CCSM Software Engineering Coordination Plan

June 18, 2002
Draft

This document is an update to a coordination plan proposal dated Oct. 25, 2001. That document proposed a coordination plan for overall CCSM software engineering. A number of issues are addressed, ranging from tools and resources to planning and priority. The first section provides some background on the formation of this document. Section two outlines improvements made most recently. Section three outlines a plan of specific changes that are recommended. Section four defines roles. Section five is a summary of recommendations. There are also appendices that describe a number of software coordination issues and an appendix that defines terms.

1 Background

CCSM is a climate model that has grown significantly over the past several years. Early on, there were a handful of scientists and software engineers working on the project and the project was reasonable to manage. In the past few years, the number of developers and collaborators has increased significantly as has the importance of this work.

Over the past year, concerns have been initiated by NCAR scientists, software engineers, the community, and external collaborators about how CCSM can continue to develop effectively as the number of people working on the project increases. Some of these concerns are summarized in Appendix A which came out of a discussion at the CCSM programmers meeting on September 19, 2001. Currently, there is a loosely defined set of rules that governs model development. Independent research and loose coordination has been the status quo. This no longer works well for CCSM.

2 Recent Work

Over the course of the last year, a number of actions have been taken. In particular,

- Weekly discussions of McConnell’s, Software Project Survival Guide.
- Weekly discussions of Robertson and Robertson, Mastering the Requirements Process.
- Construx two-day training course titled, Software Project Survival Guide, followed by a one-day consulting session. A list of tasks was identified to understand and improve software process at NCAR. These are included in appendix B.
- Presentations to CCSM scientists, at the SEWG meetings, and to the SSC with respect to CCSM software engineering process improvement.
- Establishment of an atmospheric model change review board. This board has started to meet and is slowly spinning up.
• Implementation of a formal CVS repository request process.
• Implementation of a CCSM CVS policy.
• Initiation of a project to update the CCSM GUI/scripts using formal software process. This project is currently in the requirements collection phase.
• Hiring of a software engineer to focus primarily on infrastructure support and tools.
• Hiring of a software engineer to focus primarily on testing and performance issues.

All of these items have had a positive impact on CCSM software engineering capability and quality. In the following section, follow-ons to these items will be addressed.

3 Future Plans

As part of an effort to continually improve CCSM software process, the following sections describe a proposal of steps to implement that, if done in a coordinated way, will improve CCSM software quality and control. To a large degree, these policies are general and should be applied to all CCSM components as well as to the CCSM model as a whole. It is recognized that individual components have different needs and concerns, and that slight variations in policy may be required for each component. In addition, there may be distinctions required between scientific development and software engineering development. However, there shouldn’t be any distinctions between internal and external developers.

There is recognition that CCSM is fundamentally a research project and that individuality is an important part of the way people work. There also must be recognition that CCSM is a project that has a mandate to release a stable, high-quality software product to the community. This plan attempts to balance the needs of the individual developer with the needs of the overall community. Hopefully the developer will not feel significant changes in the way they work, but the release product quality will be higher. Many of the new processes implement additional controls between what is considered development code versus what is considered production code.

The recommendations in this document strive to improve the software quality of CCSM and make people in the project more productive, which will help achieve the ultimate CCSM goal, to do better science.

3.1 Change Review Board (CRB)

As the current time, an atmosphere CRB has been formed. This group has been meeting at irregular intervals mainly to address CCSM/CAM release issues. Some time has been spent outlining a CRB process, but this is still not well defined. It is recommended that the atmosphere CRB meet regularly; first, to outline a CRB process and role, and second, to carry out this process. It is recommended that the CRB process include:

• formal reviews of repository activity.
• formal reviews of the status of release and development versions with regard to both scientific and software testing.
• formal reviews of individual changes to the primary development version.
• formal reviews of proposed changes to the primary development version.
• formal review of tasks.
• formal review of priorities and the assignment of individuals to take responsibility for individual tasks.
• some level of planning.
• the ability to accept, reject, or defer code changes.
• a procedure for merging changes onto the development branch and testing them.
• a documentation process.
• a procedure that outlines how developers interact with the CRB.
• a procedure that outlines how to get feedback from the community on the priorities of change requests.
• a procedure for implementing simple bug fixes quickly that bypasses some of the CRB formal process.

These items are very important to assess the impact of changes, understand the potential for hidden bugs and unintended interactions, and coordinate and prioritize the work.

At the same time, a formal CRB should be formed to address CCSM issues. In particular, there is currently lack of coordination with regard to cross component interactions. The CCSM scientists and programmers meetings have served as an arena to discuss cross component interactions, but there is currently no responsibility on any person or group to coordinate these activities.

It is recommended that CRBs be created for all CCSM component model groups. However, before doing so, the benefit and cost of forming additional CRBs for the ocean, ice, land, and coupler components should be considered.

3.2 Repository

Over the past several months, a CVS policy and access request procedure has been put in place, largely formulated out of this document dated Oct 25, 2001. As this process moves forward, the policy and request procedure should be updated when necessary to address individual concerns. There are still open issues related to security in the CVS framework and tools should be built or local modifications to CVS made to ensure CVS is being used properly and that individuals are following policy. The CSEG group is currently investigating the use of CVS import/export as a way to instantiate multiple copies of the CCSM repository as a means of providing enhanced security. This work should continue. The CSEG group is also looking into improving the CVS access request and status capabilities mainly by implementing a web based form and database tool. This work should continue.

CCSM should continue to use CVS as a tool for source code control. CVS is free and CCSM developers have become experienced and comfortable with it. Evaluating, choosing and spinning up a new source control tool would require a commitment of at least one person full time for several months for tasks like evaluation, testing, and training. The risks and cost are significant in terms of potential resources required, potential financial cost, and/or potential loss of productivity during the transition period. In the background, CCSM should look into the availability of other tools in conjunction with outside collaborators. There should be an ongoing community effort to evaluate new (free and commercial) software tools, so if a new tool becomes available and recommended, CCSM could migrate to it.

3.3 Testing

An individual has been tasked with developing, running, and maintaining testing software for CCSM on a full-time basis, largely as a result of this document dated Oct 25, 2001. The CSEG CCSM test engineer is developing test procedures and tools to validate both release and development versions of CCSM. Ultimately, the goal is to have a suite of tests that are used by the community to accomplish testing tasks, such as weekly
tests of released software on all validated machines, daily “build and smoke” tests of software under development, and regression testing of source code changes. These testing capabilities must be made available to the community as soon as possible in order to increase the quality and decrease the risk associated with many developers working on the models simultaneously.

3.4 Status Accounting

Status accounting is the ability to understand what changes have been implemented, what changes are proposed, what defects have been discovered, who is working on these, and the status of each. Over the last 6 months, a ccsm-bugs group has been added to the CGD wreq system and is being used to track CCSM bugs. While this is not ideal, it is a significant improvement over previous actions.

In the future, a better software tool to document the CCSM status, bug tracking, and testing status needs to be found. There are several free, publicly available tools for status accounting and tracking, including GNATS and Jitterbug. Someone should be given responsibility and time to explore and implement one of these tools for use in CCSM. The accounting tool could be an integral part of the change review boards and vice versa.

3.5 Documentation

Documents are important for good communication as well as project tracking. Documents include things like task lists, meeting notes and web pages. There are several documents that need to be developed or improved. Some, like the CVS policy statement, planning documents, and a status accounting document have been discussed above and implemented. The CCSM developers guide, which also needs to be kept current and more complete, serves as the guide for all CCSM development. An individual should be put in charge of coordinating documents, including the developers guide, to assure they remain up to date. It is also recommended that a review of the CCSM public and local web pages be undertaken in an attempt to clarify and improve document visibility.

3.6 Planning and Prioritization

Because of the size and importance of CCSM, CCSM development needs to become better focused on stability and quality than it is now. Better planning is required not only for CCSM science and software engineering, but for infrastructure support as well. CCSM needs to improve short-term, long-term, and strategic planning, and a process should be put in place to address this task. There is little short- or long-term planning now, and what’s done is not formally recorded. CCSM software engineering strategic planning does occur now to some degree. There is a “CCSM Software Engineering Plan 2000-2005” that was prepared largely by the software engineering working group that addresses many strategic planning issues. This document is currently under review and is being revised. Part of the planning process on all time scales should include improved cost/risk/benefit analysis to set priorities to make best use of all resources.

Some of the planning issues will be solved if change review boards are created. However, each component and the CCSM as a whole should regularly outline some long-term goals, what individual models are going to accomplish in time; when code is going to be frozen; how much time is going to be set aside for testing, porting, validation, tuning, and documentation; and when new code is going to be released. The goals need to be realistic and the planning documents should include schedules, milestone targets, and priorities. In addition, longer-term planning should be used to prioritize issues like model performance, base code rewrites,
infrastructure support, and new physics.

It is recommended that a formal planning process take place before any work is done on the next major CCSM release (CCSM2.1?), outlining goals and schedules.

In order to improve our ability to plan changes and estimate schedules, regular reviews of progress should occur. Does CCSM meet short-term goals on schedule? Are long term-goals met? How accurate are the scheduling estimates, and are there particular issues that come up regularly that cause delays? Reviews should become a standard part of CCSM software engineering process.

3.7 Management

Many of the recommendations in this document require a stronger management structure. There needs to be agreement within the community that these policies will be used and enforced. The CCSM scientific coordinator and the SSC need to be in agreement with these policies, and they need to make sure processes and policy are understood. There also needs to be a oversight process where individuals can be disciplined if they violate policy. The goal is to better control quality and improve productivity, while still allowing individuals to carry out independent, creative scientific and software development.

3.8 Training

Over the last year, there have been a number of useful training activities including formal book discussions and local training by an external company. These activities have focused on improving the group’s understanding of software engineering process. It is now time to review what has been learned and use some of that new knowledge. In particular, the action items outlined in the Construx training need to be revisited.

There is currently a project to improve the CCSM GUI and scripts that will attempt to take advantage of these newly learned skills. Currently, the project is in the requirements collection phase. The project will move through an architecture and design phase before beginning implementation. It is recommended that new projects also adopt some formal software engineering processes in order to improve quality and productivity.

Training opportunities should continue to be made available to CCSM software engineers in areas related to procedure, process, project management, and technical skills.

3.9 Other Issues

There are a number of other issues that are worth bringing up as a part of this discussion.

A code review policy should be implemented. All component code changes should be reviewed before merging onto the main trunk and/or before release to CCSM. Another final review is probably required of CCSM code that is to be formally released to the public. The CRB process will handle some of the reviews, but maybe there needs to be more formal review upstream of the CRB.

Model output control and access needs to be better defined. A policy should be put in place with regards to data storage, naming conventions, data locality, data compression, and access. A separate data management group should be formed to deal with this issue immediately, and it should include members from the CCSM community, especially the Climate Change and Paleoclimate working groups. Along the same lines, standard
analysis plots need to be better defined and available, and questions regarding analysis and plotting tools needs to be addressed. Some progress has been made on these issues in the last six months, particularly with regard to naming conventions and standardization of model analysis plots, but there is still a deficiency in procedures related to model output and analysis.

The role of a computer scientist in the overall strategic planning of CCSM software engineering should be investigated.

4 Roles

The question addressed in this section is “who is going to do what?” The CCSM Software Engineering Group (CSEG) will continue to take a role in managing the process described in this document. However, much of the work will be done in conjunction with the CCSM working groups and the wider community. The component working groups will be responsible for helping manage repository access, change review, planning, and documentation of individual components. The software engineering working group will continue to oversee the CCSM software engineering process and make recommendations. The applications working groups (Climate Change, Paleoclimate) will be carrying out much of the production runs. As such, they will be granted limited read/write access to the repository like any other development group so production modifications can be archived and possibly merged back if requested and approved. The applications groups also need to play a large role in defining data access policy.

As indicated above, CSEG will play a large role in coordination. Individual tasks in CSEG include overall coordination, liaison to outside collaborations, and repository access monitoring. Individuals will continue to be tasked with component development, documentation, and performance optimization. Significant resources will also be allocated to testing, documentation and web support; and tools support, evaluation, and implementation. CSEG should also play a role in data management. The CSEG liaison to a component might also act as gatekeeper or test engineer for that component.

5 Summary

- Develop a formal process for the atmosphere CRB, meet regularly to carry out this process.
- Implement a CRB for the overall CCSM.
- Implement CRBs for ocean, ice, land, and coupler components.
- Improve repository monitoring tools.
- Improve testing capabilities. Make these capabilities available to the community.
- Investigate status accounting tools to replace csm-bugs wreq software and add new features.
- Continue to improve documentation. In particular, review the set of documents that currently exist, remove redundancy, and update documentation.
- Carry out some short- and long-term planning, especially as related to a next major CCSM release. Record plans.
- Review what we’ve learned so far with respect to software engineering process and start using some of the new knowledge.
• Continue to provide training to software engineers to improve their productivity and quality of work.
• Review the needs with respect to a data management group.

6 Appendices

6.1 Appendix A: Software Coordination Issues raised Sept 19

Several issues surrounding the coordination of the atmosphere model development and interactions within
the group and with collaborators was brought up at the Sept 19, 2001 CCSM programmers meeting. The
point was that CCSM as a whole probably has to deal with many of these issues as well. A group discussion
ensued. The items discussed included

• loss of control of model versions. Model changes and model versions are either not well enough doc-
dumented or not well enough communicated within the community. There is a lack of understanding
about which mods and which physics are in model versions and runs.
• poor understanding in the model development community about who is doing what.
• how to coordinate changes among several development groups including wacm, fv-dyn, etc.
• how to interact with outside developers including NEC and Fujitsu with regard to access to the repo,
testing, verifying code changes, and merging code back into the repository.
• lack of control of a "frozen" version.
• model release frequency.
• concept of a change control or change review board to control changes and slow down and better
coordinate the development process.
• inadequate testing standards. "A bug shut us down for 2 weeks".
• lack of policy on repository access.
• inability to secure branches of the repository from global access.
• increasing long-term planning and goals for model development.
• whether cgd has the software engineering expertise and resources to do both code development and
manage infrastructure.
• how all this fits into software engineering training and consulting.
• how other model development centers and commercial software shops manage their projects.
• the possibility that the project should hire a computer scientist to look into and plan for software
engineering and infrastructure issues.
• what the software engineer new hires might do.
• how the atm model / working group differs from other model development group in terms of coordi-
nation and control.
• whether we need to "stop" to get the infrastructure together.
• the fact that there is no clear, well-defined management /coordination plan associated with scidac and
that little resources are specifically allocated to deal with the software engineering issues. This is true
of other collaborations as well. They focus on code development and not enough on infrastructure.
6.2 Appendix B: Construx Training Action Items

In December of 2001, Construx and NCAR’s CCSM Software Engineering Group (CSEG) had a three-day training and consulting session at NCAR. The training was given by Steve Tockey and was based on Steve McConnell’s book, "Software Project Survival Guide." The CSEG had previously read and discussed the book, chapter by chapter, and felt the general approach was reasonable and appropriate for improving CCSM’s software development process.

The resulting identification of CCSM/CSEG project challenges, potential solutions, and a suggested action plan are summarized here.

Software Project Challenges:

- Uncontrolled requirements creep
- Small penalty for lateness/no incentive for early (Or are we overly minimizing the penalty for lateness, given the impact on respect and credibility?)
- No call to do any planning
- Can’t test some theories without huge investment
- Inadequate testing
- No consensus on what testing is necessary
- Little review of process for improvement
- Increasing complexity of the environment
- Non-existent requirements
- Lack of planning
- Lack of active risk management
- We have to deal with some poorly developed code (legacy code?)
- Missing product management

Back-to-Work Action Plan:
By next week

- Revisit this and do a severe sanity check on it - Tony
- Look at SEP-PCH and CxOne charter templates and make CCSM Charter template - Brian
- Look SEP-RAA and CxOne to develop risk and asset process/template - Lawrence
- Draft up charters for CCSM and 5 components - Lawrence to lead, liaisons to do for components
- Discuss in defect tracking configuration agenda: Track defects including when injected, when detected, effort/cost to fix - Lawrence
- Send Earned Value references to Tony - Steve
- Get and read these books on testing: - Lawrence

- Send lessons-learned process and template to Tony - steve

By next month

- Make a presentation of all of this to the scientists - Tony
- "Finalize" charters for CCSM and 5 components (should be put under change control) - Jeff
- Update the risks list (look at assets also) (not necessarily under change control) - Lawrence
- Plan (for the software) based on charters, risks, and assets - Tony and Component Liaisons(?)
- Get Suzanne Robertson and James Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999. Pay particular attention to Volere Template and the "Fit Criteria" - Tony
- Put in place a change control process - Tony
- Start meetings of the atmosphere "change board" - Jeff
- Put planning checkpoint reviews on Jeff's radar screen (point him to relevant chapter in SPSG, point out that "no go" option is off the table) - Tony
- Have people who do planning read Chapter 7 (Lifecycles) of McConnell’s Rapid Development
- Look at CxOne project plan and lite project plan templates, start thinking about your own project plan template - Tony
- Review the last 18 months/lessons learned - Tony
- Make sure lessons-learned is in the project plan template - Tony

By next quarter

- Develop a consistent approach to reviewing code - Brian
- Get and read the PMBOK (<a href="http://www.pmi.org">www.pmi.org</a>/ - Tony
- Start meetings of the remaining "change board" - Jeff
- Complete a reading/discussion group on Suzanne Robertson and James Robertson, *Mastering the Requirements Process*, Addison-Wesley, 1999 - Tony
- Go investigate Earned Value - Tony
- Get copies of
  - Donald C. Gause and Gerald M. Weinberg, *Exploring Requirements*
  - Quality Before Design, Dorset House, 1989
- Look into peer reviews: - Lawrence
– Daniel Freedman, Gerald Weinberg, Handbook of Walkthroughs, Inspections, and Technical Reviews
– CxOne peer reviews stuff

By next 6 months

• Extract defect tracking data and plot your own "Upstream, downstream" graph - Lawrence

By next year

• Start looking into the Cone of Uncertainty. Think about how it might be applied within the organization given all that’s happened up to now - Tony

Background stuff

• Get a couple of copies of
  – Michael Doyle and David Straus, How to Make Meetings Work
  – Everett Rogers, Diffusion of Innovation, 4th Ed.(?), Free Press, 1994
  – Project Management Institute, Project Management Body of Knowledge (PM-BOK), available at www.pmi.org

• Think about Product Management

6.3 Appendix C: Terms

• CCSM: Community Climate System Model
• CSEG: CCSM Software Engineering Group
• CRB: Change Review Board
• CVS: a version control system