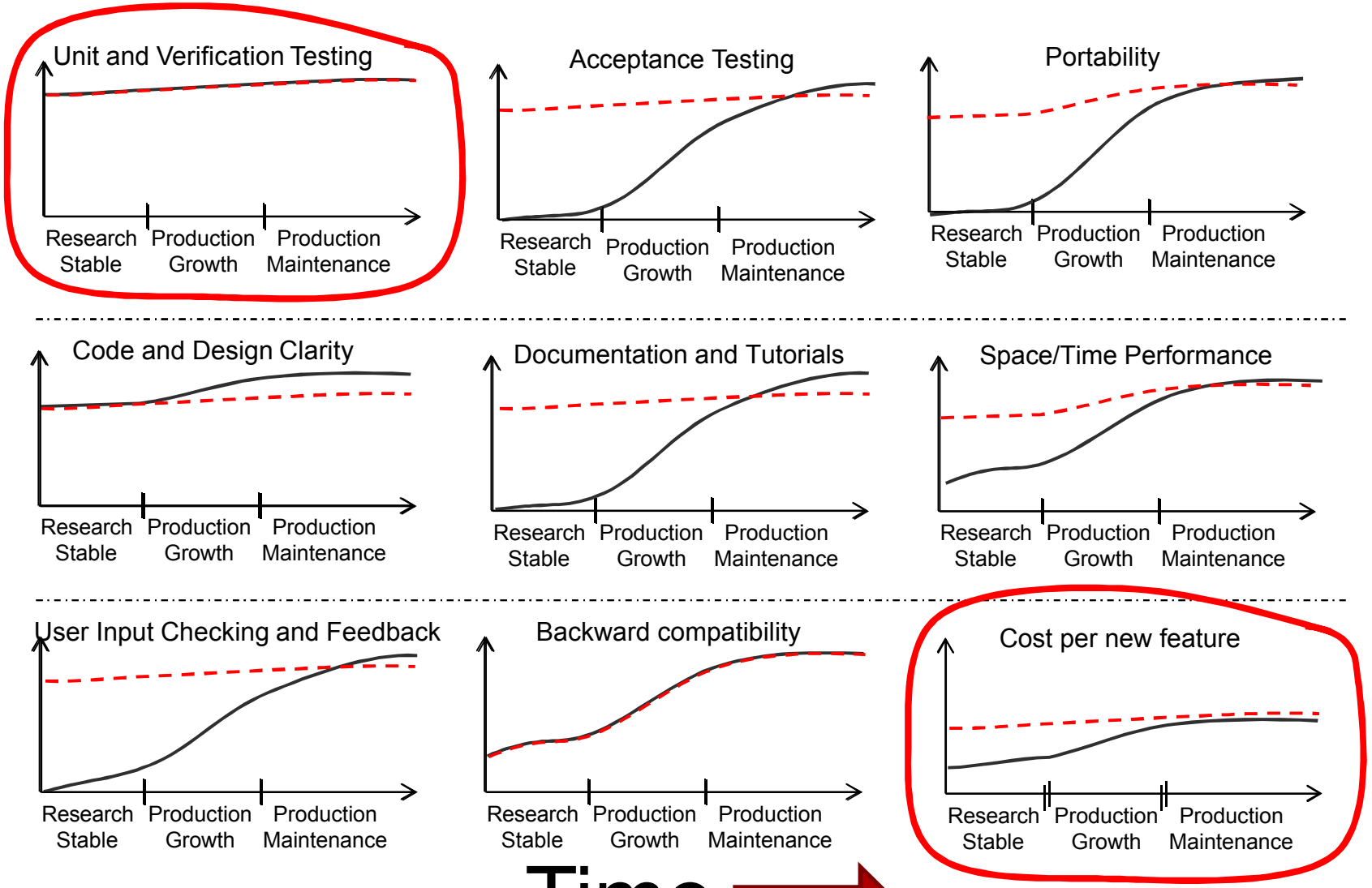
Example: Reference LAPACK Implementation

TriBITS (–) vs. Validation-Centric Approach (--)



33

TriBITS(–) vs. Pure Lean/Agile Approach (--)



34

End of Life?

Sandia National Laboratories

Long-term maintenance and end of life issues for Self-Sustaining Software:

- User community can help to maintain it (e.g., LAPACK).
- If the original development team is disbanded, users can take parts they are using and maintain it long term.
- Can stop being built and tested if not being currently used.
- However, if needed again, software can be resurrected, and continue to be maintained.

NOTE: Distributed version control using tools like Git greatly help in reducing risk and sustaining long lifetime.

Addressing existing Legacy Software

Sandia National Laboratories

- One definition of "Legacy Software": Software that is too far from away from being Self-Sustaining Software, i.e:
  - Open-source
  - Core domain distillation document
  - Exceptionally well testing
  - Clean structure and code
  - Minimal controlled internal and external dependencies
  - Properties apply recursively to upstream software

- Question: What about all the existing "Legacy" Software that we have to continue to develop and maintain? How does this lifecycle model apply to such software?

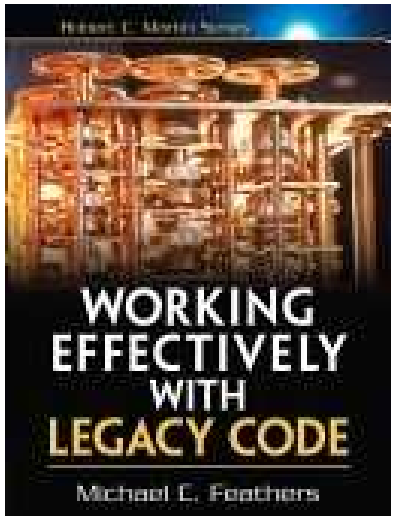
- Answer: Grandfather them into the TriBITS Lifecycle Model by applying the Legacy Software Change Algorithm.

Grandfathering of Existing Packages

Sandia National Laboratories

Agile Legacy Software Change Algorithm:

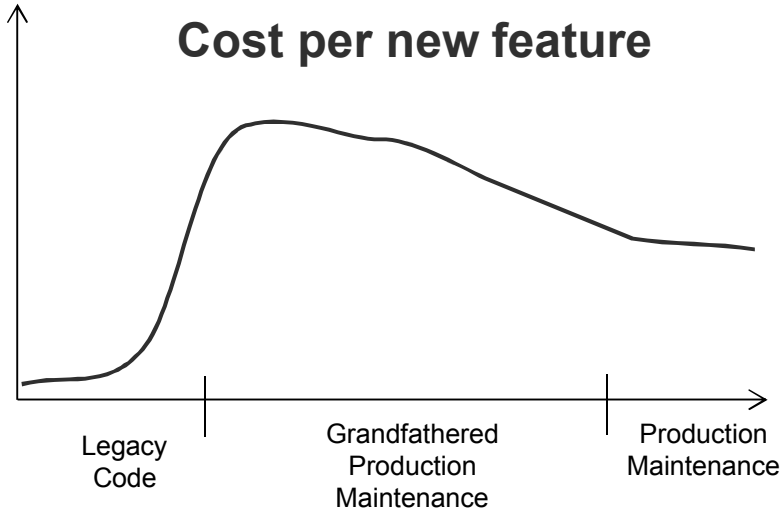
1. Identify Change Points
2. Break Dependencies
3. Cover with Unit Tests
4. Add New Functionality with Test Driven Development (TDD)
5. Refactor to removed duplication, clean up, etc.



Grandfathered Lifecycle Phases:

1. Grandfathered Research Stable (GRS) Code
2. Grandfathered Production Growth (GPG) Code
3. Grandfathered Production Maintenance (GPM) Code

NOTE: After enough iterations of the Legacy Software Change Algorithm the software may approach Self-Sustaining software and be able to remove the "Grandfathered" prefix.



Test Driven Development

Sandia National Laboratories

- Write tests first:
  - Guarantees that tests will be written.
  - Debugs the API: First attempt to use SW as intended.
- Use tests during development:
  - All tests fail at first.
  - Pass incrementally as SW written.
  - Measure of progress.
- Use tests forever more:
  - Regression.
  - Backward compatibility.
  - Aggressive refactoring.
- Single most important activity:
  - Assures long, happy life for your product.

Sandia National Laboratories

COMMODITY SOFTWARE DEVELOPMENT PLATFORMS

Day 1 of Package Life

New Package Starting in the Trilinos Project

Sandia National Laboratories

- **Git:** Each package is self-contained in Trilinos/package/ directory.
- **Bugzilla:** Each package has its own Bugzilla product.
- **Mailman:** Each Trilinos package, including Trilinos itself, has four mail lists:
  - package-checkins@software.sandia.gov
    - CVS commit emails. "Finger on the pulse" list.
  - package-developers@software.sandia.gov
    - Mailing list for developers.
  - package-users@software.sandia.gov
    - Issues for package users.
  - package-announce@software.sandia.gov
    - Releases and other announcements specific to the package.
- **New_package** (optional): Customizable boilerplate for
  - CMake/Doxygen/Python/TestHarness/Website

Day 1 of Package Life on GitHub & Atlassian

Sandia National Laboratories

- **Private Git -> Public Git:** Each package is self-contained, clonable.
- **Bugzilla -> Issue tracking:** Integrated, Issue listings, Milestones & labels, Commit keywords. Pull requests.
- **Mail lists -> Forums:** More diverse, interactive.
- **New_package** -> Stubbed out project.

- Atlassian: Even more advanced but costly.
  - Dev Tools: Git, mercurial, other tools.
  - Jira: Issue tracking.
  - Confluence: Collaboration, communication.

SW Development Platforms are Commodity

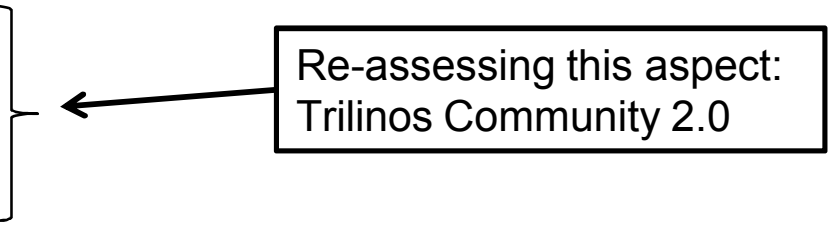
Sandia National Laboratories

- Projects can start from Day 1 with high-quality environment, global visibility.
- Need to re-state the Trilinos value propositions from first principles.

Trilinos Community Membership Value Proposition

Sandia National Laboratories

- Established Tools, Processes.
  - Huge boost in getting started.
  - Challenge when toolsets change.
- High quality, complementary components.
  - Ready-made interactions.
  - Insulation from architecture details.
- Improved Personal Experience.
  - Personal tool & process choices defined, common.
- Improved Team Experience.
  - Ready-made collaboration tools & processors.
  - Checklists.



Trilinos Community 2.0

Sandia National Laboratories

- GitHub, Atlassian:
  - Open source SW development, tools platforms.
  - Workflows for high-quality community SW product development.
- Trilinos value proposition:
  - Included these same things, but must re-evaluate.
  - Must address packages that want GitHub presence.
  - Must (IMO) move Trilinos itself to GitHub.
- New types of Trilinos packages (evolving):
  - Internal: Available only with Trilinos (traditional definition).
  - Exported: Developed in Trilinos repository, available externally.
  - Imported: Developed outside of Trilinos, available internally.

Trilinos Community 2.0

Sandia National Laboratories

- Case studies:
  - TriBITS: Was an internal package, now external.
  - DTK: Has always been external.
  - KokkosCore: Is internal. Needs to be available externally.
- Issues to Resolve:
  - Package inclusion policies: Define for each package type.
  - Quality criteria: Contract between Trilinos and packages.
  - Workflows: Development, testing, documentation, etc.
  - Trilinos on GitHub: Evaluate.
  - Trilinos Value Proposition: Re-articulate Trilinos Strategic Goals implications.

Sandia National Laboratories

IDEAS: A NEW DOE PRODUCTIVITY-FOCUSED PROJECT

Institutional Leads (Pictured)
Full Team List

IDEAS productivity

Sandia National Laboratories

Science Use Cases

J. David Moulton

Tim Scheibe

Carl Steefel

Glenn Hammond

Reed Maxwell

Scott Painter

Ethan Coon

Xiaofan Yang

Project Leads

ASCR: M. Heroux and L.C. McInnes

BER: J. D. Moulton

Extreme-Scale Scientific Software Development Kit (xSDK)

Mike Heroux

Ulrike Meier Yang

Jed Brown

Irina Demeshko

Kirsten Kleese van Dam

Sherry Li

Daniel Osei-Kuffuor

Vijay Mahadevan

Barry Smith

Hans Johansen

Lois Curfman McInnes

Ross Bartlett

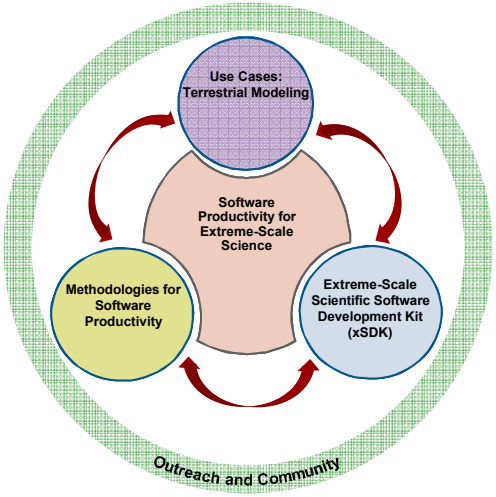
Todd Gamblin*

Andy Salinger*

Jason Sarich

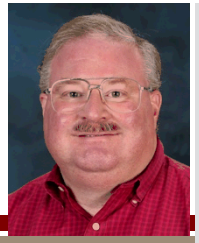
Jim Willenbring

Pat McCormick



Methodologies for Software Productivity

Outreach



David Bernholdt

Katie Antypas*

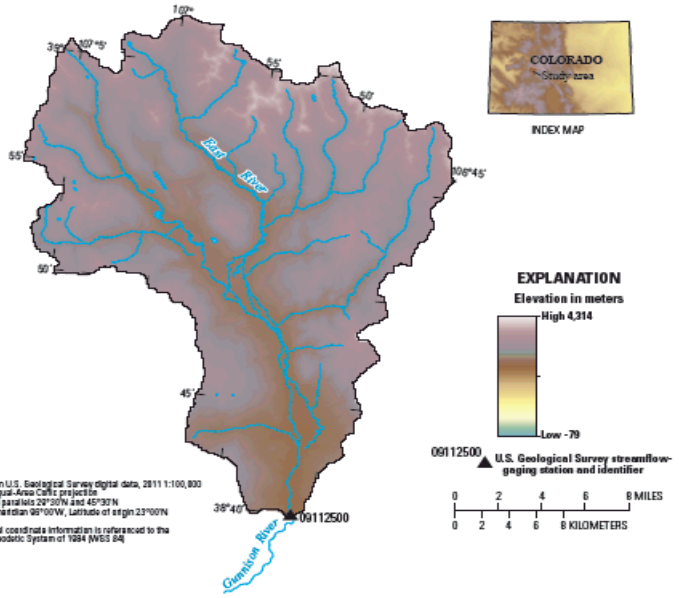
Lisa Childers*

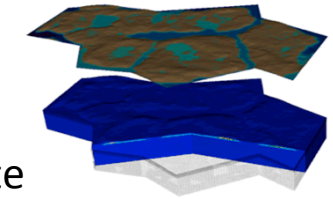
Judith Hill*

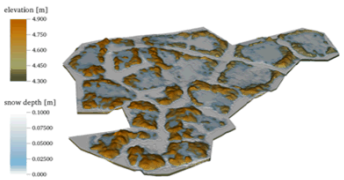
*Liaison

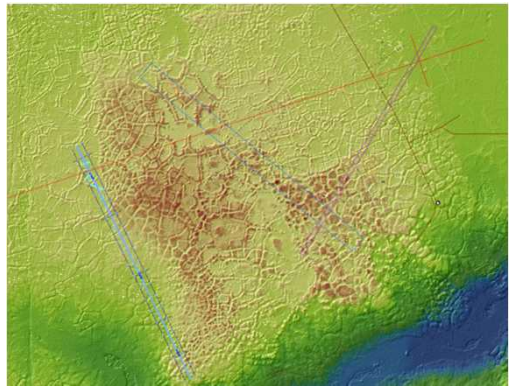
Use Cases: Multiscale, Multiphysics Representation of Watershed Dynamics

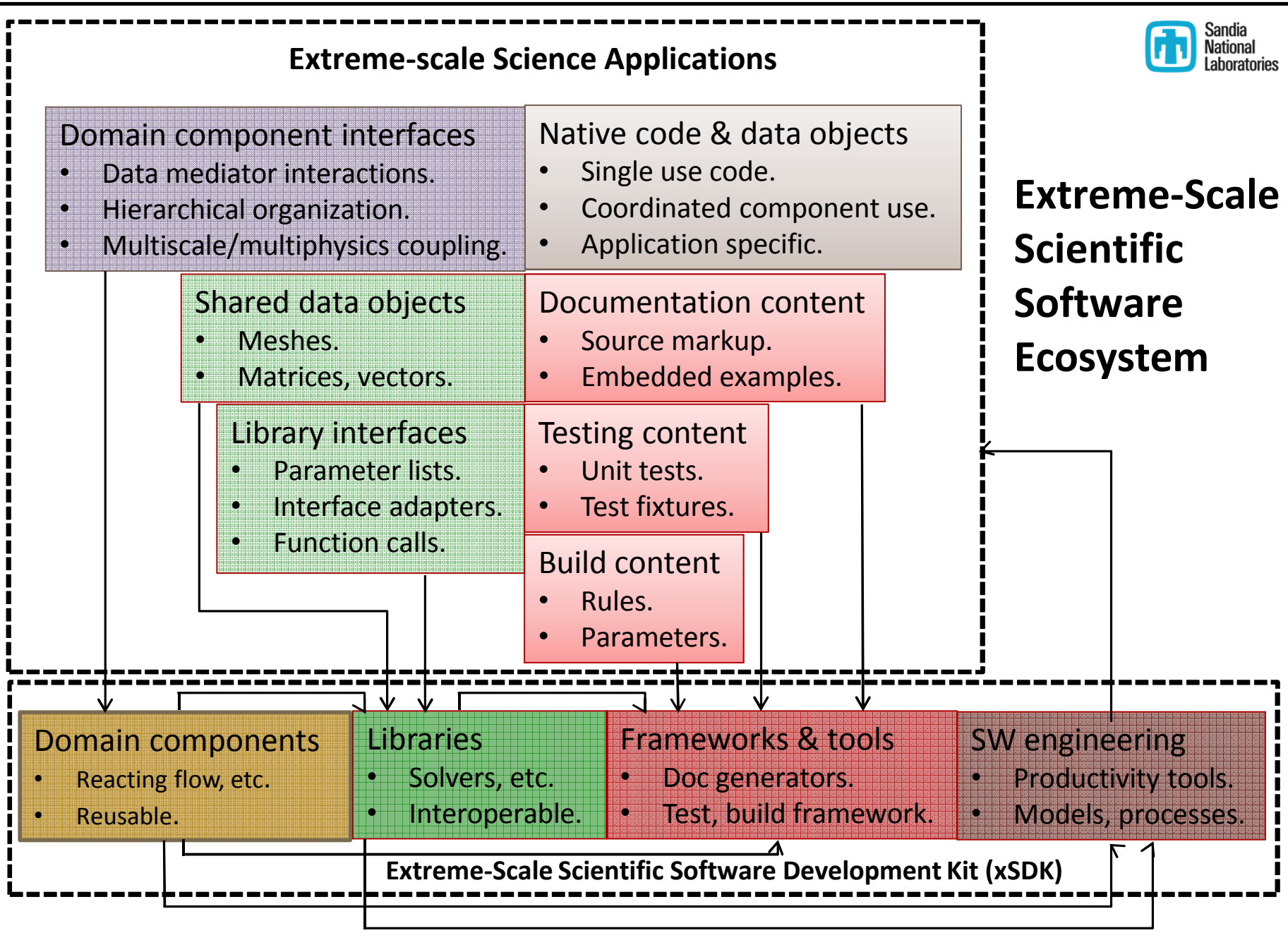
- **Use Case 1:** Hydrological and biogeochemical cycling in the Colorado River System.
- **Use Case 2:** Thermal hydrology and carbon cycling in tundra at the Barrow Environmental Observatory.
- **Leverage and complement existing SBR and TES programs:**
  - LBNL and PNNL SFAs.
  - NGEE Arctic and Tropics.
- **General approach:**
  - Leverage existing open source application codes.
  - Improve software development practices.
  - Targeted refactoring of interfaces, data structures, and key components to facilitate interoperability.
  - Modernize management of multiphysics integration and multiscale coupling.



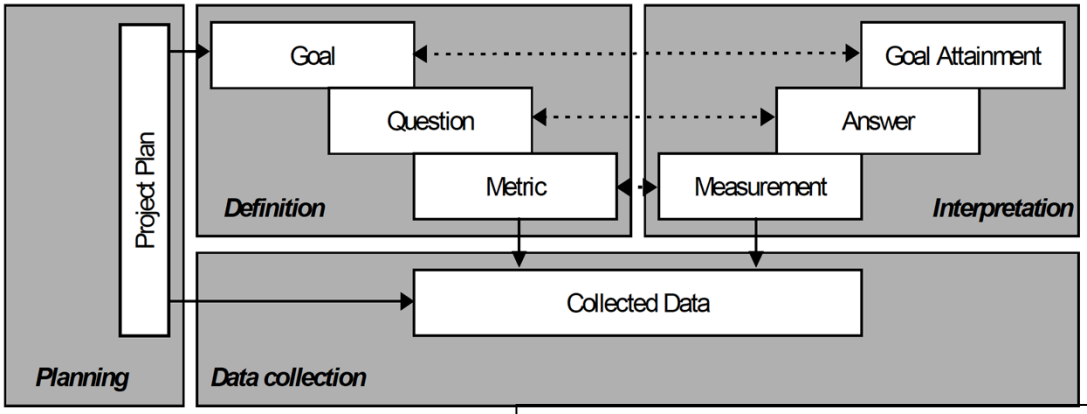







Extreme-scale Science Applications
Domain component interfaces
• Data mediator interactions.
• Hierarchical organization.
• Multiscale/multiphysics coupling.
Native code & data objects
• Single use code.
• Coordinated component use.
• Application specific.
Shared data objects
• Meshes.
• Matrices, vectors.
Documentation content
• Source markup.
• Embedded examples.
Library interfaces
• Parameter lists.
• Interface adapters.
• Function calls.
Testing content
• Unit tests.
• Test fixtures.
Build content
• Rules.
• Parameters.
Extreme-Scale Scientific Software Ecosystem
Domain components
• Reacting flow, etc.
• Reusable.
Libraries
• Solvers, etc.
• Interoperable.
Frameworks & tools
• Doc generators.
• Test, build framework.
SW engineering
• Productivity tools.
• Models, processes.
Extreme-Scale Scientific Software Development Kit (xSDK)

Methodologies: SW Productivity Metrics



Source: *The GQM Method: A Practical Gu*
for Quality Improvement of SW Developme
Solingen and Berghout.

- Define *processes* to define metrics.
  - Starting point: Goals, questions, metrics (GQM).
    - Define goals, ID questions to answer, define progress metrics.
- GQM Example:
  - Goal: xSDK Interoperability.
  - Question: Can IDEAS xSDK components & libs link?
  - Metric: Number of namespace collisions.
- Cultivate effective use of metrics:
  - Use metrics to drive and track use case progress.
- Promote use of metrics via Outreach.

Software Engineering and HPC: Efficiency vs Other Quality Metrics

Sandia National Laboratories

How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Source:
Code Complete
Steve McConnell

Helps it ↑

Hurts it ↓

Summary

Sandia National Laboratories

- Disruptions:
  - Disruptive architecture changes force disruptive software refactoring.
  - Capabilities Drive Ability to Couple physics and scales, need for modularity.
- A Productivity Focus is promising:
  - Walking back to first principles, iterating forward.
  - Provides guidance in time of disruptive changes.
- Programming:
  - Current SMP environments are not adequate (OMP, etc.)
  - New programming models, environments are OK, but no new languages.
  - Manytasking: Shows promise as “universal approach” to future app design.
    - Enables use of classic languages and environment.
    - Requires significant runtime system, API investments.
- Community Changes:
  - SW Lifecycles: Explicit models bring rigor, foster communication.
  - Commodity development platforms: Enable world-wide, high-quality collaborative development.
  - DOE focus on Productivity: IDEAS Project, goal of an Extreme-scale Scientific SW Ecosystem.