CCSM Software Engineering Coordination Plan

Oct 25, 2001

This document proposes 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 specific changes that are recommended. Section three defines roles. Section four is a summary of recommendations. There is also an appendix that describes a number of software coordination issues this document is intended to address 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 is the status quo. This no longer works well for CCSM.

CCSM is currently coordinated as follows. There are individual working groups that are responsible for particular model components and/or analysis. These groups are relatively free to develop and test their components any way they choose. The one requirement is that they check the model into a CVS repository at NCAR at some point. The overall CCSM model is periodically tagged and production runs are then typically started. Very limited testing is conducted, there is no formal review of changes and change requests, and there are few policies with regard to process or coordination. Separate scientist and software engineering meetings are held nearly weekly and much coordination is carried out through personal communication.

2 New Policies

The following sections describe in greater detail improved processes and tools that, if used in a coordinated way, will improve CCSM software quality and control. In particular, implementing a new CVS access policy, creating change control boards, increasing testing significantly, and maintaining project status are all a part of the plan.

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 software product to the community that is stable and of high quality. This proposal attempts to balance the needs of the individual developer with the needs to 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.

Ultimately, the goal is to be able to do “better science”. One part of that goal can be achieved by improving the software quality of CCSM and making people in the project more productive. The recommendations in this document strive toward those goals.

2.1 Change Review Board (CRB)

To date, the CCSM project has relied almost entirely on informal software configuration management processes to coordinate CCMS code modification. Our current tool, CVS, deliberately puts very few restrictions on the developers ability to change code, allowing for wide-ranging modifications to be easily applied into virtually any element of the code. This capability for rapid development is of great utility during the development and testing stage of the software process. However, this freedom comes at a high cost. Without formal reviews or impact assessment of changes, hidden bugs and unintended component interactions can be introduced into the code with equal rapidity and ease.

As the code matures or release dates near, the stability of the system becomes an increasingly high priority. Experience from the software industry has shown that when code quality and stability, rather than rapid code change, is desired, it is wise to incorporate a more formal review process. Under formal review, modifications to production level products (code or documents) are made only with prior approval. Upon completion, all modifications undergo evaluation for correctness and impact on the rest of the system. Ad-hoc code modification is no longer permitted in release code.

In conjunction with a better defined policy for repository access (see below), separate change review boards (CRBs) should be defined for both CCSM and CCM. The other components can incorporate this process as they see fit. The CRB is responsible for approving all configuration changes to baseline code (code on the main trunk). This ensures that all proposed changes receive a technical analysis and review and that they are documented for tracking and auditing purposes. The board also has final responsibility for release management. Many elements of the change review board already exist in CCSM. The goal must be to officially establish the board to gain formal control over the change approval process as it affects configuration control. The basic task of the CRB is to declare baselines and then review changes and approve, disapprove, or defer their implementation. The above is an extremely important task. The CRB must have control of the project. Nothing should be changed without their approval or knowledge. For this reason, board members must be chosen carefully.

Requests to the CRBs would be via a web form. Requests would take several forms. Developers would add and update requests to document current efforts and status. They would then make a separate request to merge their development work into the production software. Only requests to merge work into production versions would be reviewed by the CRBs. The general status requests are used to improve visibility of development efforts.

In many ways, the individual working groups act as change review boards for their components and the CCSM scientists meeting serves as a surrogate to a CCSM change review board. However, the process should be
better formalized within both CCSM and CCM. For CCM, the change review board should include a small number of scientists and software engineers from the climate modeling section and/or atmospheric model working group. All CCM change requests and the status of changes would go through this change board. All development work would take place on CCM branches and the change board would meet regularly (about 1x/month) to approve branch merges onto the main trunk. A gate keeper would merge these changes, carry out testing, and then tag the version. The merge onto the main trunk would also be an opportune time for a code review. This would solve a number of concerns indicated in Appendix A including important issues like understanding who is working on what, an increased ability to document the model status, prioritize efforts, coordinate the repository, test appropriately and maintain quality. In CCSM, the CRB should include at least Jeff Kiehl and Tony Craig plus some additional representation from the science and software engineering areas. The CCSM CRB would meet to plan, prioritize, and schedule changes to CCSM. These can be both for short term and long term changes. The CRBs act as a formal review board. Input from a wide range or perspectives is required to make decisions. The mandate is to review change requests based on input from many.

A fast-track review process will be put in place to correct model bugs. This will only be used to quickly modify a recent production version and only for fixes that do not require rigorous reviews or testing.

2.2 Repository

There are a number of issues surrounding the CVS repository and access. CCSM has a large and growing developer base and there are concerns over privacy, testing, and control. Sharing code via tar files is impractical because of the code divergence and merge costs. Granting open access to the repository is dangerous. There are limited resources available for exploring or purchasing new tools.

CCSM should continue to use CVS as a tool for source code control. CVS is free and CCSM developers are finally becoming 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.

The primary problem with CVS is its inability to read and write protect parts of the repository. In the CCSM repository, access can be limited to particular components, but access within a component cannot be formally controlled. Access control within a component can be solved by implementing a formal CVS policy in CCSM that grants access with certain constraints. Users would be granted permission to read and/or write only on specific branches and/or over specific time frames, usage would be monitored via the CVS logging capability, and user access would be terminated upon violation of the terms of the policy. The scope of access would be determined on an individual basis by the appropriate CCSM component working group. The policy would also outline rules on things like naming conventions and testing requirements. In addition, individual components should consider implementing a policy where development is allowed only on branches and that a gate keeper is in place to manage the main trunk under direction of a change review board. This allows for continued rapid development while maintaining control of the production code.

Another option for controlling CVS access would be to have separate, distinct CVS repositories for individuals who are concerned about privacy. CVS import and export would then be used to bring source code into and out of the CCSM repository. This option needs to be explored further, and if feasible, could be used to protect the most sensitive pieces of source code.
2.3 Testing

An individual should be tasked with developing, running, and maintaining testing software for CCSM on a full-time basis. The CCSM test engineer would be responsible for 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. In addition, the CCSM test engineer would provide test scripts for developers and assist in developing acceptance and unit tests. It is likely the CCSM test engineer would be a new hire in the group.

A substantial increase in personnel resources and time must be spent on testing. At the current time, changes go through minimal or no testing before climate simulations are started. This is very dangerous. Significant increases in quality and decreased risk will accompany increased testing. For comparison, typical software shops have about one test engineer for every two developers. CCSM falls far short of that ideal. Given the number of scientists, software engineers, and collaborators that carry out development, setting aside one full time person to be a test engineer and asking that all developers spend more time testing is a minimal requirement for success. Increased testing will increase the turn-around time of source code changes, but it should also increase quality and overall productivity.

2.4 Status Accounting

Status accounting is the ability to both understand what changes have been implemented, what changes are proposed to be implemented, what defects have been discovered, who is working on these, and what the status is on each. A software tool to document the CCSM status can also be used for bug reports and to keep track of testing status. 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 and that this be made relatively high priority. The accounting tool would be an integral part of the change review boards and vice versa.

2.5 Documents

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. The CCSM developers guide also needs to be kept better up to date and needs to be more complete. This document serves as the guide for all CCSM development! Most of what is in the document you’re reading now, if it’s adopted, belongs in the CCSM developers guide. 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.

2.6 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. Part of the planning process on all time scales should include improved cost/risk/benefit analysis to set priorities to make best use of people’s time.
Currently, short term plans span about a week of time and within that time frame the plan changes continuously. A plan is often put in place on Wednesday which includes changes to several components and a requirement that the system be running production by Friday. In addition, the plan may change on Thursday and Friday but the requirement to run on Friday does not change. This leads to minimal testing, little quality control on source code, and no time to document the changes that are contained in a particular version.

Model changes should no longer be incorporated into production versions unless they have been thoroughly validated and are shown to improve model simulation, and all change requests should go through a change review board. The risk associated with incorporating incompletely validated code is very high. The project risks releasing low quality code and carrying out suites of simulations with physics and software that are not fully understood. Most other climate modeling efforts take a much more conservative approach to incorporating new code than CCSM.

In terms of longer term CCSM planning, what typically happens now is that a model release date is stated months in advance with little thought given to what the model will contain or how the development will advance given the schedule. Under CCSM2 development, a December, 2000 release date was announced 6 months in advance and then there was a mad scramble to try to meet that date. In the end, the model will probably be released a year late (18 months rather than 6 months). During this 18 month development and release process, there was little planning. Each week was a new frenzied attempt to get the model frozen with no broad understanding of the requirements.

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 to prioritize issues like model performance, base code rewrites, infrastructure support, and vectorization optimization is required.

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 needs to be reviewed and revised regularly. In addition, the role that a computer scientist could play in the overall strategic planning should be investigated.

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 poor 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. In December, 2001 the past 18 months of CCSM2 development should be reviewed to understand what went well and what went poorly to improve future expectations, process, and scheduling.

2.7 Management

Many of these recommendations 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 on-board, 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.
2.8 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.

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 applications groups Climate Change and Paleoclimate. 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.

3 Roles

The question addressed in this section is “who is going to do what.” The CCSM Software Engineering Group (CSEG) should 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 rest of the 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 might 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.

4 Summary

4.1 General Summary of recommendations

A summary of recommendations is presented here. The recommendations reflect a coordinated effort to improve software quality and control by implementing a suite of new policies related to repository access, change review, planning, testing, and documentation. If implemented correctly, all of these recommendations can work together to significantly improve productivity and quality.

- Better planning should be incorporated into the CCSM process at short term, long term and strategic time scales. Process reviews should be done on a regular basis, and in December, 2001 the past 18
months of CCSM2 development be reviewed to understand what went well and what went poorly to improve future expectations, process, and scheduling.

- CCSM should continue to use CVS as the tool for source code control. A policy should be put in place for CVS access, and development should be limited to branches only.

- Separate change review boards should be defined for both CCSM and CCM.

- A person should be tasked with developing, running, and maintaining testing software for CCSM on a full-time basis. CCSM should focus more on quality and stability and closer scrutiny and testing must be done on all code modifications before they are incorporated into the production model.

- A person should be given responsibility and time to explore and implement a status accounting / tracking tool for use in CCSM and this should be relatively high priority.

- A person should be put in charge of coordinating documents to assure they are and remain up to date. It is also recommended that a review of the CCSM public and local web pages be reviewed in an attempt to clarify and improve document visibility.

- The CCSM Software Engineering Group (CSEG) should be given responsibility for managing this process.

4.2 Summary of changes to CCM coordination

A summary of how CCM would operate based on this document’s recommendations is presented here. Much of this also applies to other components.

Repository access would be granted to users on an individual basis. Limits on how, where, and when they could read and write would be outlined and their use would be monitored. Violators would lose repository privileges. Development would only occur on branches and a gatekeeper would be in charge on the main trunk.

A CCM change review board (CRB) would be put in place to review, prioritize, and coordinate changes to the main trunk. Individuals would make change requests, the board would ask for comments from the community, and then approve, defer, or deny a change request. The approved requests would then be prioritized for merging onto the main trunk. A status accounting / tracking tool would be put in place to help organize requests/status and to make sure the CRB status is visible to the community.

All approved changes will be thoroughly tested by the developer prior to being merged on the main trunk. As those changes are merged onto the main trunk, code reviews and independent, thorough testing will take place. Once a new version of CCM has been validated, it will be tagged. At the same time, all documents will also be updated and tagged along side the source code. No source code will be tagged until the accompanying documentation is updated.

Because of the new coordination and testing requirements, these tags will likely not take place more than once per month. These tagged version will formally be released for production. In addition, developers will
be asked to regularly merge the updates from the main trunk into their development branches. Bug fixes can be implemented quickly by creating bug-fix branches off tagged versions. These changes can be fast-tracked through the review process and may not require formal testing.

The atm working group would be responsible for long-term coordination and planning for CCM. The working group would continue to function as meeting place for developers and as the main community served by the CRB.

4.3 Summary of changes to CCSM coordination

CCSM work revolves around gathering the individual components together and running them in production. As such, improved coordination and software quality within the individual components will be a major benefit to CCSM.

CCSM development will now proceed along the following path. There will be improved planning and prioritization. This will be managed mainly by a CCSM change review board (CRB) which will meet regularly to assess change requests and prioritize them. Input on changes will be received mainly from the working group co-chairs and impacts assessment will be required of everyone. Long-term planning will also be part of the CRB’s charter. The CRB is also responsible for making sure changes are coordinated between components; for instance, if an interface is changed.

When new versions of CCSM are recommended, the complete model will be tagged and thoroughly tested. All associated documents will also be updated. Once the new version is validated, production can begin. Because of the new testing and documentation requirements, new CCSM tags will likely not be produced more than once or twice a month. Bug fixes can be incorporated quickly by using bug-fixed versions of released components. In that case, less rigorous testing of the updated model may be possible depending on the extent of the changes.

5 Appendices

5.1 Appendix A: Software Coordination Issues raised Sept 19

Several issues surrounding the coordination of the atm 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 documented or not well enough communicated within the community. There is a lack of understanding about which models 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 wacmx, f-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 coordination 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.

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