Answer to Anonymous Referee #1

>1. There are plenty of widget toolkits. Why do you use Qt? Is this framework suitable to other widget toolkits/programming-languages?

We have chosen Qt for its multi platform abilities that go beyond the widgets, including some low level functions that allow us not to have to care about low level platform specifics while offering a good integration within each supported platform. Moreover, it is object oriented which we found helpful for our tool and it is open source which means that are source code can be recompiled by all our users. Qt is also one of the oldest and best maintained toolkits to write generic (C++) GUIs and a de facto standard in this domain.

>2. Is this framework general purpose?

Yes, Qt is general purpose (it offers both GUI widgets and non graphical functions to fully abstract the platform differences).

> 3. You use too much the word "model". I think that you use the word "model" also to point out the "template of the GUI layout". Moreover, the "numerical model" is an entity where the association between one widget and one parameter is accomplished (?). On the contrary, Must the association be "handmade"?

All occurrences of "model" in our manuscript refer to a numerical model, that is a computer program that simulates physical processes such as the Snowpack snow cover model, the WRF (Weather Research and Forecasting) model, the ECMWF models or any Numerical Weather Forecast model.

We have significantly rewritten the introduction and the "Principles" section because it appears that they severely lacked clarity and we have also introduced a discussion of our approach from the point of view of data models since it should be an interesting addition. A new figure has been created to show the general principles of numerical models from the user's point of view and the figure showing the principles of Inishell has been slightly expanded for clarity.

>4. The information model is not defined in your work. For e.g., you could use an ERD scheme to define the structure of the information in your work. The UML class diagram could show the code structure and the relationships among the classes. (See the first suggested reference at point 8. to see how to define an information model).

This is also a consequence of our lack of clarity: we do not define a data model in our work because Inishell consumes a data model provided as an XML file to populate a GUI where a user can enter configuration inputs (for example the spatial resolution or the timestep) for a numerical model (for example Snowpack) that will be stored as an INI file. This INI file is then read by a numerical model (for example Snowpack) to produce some outputs. So the numerical model assumes a given data model for its configuration data that is then formerly laid out in the XML file (to be written by its developers, not by Inishell's developers). Therefore, we have the following components:

1) a numerical model written by a third party (ex. Snowpack) that can read its configuration from an INI file;
2) an XML file that describes the data model of the configuration inputs of the numerical model (to be written by a third party, for example Snowpack's developers);
3) a GUI populated by Inishell based on the provided XML file;
4) configuration inputs provided by the numerical model user by the mean of the Inishell generated GUI and stored as an INI file by Inishell.

This has now been described in the rewritten sections.

>5. There are two aspects in a GUI: the layout and the callback functions. I cannot see the latter aspect. The entire system seems to be a "semi-automatic form filler" because there are no functionalities to be evoked.

There are not really any callback functions besides implementing what has been specified in the provided XML file: the GUI widgets receive the user inputs, validate them based on the rules defined in the XML file and write them down into an INI file. In this sense, Inishell is an automatic form generator with a few auxiliary features (such as generating a preview of the INI file or loading an existing INI file). Inishell does not "process" data or produce any outputs other than the conversion of the user provided inputs in the GUI into an INI file with its syntax.

>6. The system is not completely automatic: it seems an interface to generate another interface. Indeed, the GUI layout should be inferred from the parameters. Given a specific domain, there should be a module aimed to perform the creation of the layout on the basis of the parameters. This module should be able to create the XML related to the layout. It shouldn’t be necessary an artificial intelligence approach: just a data management approach could be sufficient. For e.g., a table containing the associations: INI ENTRY station1=xxx.smet -> ASSOCIATION TABLE station=filename -> AUTOMATICALLY GENERATED XML CODE: <parameter key="STATION#" type="filename"

In order to remain generic, there is no domain-specific knowledge in Inishell. Each numerical model might come with its own parameter names and meaning. For example, even within a specific numerical model the same parameter name might have different meanings depending on the context: a configuration entry such as STATION1 might receive a filename or a database key or even geographic coordinates within the same numerical model, depending on the context! This means that there is not enough information to generate a relevant GUI only with the parameter names. Moreover, the INI file is quite user friendly (because of it is only very loosely structured) but it can not provide any semantic information (such as data types, ranges, regular expressions for validation or dependencies). Since the INI file will be edited or even created from scratch by the users, it is also not possible to store there any help text or other information. Therefore, the XML file contains all the semantic information for a specific numerical model (and it is often the only place where such information is formally expressed besides partly in an implicit, diffuse way within the numerical model source code, as assumptions or checks about the configuration data).

>7. A validation section is absent. I do not pretend a user experience study, but a section where the use of your system is easier than the "traditional" way.

More details have been added in the conclusion about our experience with and without Inishell from the point of view of our users and from the point of view of support requests.

>8. The paper is lack of recent bibliography. The following two papers are interesting but you should include other papers recently published. Orazio Gambino, Leonardo Rundo, Vincenzo Cannella, Salvatore Vitabile, Roberto Pirrone, A framework for data-driven adaptive GUI generation based on DI- C2 COM,Journal of Biomedical Informatics, Volume 88,2018,Pages 37-52,ISSN 1532- 0464, https://doi.org/10.1016/j.jbi.2018.10.009. Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI design image to GUI skeleton: a neural machine translator to bootstrap mobile GUI implementation. In Proceedings of the 40th International Conference on Software Engineering (ICSE ’18). Association for Computing Machinery, New York, NY, USA, 665–676. DOI:https://doi.org/10.1145/3180155.3180240

Although there are some data driven parts in Inishell, it remains quite basic: some keys are shown or hidden depending on other keys (just acting as booleans). The second paper is quite interesting but Inishall actually does somehow the opposite: without any hints regarding how the GUI should look like, Inishell populates its GUI as some sort of a regular grid of widgets.

>From the introduction: The paper concerns a framework aimed at the automatic generation of GUI by means of an XML description and an INI file containing structured information about the parameters having a relationship with each widget.

This is what we must better describe: The XML file contains the structured information that allows populating a GUI to gather user inputs and write an unstructured INI file containing these user inputs. Inishell is therefore "only" a data entry software for configuration files with no domain specific knowledge of its own, instead relying on an XML file to provide it. The INI file is an output of Inishell (although it can re-read an existing one in order to edit it) and an input for the numerical model (such as Snowpack).

Answer to referee #1

> By the way, a manuscript with only 10 references, where 3 over 10 are auto-citations, is not permissible for a journal. Also for this reason I suggested including other works in the references, like the ones I mentioned in the previous review, even if they belong to other domains. Substantially, there isn't a good background literature where other methodologies are applied to solve similar problems.

In the sections that we have rewritten we have also added references belonging to other domains. We have now reframed Inishell within the context of declarative User Interface Model (UIM). Here we see Inishell as a simplification of these more generic works thanks to the much more limited complexity of handling the configuration data of scientific numerical models (reduced diversity of inputs, much simpler dependencies between input fields and much simpler workflows). We are grateful for the feedback that has helped us to step out of our domain and to present Inishell in a wider context that will make the paper more valuable to a broader audience.

We have now significantly expanded the bibliography to also include recent works aimed at similar simplifications.

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 geospatial 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.