Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-339-AC2, 2021

Geoscientific Model Development Discussions

Interactive comment on "Inishell 2.0: Semantically driven automatic GUI generation for scientific models" *by* Mathias Bavay et al.

Mathias Bavay et al.

bavay@slf.ch

Received and published: 1 March 2021

Answer to Anonymous Referee #2

> 1) It seems to me that the authors did not adopt the best techniques for the objective. For example, the authors adopted C++/Qt for creating the GUI, which means that the tool is dedicated to the local desktop environment. I was hoping that the tool can be Web based to allow wider and easier access. Another example is about the XML and INI. I understand that INI is more human friendly; however, a better option is YAML, which is also human friendly and better on supporting complex hierarchical structure, which is important for describing the models.

Regarding the choice of execution environment, the authors had an extended period

of time to reflect on this topic before starting working on the major rewrite that is presented in this paper. We considered both a web based version and a fat client version. In the end, we felt that on the long term, most probably both version will be required, although for different use cases. As we have also been leading the development of a complex web based data visualization tool (see https://niviz.org for visualization and editing of snow profiles, 121k lines of code) over many years, we have gathered experience in both kind of technologies. A web based version offers an easier access (no need to install anything) but suffers from our point of view from the following limitations: first, we had noticed that for our users, being able to run the numerical model from within the GUI is a major requirement. On of the major negative feedback to our numerical models was that the users had to open a terminal, go to the proper directory and run the numerical model from the command line. This had implications in terms of support requests (many requests were directly related to using the terminal), users frustration and acceptance of the original Inishell GUI (many users would not go back and forth between Inishell and their terminal emulator and ended up discarding the original Inishell, thus loosing the benefits of input validation, online documentation etc). Unfortunately, because of the sandboxing requirements of web technologies it is not possible to run a local program on the user's computer from a web environment. Another limitation that we saw is the relative lack of maturity of most of the client-side web technologies. A complex web based tool will be based on numerous third party libraries that so far evolve very fast and often lead to incompatibilities in a very short time frame. This means that the long term maintenance of a complex tool almost mandates full rewrites of the said tool every couple of years in order to migrate away from deprecated dependencies. This represents a major investment for an open source software that is not self funded. Finally, the programming model of Qt/C++ is much more familiar to the traditional scientific model developer than that of Javascript, meaning that low complexity changes can be implemented by a scientific model developer if need be in the fat client version while almost any change in a web based tool requires an outsourcing contract (although the idea behind Inishell is to avoid having to edit the

source code, there might always be some minor bugs and annoyances to fix).

Regarding the choice of file formats, it was also a compromise. We considered moving away from the XML files to store the configuration data data model but preferred to keep this format in order to reduce the risks associated with the rewrite that we were performing. As these files are not directly exposed to the end users, we might still move to a different format and different syntax at a later stage. But independently of the format and syntax, the main point is that the current level of expressiveness covers adequately the needs of multiple scientific numerical models. The format of the storage of the configuration data (INI files) has also been the result of a compromise. Within our field of cryosphere modeling, there is already a wide range of strategies to deal with configuration parameters, from direct source code editing (the parameters are therefore hard-coded, this happens often in models implemented in a scripting language such as Python, Matlab or Julia), INI variants (usually without even an explicit and consistent syntax definition), INI variants with some namespace structures (although some model only have one namespace that starts and ends the configuration file) and XML files. Some models relied on multiple configuration files or a mixed approach between hard-coded configuration parameters (that had to be manually edited in the source code) and some configuration files. We decided quite early in favor of an explicitly defined INI syntax in order to keep the configuration files sufficiently similar to the legacy ones so the end users would not struggle too much when porting their configurations to the new syntax. This is also the result of a compromise between the user friendliness of the syntax and its expressiveness and robustness: we always had to support direct editing of the configuration files (ie without relying on our GUI which is specially relevant until we can ensure that our GUI can cover all the needs of all our users) as well as editing through scripts (for automation such as running a large number of related simulations).

> 2) The paper did not provide a clear approach on handling geospatial information, which is quite common for geoscientific models. I noticed that the paper briefly mentioned about the geographical coordinates, but did not mention other forms of geospa-

tial information, such as polygon, raster.

So far, all the geospatial information that we have encountered beyond simple geographical coordinates have been handled directly by the underlying numerical models as input data and not as configuration data, therefore this kind of information is not "seen" in the GUI itself besides providing input files or web services end points. As we've noticed that snippets of information (such as a few time ranges) are more obviously visible when provided as configuration data than as input data (somehow hidden within an input file), we might add more support for other forms of geospatial information in the future (but this will also have to be a compromise between the amount of information contained in the configuration file and the readability of the said file).

> 3) It is important to clarify how the tool is compatible with the standards and specifics that are using by the community, such as these from the OGC. Otherwise, it will be a closed system that is hard to adopt by others.

So far we have adopted several standards (such as date and time representations) but as Inishell is only focused on configuration data and mostly handles quite basic types, there is little overlap with standards such as those from the OGC (on the other hand, there is much more overlap with OGC standards in the underlying numerical models). In the future, depending on the needs that may arise, Inishell might gain more complex data types that would be more within the scope of standards such as OGC.

> 4) The paper has been written like a technical document instead of a research article. It has been focused on presenting the specs of the tool, but not much on the justification of the approach and the scientific contributions the tool can bring to the community, especially on reusing the scientific models.

Following the comments of reviewer #1, we have expanded the introduction section and reframed Inishell within the context of declarative User Interface Model (UIM, see (DaSilva, 2000) and (Díaz et al., 2020)). In this context, Inishell shows that within a niche application such as the configuration of scientific numerical models, it is both

easy and efficient to rely on this approach in order to provide a GUI to the end users. We have also expanded the description of our approach to include the strategy to deal with configuration data within the scientific numerical model. We hope to show to numerical mode developers the benefits this approach can bring: less support requests, low maintenance needs, high flexibility to accommodate new configuration options and better reproducibility of the model results. We have also added a discussion part to the paper in order to reflect on our practical experience from the point of view of model developers and maintainers.

[DaSilva, 2000] Da Silva, Paulo Pinheiro. "User interface declarative models and development environments: A survey." International Workshop on Design, Specification, and Verification of Interactive Systems. Springer, Berlin, Heidelberg, 2000.

[Díaz et al., 2020] Díaz, Eduardo, et al. "An empirical study of rules for mapping BPMN models to graphical user interfaces." Multimedia Tools and Applications (2020): 1-36.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-339, 2020.