Comment on gmd-2021-311
Stephan Kramer (Referee)

The manuscript "CP-DSL: Supporting Configuration and Parametrization of Ocean Models with UVic (2.9) and MITgcm (67w)" describes the implementation of a Domain Specific Language approach for the configuration of ocean models in the form of a new language CP-DSL. The subject of this paper addresses an important topic regarding the often complex configuration process of ocean models with a number of strongly conflicting issues like user friendliness for users with a wide range of expertises, the wide variety of competences involved in the construction of accurate and efficient ocean models which affect both the way the software is developed but also its optimal configuration, and the fact that different models have quite different configuration processes, where the expertise of an advanced user of a specific model does not easily translate to that of another model. Although I believe a number of interesting ideas and techniques to address these challenges are presented in this manuscript, a number of issues in the presentation make it quite hard to evaluate what has actually been achieved.

Specific issues throughout the manuscript:

- the most problematic section in my view is section 5 which provides a very abstract overview of the syntax of the proposed language but only through a few excerpts that do not give a very clear picture of the language as a whole. I also do not find the UML
diagrams to be particularly enlightening (figures 3-6). I would really like to see a more complete overview of the features that have been implemented, and a lot more concrete examples of actually what goes into the "configuration model" vs. the "declaration model" so we can get a better idea how universal the language is and able to specify things in a model independent way.

- it's only in section 6 that we finally get told what CP-DSL is actually made of, but the description of the key components in section 6.1 is very terse. Please explain for instance what EMF is. As a side note, it also seems that some of these components bring in a dependency on a specific version of Java which seems to be in contrast with one the requirements (line 251-253) and isn't particularly well supported on some of the HPC systems that ocean models run on.
- as mentioned before the authors do not really evaluate or discuss what has actually been achieved in this work. The evaluation by other users in section 7 is described in a rather superficial way. The first evaluation in section 7.1 seems to be about a quite different version of the language, so the only information we get, about a second evaluation is "The division in general parameters and modules was considered useful. Also the reworked YAML syntax was rated easy to understand." This is actually the very first place (right at the end of the manuscript) where it is actually mentioned that the CP-DL syntax is closely (?) related to YAML - the other places YAML is only mentioned to contrast with XML and JSON. From the conclusions:

"As this is an ongoing research project, we aim to further extend and improve CP-DSL in close contact with users and active scientists from the domain. We initially developed the DSL for a representative subset of MITgcm ocean modeling scenarios and are currently evaluating it to be able to support all modules of MITgcm. However, the current syntax for diagnostics caused a larger comment by our domain experts, as diagnostics are an important topic in climate modeling for various purposes."

Having read the paper I still have little idea what representative subset has actually been implemented.

- I tried to evaluate the software, also to get a better idea about the structure and functionality of the language, but I'm afraid I didn't get very far. There is no documentation at all that is directly accessible, just some installation instructions that were unfortunately insufficient to guide a user, like me, with no experience of using java, maven, etc. For me it is very unclear what the different parts (cp-dsl, cp-dsl-replication, cp-dsl-jupyter-kernel) consists of and how they are supposed to work together, and how they should be installed such that the different components can access each other.
- As discussed ocean modelling already brings together a variety of expertises (e.g. oceanography, numerical analysis and HPC) and the CS flavoured approach followed in this paper with DSLs, context-free grammars and metamodels adds a whole layer on top of that. This makes it important to have a clear view of the intended audience and adjust the language to it, briefly explaining key concepts. In particular if the intended audience is ocean model developers, who may be familiar with many advanced computational and numerical techniques, but not with tools and terminology common in DSL approaches, some more guidance would be helpful. Here it is also important to be
aware of how terminology various between different communities and be specific about what definition is being used. As an example, the authors already point out that the concept of language models adds a new meaning to word model in the ocean modelling context, but within the ocean modelling community the word already has a range of meanings: a conceptual model of the global ocean, a description of its physics, a translation of that into a mathematical model, which in turn are translated into numerical equations whose specific software implementation is also referred to as an ocean model, and finally a specific configuration of such a model for a specific scenario is again referred to as an ocean model. In this paper the authors choose the (appropriate) definition of a specific software implementation (line 16), but then in line 140 we have "The Model Developer is a software developer and responsible for transferring the ocean models into code" which contradicts that definition and makes it hard to understand what the difference between a "Scientific Modeler" and a "Model Developer" is. As another example, the authors make a distinction between configuration and parameterisation. Here it is important to note that "parameterisation" already has a very specific meaning in the ocean (and atmosphere) modelling community, it refers to processes that are not modelled through PDEs on a numerical grid, but rather through empirical parameterisation of these processes (typically on the sub-grid scale). "Parameter selection" might be a better description of what is meant in the paper.

Configuration is defined in line 37 as "the selection of features and code to be used for the model, as well as, the build configuration." but then on line 268:

"For each simulation experiment, we need to define a configuration and a parametrization. Independent of a specific experiment, we declare settings that are specific to an ocean model. Thus, the parameters and configurable features are declared with CP-DSL in a Declaration Model specific to each supported ocean model, as depicted in Figure 2. The Configuration Model is independent of a specific ocean model, it defines the settings of a concrete experiment, whereby the declarations in the Declaration Model for the specific ocean model are referenced. This way, we separate the ocean-model-independent and the ocean-model-dependent settings"

which seems to bring an entire new definition of configuration which is used along with the old in the same paragraph.

As a final example, the word "deploy" and "deployment" is used in a number of places: "Model developers also deploy the software" (line 141), "the deployment is merely configuration and parametrization" (line 163) - and I don't really understand what is meant there - I'm more familiar with its usage as in line 207-214.

Some suggestions for further references
Note: these are suggestions from personal experience only, I don't think the current references are lacking
Section 2 provides a fair overview of previous DSL approaches in the ocean modelling
context. Although section 4 gives a good overview of the specific requirements for a configuration system for an ocean model, there isn't a clear separation between those requirements that are specific to ocean models, and those which are in common with the configuration of other types of scientific modelling software, e.g. atmosphere models. As one of the key decisions in the design of DSLs is based around finding the right level of abstraction, it would be worth to extend this discussion to scientific models in general and explain why an ocean-modelling specific language is required, or whether it could be built on top of a more generic approach for the configuration of scientific models in general. As an example of the latter, I have personally been involved in SPUD [5] an XML + RELAX NG based configuration system for scientific computer models (I do share the author's preference for more plain text formats btw).

Already mentioned are Psyclone, Dusk/Dawn and Sprat which target other layers of the software stack, in particular PDE discretisation in combination with automated code generation. In this context it might be worth mentioning the popular FEniCS [1] and Firedrake [2], and DUNE [3] projects which make extensive use of such approaches (for context I'm one of the authors of Thetis [4] a coastal ocean model based on Firedrake). You do mention ICON in the context of diagnostic configuration, but I believe there is more DSL-based ICON development described in [6]. Finally the Atmospheric Modelling Language (ATMOL) [7, 8] developed with the Royal Netherlands Meteorological Institute may be worth a mention.


Some smaller specific comments:
line 89: "by model developers what, in principle, allows" - I think you mean "by model developers which, in principle, allows"
lines 186-190: I would suggest replacing grid density, dense and wide with grid resolution, fine (grid resolution), and coarse which are more common
section 3.5: I don't really understand the difference being made between versions and variants, and how it is relevant to the rest of the paper
line 203: why mention openmp but not MPI (the Message Passing Interface not Max Planck!)
"Code management and sharing via tar-balls, ssh and email is mostly deprecated and replaced by Git, often with the Git Large File Storage (Git LFS) extension." I think that's a little too strong; the use of ssh and shared file systems is still very common practice.

What is meant by two layers?