



# Changes in CEPTRE Development Practices

**Jennifer Liscum-Powell**

**Shawn D. Pautz**

**Simulation Research Technology Department  
Sandia National Laboratories**

**Presented at Texas A&M Labfest  
(Workshop VI on Parallel Transport )  
May 2005**

SAND 2005-2833P

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,  
for the United States Department of Energy under contract DE-AC04-94AL85000.





# Outline

- Introduction
- Need for Change
- Principles Driving Changes in CEPTRE Code Development
- The Changes
- Benefits
- Issues



# CEPTRE Features

---

- Multigroup energy discretization
- Discrete ordinates angular discretization
- Arbitrary order of anisotropic scattering
- Unstructured-mesh Galerkin finite elements
- Second-order forms of the transport equation
  - **Self-Adjoint Angular Flux (SAAF)**
  - **Even-Odd Parity Flux (EOPF)**
- Parallel implementation with spatial domain decomposition
- Object-oriented program design with C++
- Integrated into an architectural framework (Nevada)
- Build on parallel Krylov solver libraries (Trilinos, AztecOO)
- Simultaneous space-direction solve



# History of CEPTRE

---

- Started as research prototype codes exploring the second-order formulation of the transport equation
- Software quality practices?
- ASCI review panel wanted us to move into framework
  - Choice of Nevada or Sierra
- Inherit software quality practices from the framework
  - Recent SQE audit we were given NO credit for framework and SQE practices



# Need For Changes

---


- Slow code development
- Difficulty porting code
- Differences in development philosophy
- Competing Needs
  - Releases of Alegra vs Ceptre needs
  - Definition of failure causes Ceptre development to grind to halt
- Frequently breaking code
  - Updates ...nothing works



# Principles Driving Changes

---

- Autonomy of Practices
- Attention to Relationships/Levelization
- Compare with Industry Practices
- Portability
- Maintainability
- Configuration Management
- Testing Philosophy
- Documentation
- Training



# Autonomy of Practices

---

- Code dependence does not mean identical development practices are needed
- Sandia's SQE policy assumes teams are autonomous
  - No credit given for framework SQE
- Tailor development practices to application specific needs



# Attention to Relationships/Levelization

---

- Structure desirable in terms of both source code and the organization of people
- Multiple classes, multiple files implies relationships that should be recognized
- Different teams have different contexts
- Critical to examine dependency levels



# Compare with Industry Practices

---

- Don't reinvent the wheel
- Make use of industry tools
- Make use of industry ideas
- Smart people exist outside the labs – we can learn from them too
- Lots of people using the same tools leads to more robust tools
- We have unique problems, but not all of our problems are unique



# Portability

---

- Portable source code – use language standards
- Portable build systems

*Built on all major [ASCI] platforms and equivalent systems without ANY modifications or additions to the [Nevada] source distribution..*



# Maintainability

---

- Readable code
- Standard idioms
- Minimal preprocessor directives
- Avoid very large components
- Minimize definitional dependencies
- Attention to levelization



# Configuration Management

---

- Use version control system
- Identification of versions (tagging, branches)
- Promotion models
- Mix and match versions
- Tension exists between the needs for active development and stable, robust code



# Testing Philosophy

---

- Unit tests
  - Test driven development
- Regression tests
- Different metrics for different purposes
  - Failure IS an option
  - Commit testing, stable testing, porting testing, and release testing
- Testing as a tool for learning



# Documentation

---

- Source code through comments
- Doxygen
- Users manual
- Physics manual



# Training

---

- Improve skills
- Formal and informal
- Code reviews
- Books and articles



# Changes from Nevada's Approach

---

- Separate repository
- “Autoconfiscation”
- External libraries
- Separate regression system
- Smart versioning
- Change Control Board (CCB)
- Training



# Separate Repository

---

- Previously application code (e.g. Ceptre) and framework code (Nevada) contained in same checkout
- Ceptre's active development line exists in separate repository
- New development merged into Nevada repository in timely manner



# “Autoconfiscation”

---

- Nevada uses sntools build system
- Development version of Ceptre uses Autoconf/Automake
- Enables a configure, make, install process
- Enables unit tests
- Enables levelized builds and partial/incremental builds



# External Libraries

---

- Nevada's build system requires all external libraries to reside in one location (the TPL directory)
- Separate build system allows us to use other (non-TPL) libraries
- Allows use of TPL versions not provided by Nevada (new or old)
- Making increasing use of boost



# Separate regression system

---

- Nevada has a regression system (testAlegra) which runs ALL applications
- Ceptre's new independent regression system test only Ceptre executables
- Multiple test suites can be defined
- Multiple options employed in building code
- Highly automated
- Failure IS an option



# Smart versioning

---

- In Nevada's development model the HEAD is king
- Now treat Nevada, the TPL's and Ceptre as separate entities in terms of versioning
  - Tagging
  - Mix and match versions
- Platform dependent versions for development
- Promotion model



# Change Control Board (CCB)

---

- Nevada developers follow practices at their discretion
- Within Ceptre development we have created a CCB to manage departures from practices
- CCB manages changes to library versions
- CCB manages recovery from abnormal events



# Training

---

- Use code reviews not just for code quality but as a teaching tool
- Actively identify good books and articles and share these ideas
  - Addison-Wesley's C++ In-Depth Series
  - C/C++ Users Journal
- Participation in Sandia SQE training and application of lessons learned



# Benefits

- Greater control of development cycle
  - Accelerated development
  - Less thrashing
- Greater confidence in code
- Greater understanding of code
- Personal improvement of team members
  - General knowledge (C++, testing)
  - Particular knowledge of project
- Internalization of API changes
- Greater feedback to framework
- Morale (ownership)



# Additional Issues and Strategies

- Certification of Ceptre on other platforms (OS and compilers)
- Additional Ceptre-owned repository for Nevada?
- “Autoconfiscation” of TPLs and Nevada
- New development independent of Nevada components
- Position Ceptre for other applications
  - Satellites
  - NuGet
  - Z-modeling