Graduate Student Test Report:
CMEPS 0.5 - SST Experiment

Report prepared by Cecelia DeLuca
January 25, 2020

About the Graduate Student Test

The Graduate Student Test (GST) is a measure of how successful the UFS project is in opening its code and development processes to the broader community.

The GST checks that a student can easily:

- Get code.
- Run code.
- Change code.
- Test code for correct operation.
- Evaluate code with standard diagnostic packages.
- Get documentation, user support, and training.
- Understand what is needed for their code changes to transition to operations.

There is not a single GST that applies to all codes, and there will be multiple GSTs developed to evaluate the usability of UFS. A particular GST may cover only a subset of the elements in the list above..

Why graduate students? The GST targets students because they are motivated to work with UFS and are important contributors, but may be new to its concepts, acronyms, and assumptions. This makes them an excellent test group for ensuring that UFS materials and processes are clear and understandable for everyone who wants to work with the UFS code.

CMEPS 0.5 - SST Experiment Test Description

This GST assesses how easy it is to access, run, and make modifications to source code to perform a sensitivity experiment with a coupled atmosphere-ocean-ice model. Preparation of the code base and test was a joint effort of the UFS Communication and Outreach Working Group and software developers at the National Center for Atmospheric Research, the NOAA Environmental Modeling Center, and George Mason University.

The model code used for this test is a prototype of the Unified Forecast System (UFS) Subseasonal-to-Seasonal (S2S) application. The model components included are the Global Forecast System (GFS) atmosphere with the Finite Volume Cubed Sphere (FV3) dynamical core (FV3GFS), the Modular Ocean Model (MOM6) and the Los Alamos sea ice model (CICE5). The components are coupled with the Community Mediator for Earth Prediction Systems (CMEPS). The Common Infrastructure for Modeling Earth (CIME) case-control system manages the workflow.

Participants downloaded and built the S2S application, ran it for 5 days, modified the source code to increase the sea surface temperature sent from the ocean to other components by 2^{o}C, reran the model for 5 days, and visually compared the results. They were given 6 hours to complete the test.

Summary:
Link to the test:
https://github.com/ESCOMP/UFSCOMP/wiki/Milestone:-CMEPS-0.5-Appendix-Graduate-Student-Test-Evaluation-SST-Experiment
Link to the questionnaire:
https://docs.google.com/forms/d/1Bc58rfkyUlSb2bftA5eNhWRiwNeD6UthjCogq9BI3m4/
Number of respondents: 4
Respondent affiliations: University of Albany, University of Michigan, University of Washington, George Mason University
Computing platforms supported: Cheyenne (NCAR), Stampede2 (XSEDE/TACC), Theia (NOAA)
Test dates: 9/26/2019 through 12/20/2019

Comments and Conclusions

This test had a twofold purpose. First, it was designed to introduce and exercise a GST protocol. The number of participants was deliberately limited to allow any major issues with the methodology to be addressed before more widespread engagement. Second, it collected information about the usability of a prototype UFS S2S application running in the CIME workflow. CIME will be used in the first community release of the UFS Medium-Range Weather Application, scheduled for early 2020.

A primary finding is that the test methodology worked well. Respondents were able to follow the steps to complete the test and provide feedback. A couple of respondents commented that the testing process was fun or enjoyable.

Another primary finding was that the coupled code running in the CIME workflow had the potential for good usability. This assessment is based on the metric that 3 of the 4 testers could get, build, run, and modify the code, and compare the difference before and after modification, within a six hour window.

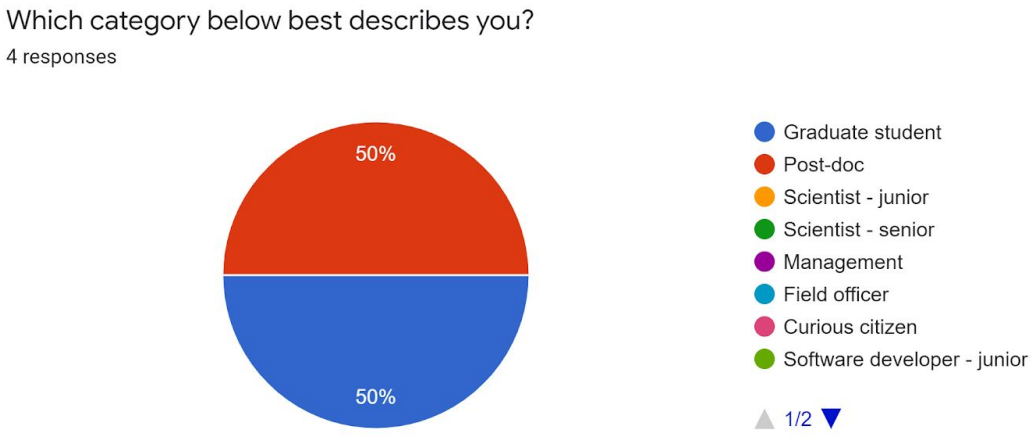
The testing process was useful in identifying and resolving access issues. One of the computer platforms used in the test was not set up correctly for external access to the test, but the issue was not visible to developers, who had ‘insider” permissions. The student trying to run the test on this platform found the issue and worked with developers to resolve it.

It was encouraging that two of the respondents flagged a discrepancy in one of the plots that was displayed as a check on results. This lends credence to the idea that enabling community members to run the code will help find errors and improve the quality of the software and documentation.

There were significant limitations to the code and documentation noted by respondents. The test instructions were prescriptive, due in part to the lack of availability of overall system documentation. The “canned” nature of the test limited its value as a teaching tool or as a basis for further development, and only a few computing platforms were supported. However, the respondents overall found the documentation that was available for this test clear and easy to follow. The best summary of what respondents wanted to see in the future was expressed by one of them as: “In short: it just needs more flexibility and documentation.”

Questionnaire Responses

1. Information about respondent



If you are a graduate student, what is your major area of study? 2 responses

Climate Dynamics

Atmospheric Sciences

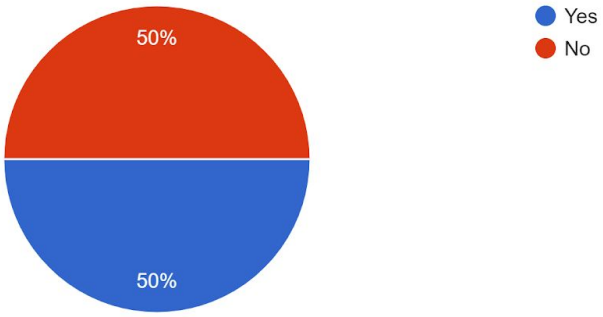
If you are a graduate student, what year are you in or what level of graduate work? (MS or Ph. D)2 responses

2nd year PHd

I am a 6th year Ph.D student.

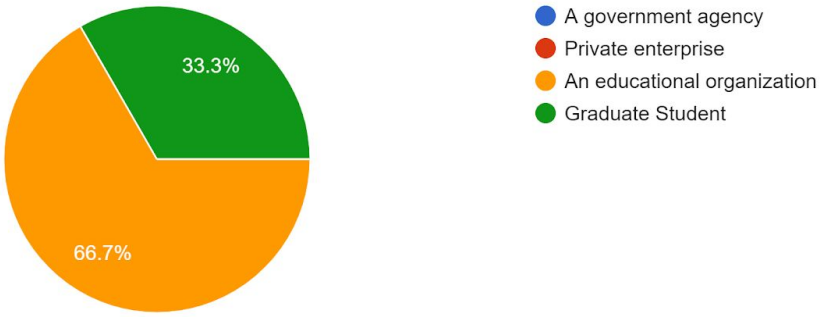
Are you employed?

4 responses



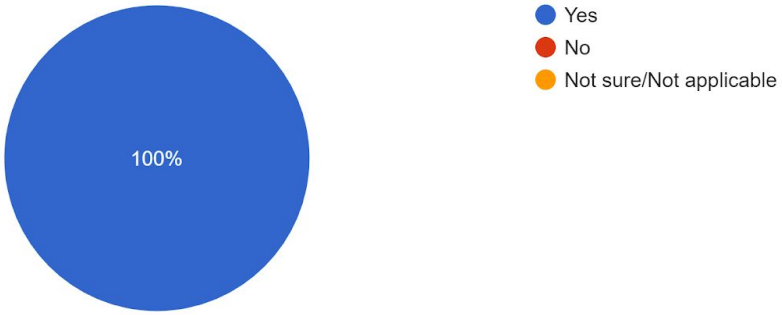
If yes, where are you employed?

3 responses



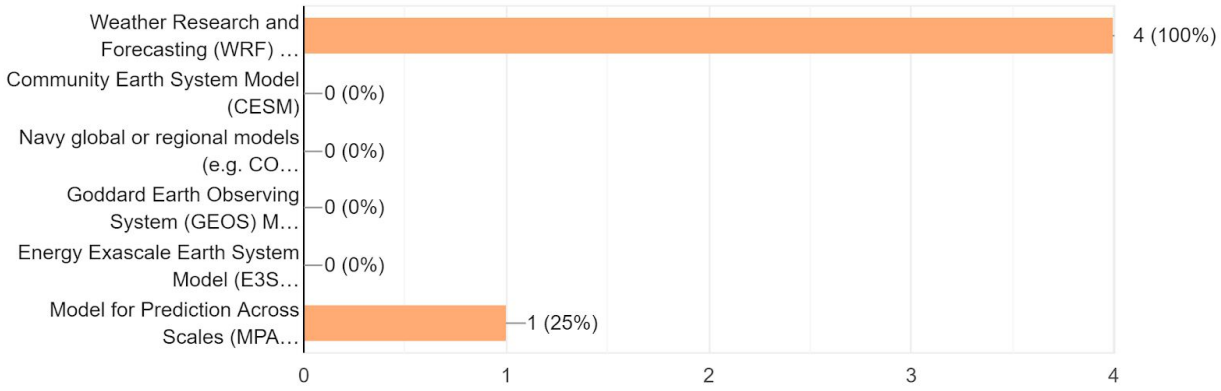
Do you use numerical models as part of your educational program or job?

4 responses



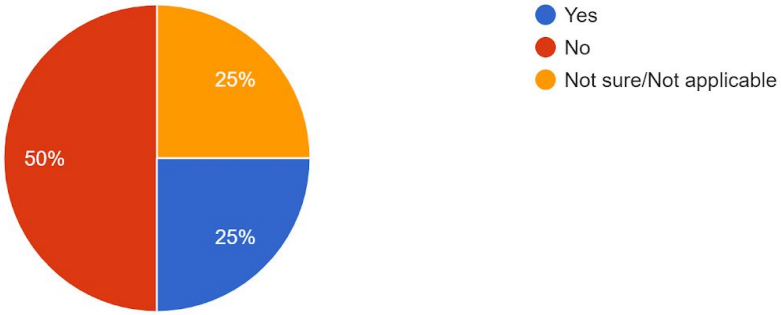
Have you run any of the following environmental models? Please check all that apply.

4 responses



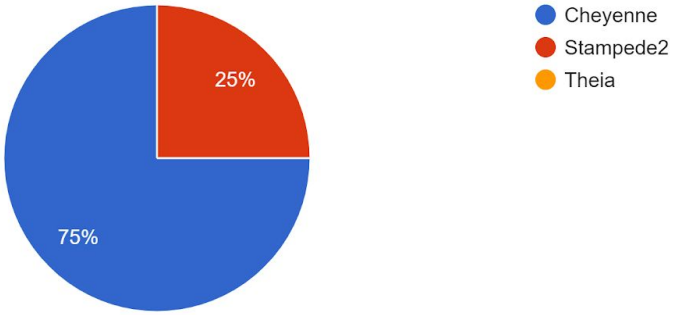
Are you currently involved in the UFS project?

4 responses



Which computer will you use to perform this test?

4 responses



2. Feedback on the test

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
I needed additional documentation in order to get started.	0	1	0	3	0
I found it easy to understand the configuration being modeled.	0	2	0	2	0
I found the code easy to get.	3	1	0	0	0
I was able to run the code without any trouble.	1	2	0	0	1
I found the code easy to modify.	1	2	0	1	0

3. Free response questions

Did the documentation provide sufficient information for you to understand the nature of the code you were asked to run and how it will be used? If not, what was missing?

4 responses

Yes. The documentation was easy to follow in order to get the model up and running.

As far as I was able to get, the documentation provided was sufficient. It was very easy for me to get started.

Yes, the CMEPS 0.5 Milestone documentation provided very clear and precise instructions for completing the test. At no point did I run into any issues (you can basically get away with copy-pasting all the commands). I would have loved a bit more description of what was going on "behind the scenes", even though it's not necessary for this specific test. For example, it took me a bit to figure out that the "./xmlchange" commands were modifying namelist entries in "env_run.xml." For those of us accustomed to modifying namelists directly, a bit more description would have been nice. (I would love a full description of all those xml files and their namelist entries). Another example would be the model outputs; there were *many* different output (*.nc) streams in the scratch directory and I had a hard time figuring out what they all were, even after looking at "diag_table" and "input.nml".

I think there should be descriptions of the given options for each step so the user has an idea of what each flag/option means. For example, what do the flags of "--driver nuopc --run-unsupported" mean when running create_newcase and how do those options influence the simulation? Explaining these flags in a test case or tutorial helps to teach the user about the different options available to them.

Were you able to complete the test in 6 hours? If not, how long did it take? 4 responses

I was able to complete the test within the six hour window. My previous experience likely helped in expediting the process, but I also think that the instructions were very clear and easy to follow.

Including all communications, it took me over 2 days to get just a reference case build to work. After that, I tried to submit my 1 hour cold start run, and while it executed, it did not produce the files seen in a "successful cold run" screenshot in the documentation. This could be due to my own inexperience with submitting jobs for stampede2 or something related to the code itself. I had plenty of permissions issues, which were documented in length via email.

Yes! I ran both 5-day simulations (concurrently) in ~1 hour and everything else took ~2 hours.

It took about 6 hours, but most of that was waiting for computational resources to become available.

What could be done to improve the user experience getting, running and changing this code? 4 responses

Nothing comes to mind. I think the color-coding of the text in the subroutine that shows what and were to modify the code is very helpful.

In short, I was not apart of the same group code, which caused a number of permission issues when trying to access files. I only was successful when the case build was not dependent on the yaml library and employed the ESMF 8. So the dependency on the yaml library was also a huge snag, but i think ufuk mentioned that a newer version of this code (when the esmf is released later this month) won't have this dependency. Another thing that I actually liked that i did not see enough of was built-in troubleshooting responses. For example, when I ran my case.build (after all of these other issues were addressed) I had another issue where git exceeded timeout limit of 300 seconds (probably trying to access another permission denied dir.), but there was auto generated output that walked me through what I should do to get around this situation (listed here): To solve this, either: (1) Find and fix the problem: [...], try to get this command to work: manage_externals/checkout_externals --status --verbose --no-logging [did not work] (2) If you don't need provenance information, rebuild with --skip-provenance-check [this resulted in successful case build]. Anytime that I, as a grad student or a code user, don't have to send an email to troubleshoot, is saving me and your team so much time and patience in information exchange. it would even be great to have auto troubleshoot output that recommends when they should contact you for troubleshooting. This obviously requires knowing the known and common problems when trying to access your code... hopefully I helped you find one pretty substantial one. I really wish i could give more feedback for the remainder of this trial, but I did not have enough time to do so.

Acquiring/downloading the code was no problem (aside from needing special permission for the FV3 Github). As a rookie Fortran user, changing the source code (mom_cap_methods.F90) without those precise instructions would have been tricky; I *really* like the inclusion of the SourceMods directory, which makes it really easy to modify the source code without having to worry about other cases. As a minor comment on the tutorial instructions: I think you should move the section where you tell users to modify the wallclock time to 6 hours to earlier in the tutorial, since they will need to do that for the control run as well. Also, it wasn't clear to me if I should modify the source code

before the "case.setup" step or the "case.build" step.

Getting the code is pretty straight forward. I think what would be helpful is a file directory guide where it breaks down what is located in each folder of the system. This would be beneficial for understanding where to start searching/exploring for potential future changes/experiments.

Could you use this code in your work? If so, how? 4 responses

Yes, the examination of aerosol impacts on weather forecasts in the UFS framework is something that I'm interested in.

---

in the future, yes, i would like to use this if it is land-ocean-atmo coupled.

In its sort of "pre-canned" release state, I don't think there's much I could do with UFS right now.

I could as I am exploring coupling of lake and atmospheric models.

What are the highest priority additions you would make to the code to make it more useful to you? 4 responses

By adding an interactive aerosol module to the code. This is something that I will be exploring in collaboration with EMC in the future.

---

you need to have an accessible troubleshooting q&a for people to reference. ecmwf data access sites and some repositories / and wrf both have one that saved me countless hours of emailing. also permission work arounds.

In order to be useful for research applications, a UFS release needs to include 1) preprocessing tools for folks to make ICs and run their own cases, 2) tools for users to change grid resolution, and 3) documentation on the modification of namelist parameters, output streams/diagnostics, physics packages, etc. In short: it just needs more flexibility and documentation.

Complete documentation on the code structure including how components are coupled and information regarding the atmospheric component.

Is there anything else you would like us to know? 4 responses

In the output section of the documentation, there is a Figure that shows the difference in the mid-level boundary layer height between the two cases using "ncview". After running my simulations, I compared this Figure to my ncview output, and they appeared to be slightly different (even after changing the range to match with the Figure). In contrast, I ran the ncl code that was provided and produced the same exact Figures shown in this subsection. It could be the case that I incorrectly typed in something wrong when performing the "ncdiff" command, but if not, then I think it would be helpful to match the ncview Figure with the user's output.

---

i found the written documentation to be written well, and I enjoyed having screenshots of what specific outputs should look like, I have not seen that before.

Thank you for allowing me to participate! It was a lot of fun to use a new model. One more question/comment: when I used "ncdiff" on the mom6 surface files, as in the instructions, my output in diff.nc looked different than your figure (I had lots of positive changes in MLD rather than negative ones...) and my SST differences were actually largely *negative*... But my figures made with the NCL script looked exactly like the ones in the instructions, so I'm not sure what the issue is/was.

The code is very easy to download and setup on the listed machines. It would be great if this could be ported to other machines, but I understand that that would be a MAJOR undertaking. For each step of the example provided, I think more details in the why these selections are being made. I know personally, from previous tutorials I have taken with atmospheric models, a lot of the selections from the tutorials