Applying the ASCI Experience

Doug Kothe

ORNL

David Nowak

ANL

Alex Larzelere

Exagrid Engineering

Outline

- **The ASCI Program**
  - **Perspectives on what worked and why**
- **ASCI code project case study**
  - **The Telluride Project at LANL**
  - **Lessons learned**

The ASCI Program

- **Had a compelling goal**
  - **Driven by a problem**
    - **July 3, 1993 – President Clinton**
      - **Continued Moratorium on Full Scale Nuclear Testing**
      - **Seek a Comprehensive Test Ban Treaty**
      - **Directed DOE to: "establish a stewardship program to ensure the preservation of the core intellectual and technical competencies of the U.S in nuclear weapons."**
  - **Needing a solution**
    - **Develop simulation tools to attack the problem**
    - **Involving three NNSA labs & the supercomputing industry**
- **Had an evangelist (Gil) running it**
  - **Who made you feel like dropping down & giving him 20 upon hearing him**
  - **Who scared you (me) to death if you had to speak in front of him**
- **Had a culture of never say never**
  - **Challenging problems with high payoff – this attracts smart people**
- **Kept the dialogue on "what" and not "how"**
  - **The evangelist just killed the bear (we labbies skinned it)**
- **Was the envy of all other simulation efforts**
  - **We had the time and resources to pull it off**

Critical to Have the Right Startup

- **Define the problem (s)**
- **Define measurable goals**
  - **Deliver a comprehensive set of computational capabilities**
    - **Applications**
    - **Computing Platforms**
    - **User Environments**
    - **Partnerships**
- **Find leaders**
  - **Who help to overcome challenges and "attitudes"**
- **Build team**
  - **industry, labs, academia, international**

Challenges to Overcome

It's all a matter of attitude …

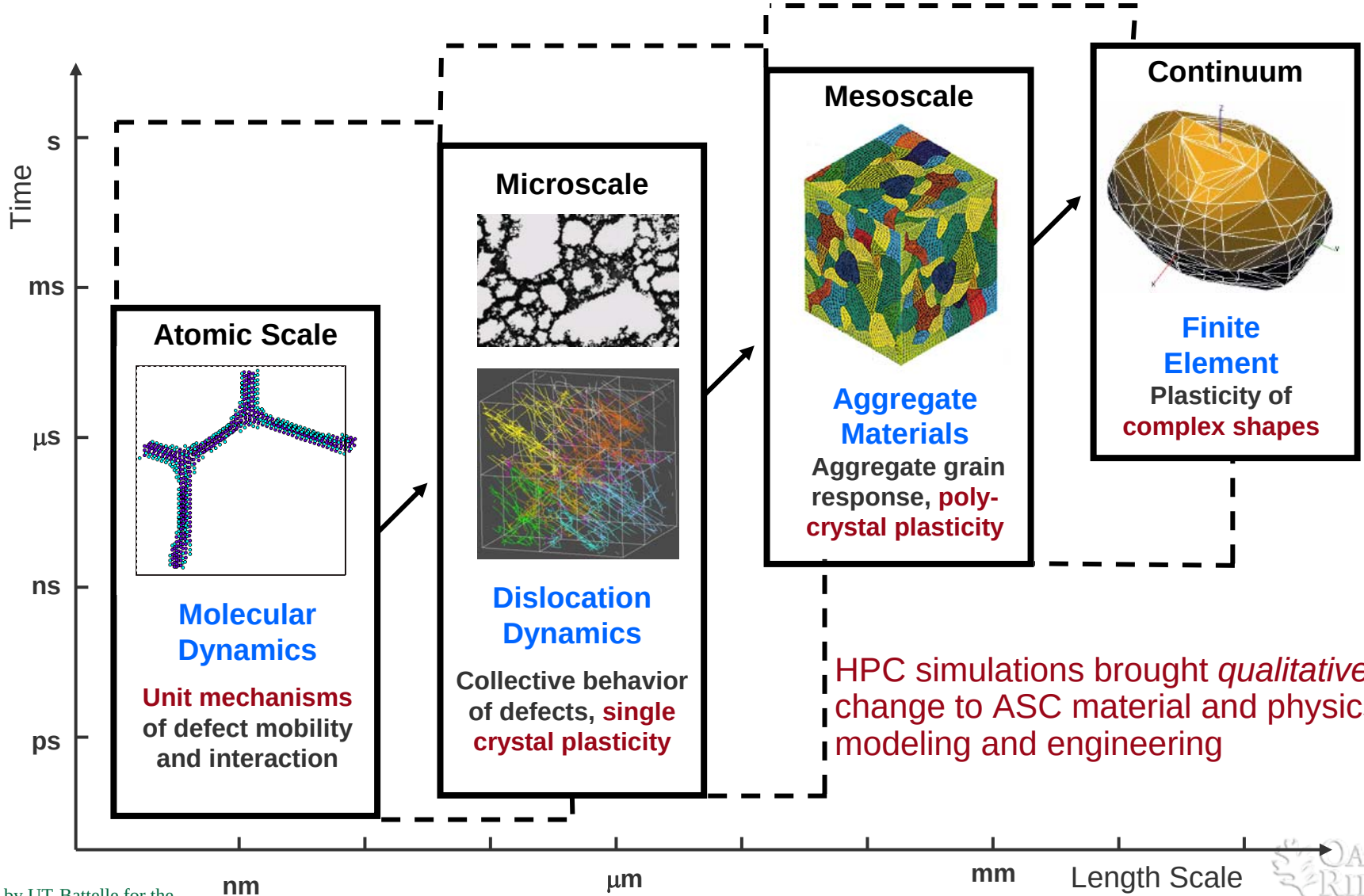
- **Current example: nuclear energy and GNEP**
  - "Let's not rush into things, there's plenty of time …"
    - ***Perhaps, but waiting (or insufficiently funding R&D) is a sure-fire way of***
      - **Guaranteeing mediocrity**
      - **Making sure the United States is technically disadvantaged vis-à-vis foreign technology leaders and ensuring that there is no U.S. industry to compete for the ongoing worldwide nuclear revival**
  - "It's all too expensive …"
    - ***Compared to what?***
    - ***Do we know enough to predict costs reliably out to 2050, especially given the likely transformational effects of computing, technological advances, … ?***
- **The ASCI Program was met with similar attitudes**

The DOE has demonstrated expertise in developing large-scale, fully coupled science-based codes

- **DOE NNSA is a success story in neutronics, thermal-hydraulics, thermal- mechanics, structural mechanics and coupled codes**
  - **The ASCI Program is an existence proof**
- **DOE has dominated the Top 500 list of the world's most powerful computers for years**
- **DOE SC is building a similar success story with its fundamental science codes**
  - **Clear compelling problems, measurable goals**
- **Energy security is a compelling requirements driver**
  - **We need to better articulate the compelling goals**

The ASC Challenge

Predictive Simulation with Quantified Margins and Uncertainties



HPC simulations brought *qualitative* change to ASC material and physics modeling and engineering

Case Study: Nuclear Energy (GNEP)

Legacy Simulation Technology is Many Decades Old

- **Almost all codes run only run on single processors**
- **Almost none of the codes have coupled physics**
  - **The coupling is done by hand**
- **The extremely limited memory requirements required the extensive use of phenomenological models**
- **Most of the codes are restricted to a single rector type**
  - **Light water, fast re-cycling reactor**
- **However, they do work –**
- **The price**
  - **Size, weight, efficiency, innovation, time to license, and, most importantly --- cost**
- **Sounds very familiar to the pre-ASCI days**
  - **Except the state-of-the-art is now much more advanced**
    - **The gap is much larger!**
  - **Fully coupled physics, single fully integrated code, utilizing tens of thousands processors**
  - **Up to a factor of 100,000 in computing capability**

Case Study: GNEP
Remember What it Takes to Get There

- **Adopt a modern science and simulation-based engineering approach**
  - Nuclear engineering must transition to a modern science and computations-based discipline
    - High fidelity (science-based) integrated simulations must form the core of the design efforts and allow for rapid prototyping
    - Science-based, validated modeling at both the detailed (small-scale) and systems-level must be part of the core capabilities
    - The field must generate internal technical excitement to attract the "best and the brightest"
- **But to get there requires sustained funding for an integrated program**
  - **Leverage other programs, but don't count on entire elements: Applications, platforms, user environments, partnerships**
  - **Such a program will attract the best people (ASCI did!)**
- **We know we can do this**
  - **It's been done before, we have the expertise, we have the people, and lengthy planning exercises are not needed**

Outline

- **The ASCI Program**
  - **Perspectives on what worked and why**
- **ASCI code project case study**
  - **The Telluride Project at LANL**
  - **Lessons learned**

Telluride Project Mission

- **Deliver verified and validated computational manufacturing tools for key DOE Complex operations**
- **Capture, within these computational manufacturing tools:**
  - **Realistic macroscopic models for free surface incompressible fluid flow, phase change heat transfer, material microstructure evolution, thermo-mechanical solid response, and electromagnetic effects**
  - **Robust, high-fidelity numerical algorithms for discrete finite volume solutions of nonlinear systems of PDEs on 3D, unstructured-mesh computational domains**
  - **Modular, extensible, maintainable, and portable software constructed with quality-assured software engineering practices**
- **Realize efficient execution on computational platforms ranging from desktop systems to high performance computing systems typical of the DOE ASCI Program**
- **Target casting and welding operations within the DOE Complex**

Lots of Cool Physics and Algorithms

Physics

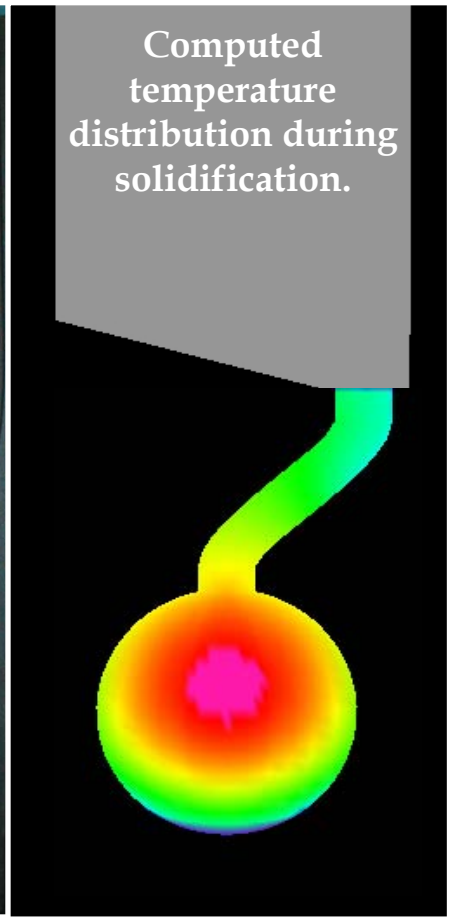
- Heat transfer
  - Conductive, convective, inductive
  - Radiative on boundaries
- Phase change
  - Liquid/solid, solid/solid transitions
  - Pure materials, alloys having n species
  - Arbitrarily-complex phase diagrams
- Incompressible fluid flow
  - Boussinesq approximation for thermal and solutal buoyancy
  - Turbulence models for fill
  - Porous media models in mush
- Interface kinematics and dynamics
  - Free surfaces, fluid/solid and fluid/fluid interfaces
  - Surface tension and phase change forces
- Thermo-mechanical material response
  - Residual stress and distortion
  - Elastic/plastic, creep, phase change
- Quasi-static MHD
  - Inductive (Joule) RF heating
  - Magnetic (Lorentz) stirring
- Material microstructure
  - Nucleation and growth models
  - Homogenization models

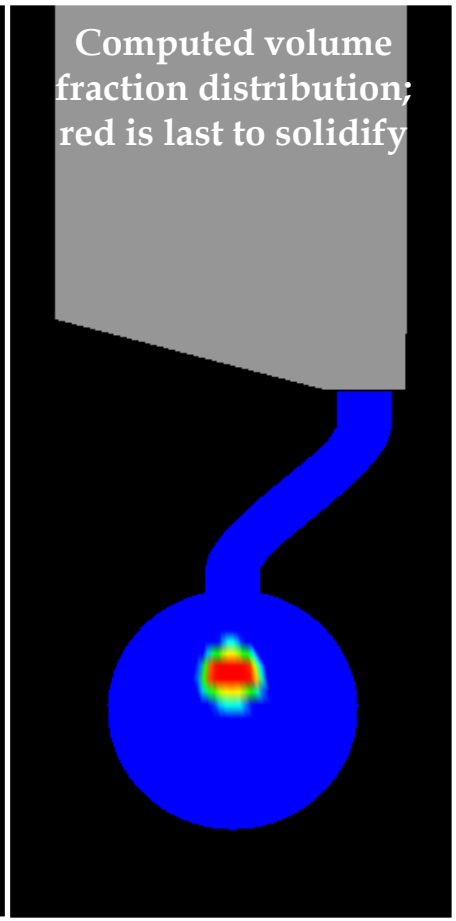
Algorithms

- Unstructured mesh domain partitioning
  - Each cell is a logical cube mapped into degenerate hexahedra: hexes, tets, prisms, pyramids
- Cell-centered collocated finite volume discretizations
  - Nearest neighbor connectivity; formally 2nd order in space
  - Support (mimetic) operators for function reconstruction and some matrix operations
- Phase change heat transfer
  - Implicit, nonlinearly-consistent NK-based enthalpy method
  - Locally nonlinear systems for alloy solidification
- Flow: semi-implicit, fractional step projection method
  - Design constraint: complex topology interfacial flows with density ratios of 1 to infinity (void) across interface
- Solid Mechanics: node-based CV-FEM
- E&M: edge, face-based staggering
- Multi-level-preconditioned Krylov-based linear solution methods

Necessary to attract good people

Aluminum Ball Casting





A ball is a difficult shape to cast since the last place to solidify will tend to be toward the center of the ball and will typically exhibit shrinkage porosity as shown.

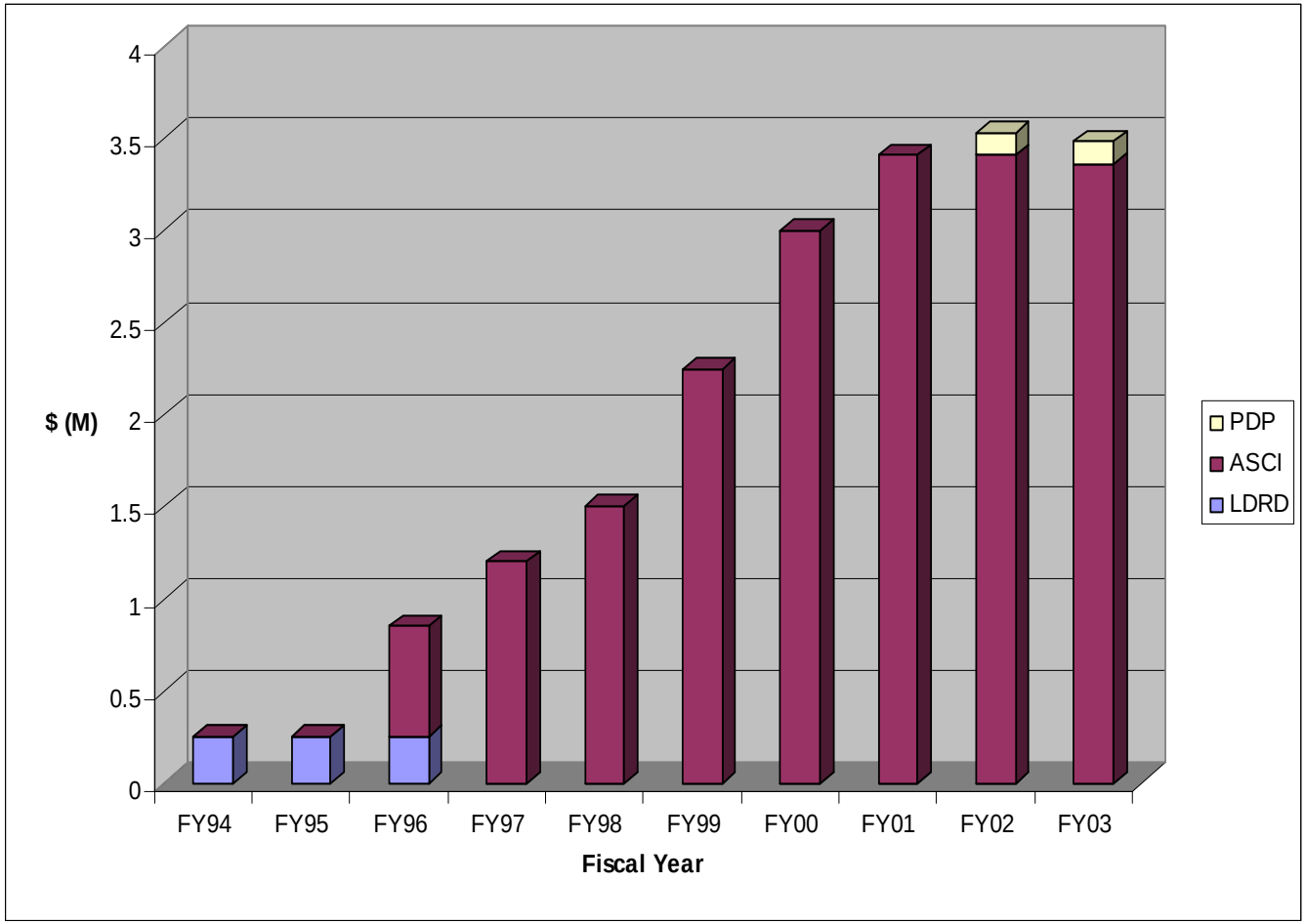
Aluminum is a challenging metal to cast as it has 7% shrinkage on solidification.

Basic phase change heat transfer simulations really do make a difference! If ICs and BCs are known, the last point to solidify is an attainable simulation result.

Telluride Project Overview

- **Principal software product: the "Truchas" code**
  - **~300K lines (including packages) of source code, primarily Fortran 90/95**
  - **Written completely from scratch (rare at LANL)**
  - **Designed for parallelism; runs/restarts on any combination of parallel/serial platforms**
  - **MPI interface library in C, glue scripts in Python, Perl, shell**
  - **Memory hog; dominated by indirect addressing; heavily implicit; poor single-processor performance**
- **Project team dynamics**
  - **Project Leader was able to hand select and recruit them**
  - **Expected turnover (no one has been on project since inception)**
  - **Lots of formal training in software engineering practices & project management**
  - **Only a few card-carrying CS people involved**
  - **Support numerous postdocs, some graduate students, even a HS student**
- **Underwent formal DOE/NNSA external SQA audits in 1999, 2002, 2004**
- **Heavily reviewed**
  - **3 formal external reviews**
  - **Quarterly internal reviews**
  - **Annual program reviews**

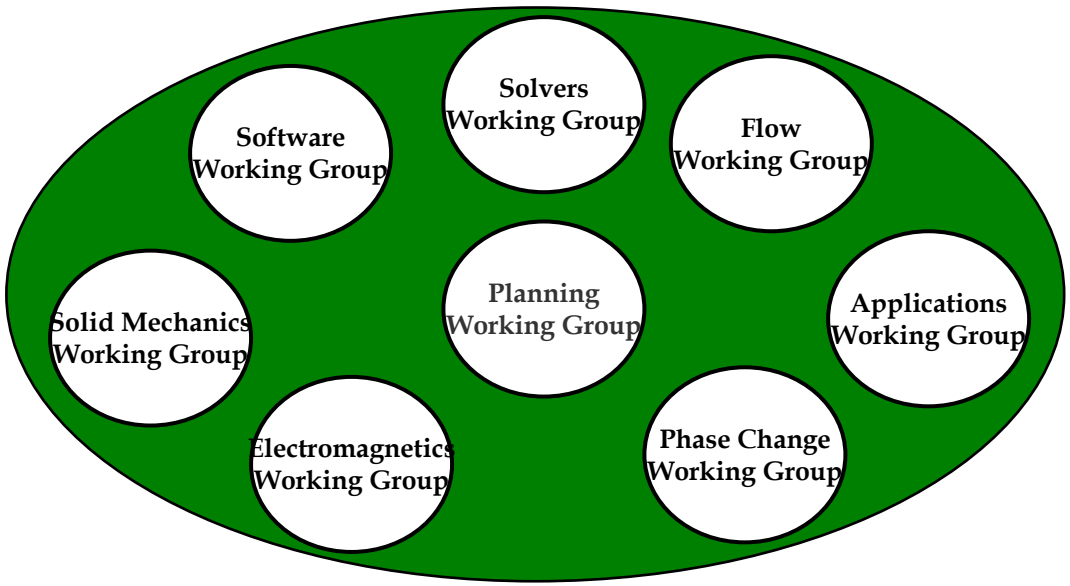
Telluride Project Budget History: 1994-2003



Consistent with bringing up a large, multi-phase application code from scratch. Investment in the >$50M range over 11 years. But we thought it would take half as much time and money (because we were smart and better)

"Working Groups" Were Effective in Managing the Telluride Project

- **Working Groups (WG) improved project productivity**
  - **Established to increase communication and to promote delegation of responsibility, accountability, ownership, & decision making**
  - **Project lead is no longer the bottleneck**
- **Each WG**
  - **is chaired by a subject matter expert**
  - **has 2-5 "card-carrying" member who meet bi-weekly for working and planning sessions**
  - **is responsible for planning and executing its portion of the project**
- **WGs foster leadership, ownership, pride**
- **WGs provide multiple conduits into and out of the project**



This is a scalable process!

Software Engineering Lessons Learned

- **Design your software for and implement unit testing**
  - **Can you "make test" in every dir? Ex: buggy pageant**
- **Be aware of the impact of your choice of data structures – you have to live with it**
- **Use levelized design (defined interfaces, data hiding) as a mantra**
- **All your tests should be tied to requirements**
- **Conduct code and design reviews!**
- **Have formal releases early and often**
- **Do not under estimate the large difference between research prototype and production. Resist the urge to view your prototype as production. Embrace throwing away prototypes.**
- **Pair programming is good thing and really works.**
- **Assessments are a big pain, but they can help to force culture change (for better or worse)**
- **Get on as many platforms as possible – it makes your software better**

Software Engineering Lessons Learned

- **It doesn't work to worry about performance later**
- **Unified Build Theory can consume all people cycles; similarly for a testing harness**
- **Training new team members takes time! Must plan for and institutionalize this**
- **Frameworks (backplanes, environments, etc.) should evolve**
- **If you don't test it, it will break ("bit rot")**
- **Even for large projects, really only see 3-5 active "committers"**
- **Don't just design tests based on physics/algorithms; think about all use cases!**
- **Establish and document your software process**
- **Commit hot spots are usually symptomatic of a bigger problem**
- **Spending most of your time on testing is not a bad thing**
- **Writing code is very personal: allow personalities to flourish; don't be too rigid**

Project Management Lessons Learned

- **Establish requirements first (Telluride ex: induction heating, residual stress first)**
- **Have a testing lead**
- **Fund your competition**
- **Adopt "alpha users" as part of your project**
- **Invite and encourage formal peer reviews**
- **Do not grow too big and too soon (big projects reach a point of diminishing returns)**
- **Do not over-promise. Set expectations early and often to clients and customers.**
- **Software culture change is possible – it must be top down**
- **Seek help from any and every body. But be careful who you bring on board formally**

Project Management Lessons Learned

- **Have a good project charter that allows you to say NO – scope creep is just too irresistible in science**
- **The Project Leader can actually be a hindrance rather than a help; he/she should not be on the critical path**
- **Milestones can be good**
- **Documentation is not easy or fun, but it's necessary. You may not find a great product that allows multiple authors simultaneously working**
- **If you have never done this before, consider the "pi factor" on estimates of time/resources**
- **Embrace process but not without scrutiny and thought. It must be tailored for your project**
- **High risk, exploratory research can co-exist within an applied software project. But nourish it**
- **Record software statistics so informed estimations can be made for time/resources required for future activities**

Takeaway: Lessons Learned

- **Leadership**
  - **PL must have vision, drive, and be a "generalist", or knowledge of the physics, algorithms, and software issues**
- **Vision**
  - **Have clear simulation targets to guide all decisions and prevent scope creep; targets must be "game changers"**
- **Endurance**
  - **Projects need multi-year (10+ years) support**
- **Milestones**
  - **Focus and direct the project if the milestones are "SMART" (specific, measurable, attainable, relevant, and timely)**
- **Partnership**
  - **No project can do it all – must reach out to industry, academia, and user community all along the way**