Answer to Anonymous Referee #1
>No information model in any form is presented. Just a graphical abstract of the system is presented in the manuscript without the description of the architecture with classic software engineer representations.

Two UML diagrams have been added: an activity diagram to show both the expected user workflow and the actions that are triggered within Inishell and a simplified class diagram to give an overview of how Inishell is organized internally and to show how the internal organization reflects the individual graphical elements that are presented later in the same section.

>Moreover, the bibliography has been updated without the suggested papers that I mentioned in the 1st round of review, especially the one regarding adaptive GUI generation based on DICOM

This reference has been added.

Answer to Anonymous Referee #2
No further answer necessary

Answer to Anonymous Referee #3
>There is a significant push in the earth science community for reproducible science, including the configuration of numerical models. […] which cleanly links with the idea of reproducible science and codifying the steps taken during the workflow of setting up a model. However, this point is never fully described. How are the changelogs produced? Where are they stored? How does the user interact with them?

There is no special mechanism in Inishell to produce or manipulate changelogs, but by providing benefits to the end user as well as to the developers that as a side effect promote good practice regarding reproducibility, Inishell can be part of a wider system that encourages all actors to produce reproducible numerical simulations. A new section on this topic with new bibliographic references has been added.

>Throughout the revised manuscript there are anecdotal comments that lack sufficient context related to end-users interacting with the models and configurations. I am familiar with this groups SNOWPACK model and their ongoing work with practitioners in the avalanche hazard forecasting community. Thus, when they make statements, for example, that users struggling to use the command line terminal, it is unclear who had this issue – are these practitioners or scientific users?

The authors have tried to clarify these points. Basically, both kinds of end users are often not very technically inclined and therefore struggle with the command line terminal. This is most probably the result of a feedback loop: as a model get easier to use, they attract users who might have previously been afraid of the complexity of the model and therefore bring even more usability requests and requirements.

>I therefore think that the authors need a bit of polish to pull together the context they added in revision two to better link the concepts of reproducible science, end-users who are domain experts but not model experts, and what types of GUI features are required to support these use cases.

The authors have added much more content on these topics.

> Having a GUI that facilitates reproducible research and ensures domain experts have access to appropriate models seems to be a useful contribution.

These points come as side effects of having an up-to-date GUI for the models. The main goal of Inishell is not to facilitate reproducible research nor to provide appropriate models to domain experts, but to provide a GUI to models that previously did not have one at all.

>[0] "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 feedback from our own experience with user support has been added. A few references have been added. Basically, this comes down to numerical model with GUI versus numerical model without any GUI at all.

>[1] "Substantially, there isn't a good background literature where other methodologies are applied to solve similar problems."

The literature that was already cited in the second version of the manuscript does cover this: there are two well known alternatives in order to provide a GUI for a software: an ad-hoc GUI (as carefully hand-designed) or a system to generate a GUI from an abstract description of the GUI (Declarative User Interface Model). None of these approaches is suitable for numerical models developed by small teams (between 1 FTE and less than 2-3 FTE, models such as the GeoTop hydrological model by Endrizzi et al.[1], the AMUNDSEN alpine snow cover model by Strasser et al.[2], the CROCUS snow cover model by Vionnet et al.[3], the radiation transfer model SMRT by Picard et al.[4], as well as the snow cover and snow hydrology models developed or maintained by the authors such as Snowpack, Alpine3D, AlpineFlow and MeteoIO). The authors have deliberately chosen not to refer to specific models in the manuscript in order to keep it generic enough (models outside of the cryospheric sciences that are developed by small teams have the same shortage of manpower).

>There is a substantial body of research in the Human-Computer Interaction (HCI) literature on how GUIs are designed, A/B tests to determine optimal layouts, etc. The authors note that "[t]he focus is also not on the visual appearance of the GUI but on the data that has to be provided by the end user." Certainly, there must be some method to how the UI is laid out. Is there a novel way in doing so to optimize interaction with numerical models? This links with [0] but also [1] where presenting a UI in a manner to help domain experts interact with models requires some thought in how the UI is designed.

The core concept of Inishell is to provide a GUI with very low investment for any model that had none (not only for the Snowpack model but any other arbitrary model that could read its configuration from an INI file. Actually, some other, unrelated models have implemented their GUI in Inishell). Therefore no effort has been made in order to optimize the interaction with the numerical model or the GUI layout as this would be very much too specific for one model or would require much greater complexity (this is feasible with Declarative User Interface Model but comes at the expense of tremendously higher complexity and skill thus making it impractical for the numerical models that Inishell wants to help). The new material in the manuscript should clarify the

1 Endrizzi, S., Gruber, S., Dall'Amico, M., & Rigon, R. (2014). GEOtop 2.0: sim-872 ulating the combined energy and water balance at and below the land surface 873 accounting for soil freezing, snow cover and terrain effects, Geoscientific Model 874 Development, 7 (6), 2831–2857, doi: 10.5194.

2 Strasser, U., Bernhardt, M., Weber, M., Liston, G. E., & Mauser, W. (2008). Is snow sublimation important in the alpine water balance?. The Cryosphere, *2*(1), 53-66.

3 Or the more recent Vionnet, V., Brun, E., Morin, S., Boone, A., Faroux, S., Le Moigne, P., Martin, E., and Willemet, J.-M.: The detailed snowpack scheme Crocus and its implementation in SURFEX v7.2, Geosci. Model Dev., 5, 773–791, https://doi.org/10.5194/gmd-5-773-2012, 2012.

4 Picard, G., Brucker, L., Fily, M., Gallée, H., and Krinner, G.: Modeling time series of microwave brightness temperature in Antarctica, J. Glaciol., 55, 537–551, 2009.

basic building blocks that are used by Inishell which then most of the time simply stacks in vertical grid the (label, input widget, help text) triplet. This is not fancy but remains very simple for the model developer to handle and is still a welcomed improvement over the command line interface (that is often problematic for the end users) and the documentation that is not read by the end users (see references in the introduction).

> It seems the authors attempted to address this in Table 1, however it is not clear to me how this table is supposed to be read – top to bottom? Left to right? I believe that the authors should tighten section 3.2 to make it more clear how the various levels in the hierarchical design works and exactly how the XML options are mapped to these internal data models for generating the UI.
The table was not really useful so it has been removed. Instead, two new UML diagrams should better show how Inishell works alongside with new text. The general principle of mapping XML elements to GUI elements is explained in the rewritten "General architecture" section and the following sections.

> I would like a more detailed description of how basic range checking and regex checking are done – is it as the user types or when the INI file is created?

This is now explicitly described in the new "Reproducible science considerations" section as these checks must be done twice: both in the GUI and in the numerical model itself (so the numerical model can still validate its configuration parameters even when provided by users who do not rely on Inishell to create/edit their configuration files. It still makes sense to perform it anyway in the GUI as this allows the GUI users to quickly spot obvious configuration errors without having to run through the numerical model).

>Lastly, and this does also relate to point 3 (generalization) below, is that describing the internal data model and class relationship as suggested by the reviewer would help demonstrate applicability to a wider gamut of toolsets. Most models do not use INI configuration files. Thus, how easily can a user swap the INI writer for a, for example, a JSON writer? These internal data model descriptions and class layout diagrams would be helpful to understand this.

This has been clarified by providing a class diagram and a description of the class diagram, including where the data is stored (as key/value pairs). Basically, any format that is based on key/value pairs with some sort of name spaces could be easily implemented. Obviously, instead of calling the data storage SectionList through the INIParser, it would make sense to call it through a generic interface class. As a side note, the syntax highlighting in the PreviewEditor would also have to be expanded in order to cover a new syntax (practically, creating another instance with different rules as well as a mechanism to determine which instance should be used based on the configuration file format that has to be generated).

>I downloaded and compiled IniShell. In general, it looks nice. However, there are non-standard UI decisions that, as a first-time user, were sufficiently quirky and non-standard that I was confused. I had to check the mouse-hover tooltip to understand what it did. For example, the save icon is not the standard floppy-disk icon. The open icon looks like the standard 'new file' icon instead of the more standard "open folder". The preview icon is a printer (I assumed this printed?).

First, thank you for your involvement in this review! The handling of icons is a little special: it is based on the XDG standard for icons themes (https://specifications.freedesktop.org/icon-naming-spec/icon-naming-spec-latest.html). None of the icons have been designed by ourselves nor are they selected by ourselves or referenced by filename in the source code. Instead, the source code calls standard names that explicitly describe the action or place that the icons should represent (for example, "document-save-as") and then the underlying Desktop Environment should select from its

currently enabled icon theme which icon to provide. This visually integrates Inishell better into the user's environment as the icons are consistent with the rest of the environment (so light or black themes are supported, 4K screens should receive the proper icons, etc) while being fully transparent to Inishell (and sparing us graphical design).

However, the XDG standard is not supported on Windows or MacOS. Therefore, a fallback had to be implemented. This consists of providing pre-packaged icons if the call to get the icons from the system failed. For better visual integration, two fallback icon themes have been selected (both XDG compliant), one for Windows and one for MacOS. Another requirement for these fallback themes is the license: they must be redistributable (so the native icons on each platform could not be used). We therefore selected Open Source icon themes that are popular and usually well considered. The icon theme for MacOS replaced the legacy floppy icon for file save operations by another visual metaphor. This has turned out not to be so understandable, and after looking for standard ways to represent "save file" on MacOS and finding nothing more or less standardized (only a few non-native applications provide a "save file" icon in a theme that definitely looks non-native), we have now reverted to using the same icon theme for both MacOS and Windows (this is an icon theme that uses the floppy metaphor). However, on the longer term the floppy metaphor will loose its relevance...

>In the left-hand side panel, the example INI files are listed twice (seemingly). These types of GUI design decisions and departures from well-known paradigms have a 'scientist UI' feel to it versus "lovingly hand-crafted" by a UI expert feel. It is my opinion "focus is also not on the visual appearance of the GUI" requires more justification.

The XML files are listed twice because Inishell looks for XML files in multiple locations. Inishell must support multiple versions of the XML files, matching multiple version of any given model (for example, a stable release and a development version, having small differences as new modules get added or new configuration parameters). In order to help the user select which one to use, the XML files are grouped together by the paths where they have been found and the path itself is shown above each group. On mouse-over, the full path and filename are given in a tooltip.

Moreover, not being able to show any XML files (or not showing the ones that the end user expects to see) at start is a much more serious issue for the end user than showing too many. Since we want to support both users of prepackaged versions of Inishell and users recompiling Inishell themselves, we need to include many search locations (this is made even worse by the fact that each compiler tends to setup a different directory structure where it builds the binaries, which changes the relative path between the executable and its XML files and we don't want to impose a specific compiler on the users, so we need to support all the build structures variations). By default on MacOS, because the example XML files might or might not be part of the build directory (depending on which build target has been called and if the packaging could be performed or not), the example XML files might appear once (only the XML files present in the source directory) or twice (adding a copy of the XML files as resources in the .app directory structure that can be packaged into a dmg or just copied to another location while deleting the source directory or copied to another computer altogether). Of course, users relying on prepackaged binaries won't see this duplication of XML files as they won't have the source directory.

Detecting and avoiding the duplication mentioned in this comment could be avoided but at the cost of increased complexity in the build system (more detection of the various paths in use by the compiler, more exchange with the c++ implementation to skip some of these paths after checking them against the compiler-specific structure in the build directory) or by limiting support for source compilation by end users (our experience with Snowpack, MeteoIO and Alpine3D also show that supporting a relatively polished experience both using precompiled binaries and source code

compilation is a challenge with respect to resources and paths. Restricting ourselves to one or the other would make everything much simpler).

> It is unclear to me how easily it would be to implement this in another model or if INI conversion tools can handle this. I would like to see an explicit table of what was extended to clearly explain what would need to be added to a downstream consuming model to take advantage.

The extensions that we have implemented are now more explicitly defined in the “Supported INI file syntax” appendix. This section has been moved to an appendix in order to prevent interrupting the flow in the main document while still documenting what is supported. The other models that we know of that rely on an INI kind of syntax don’t rely on these extensions and therefore don’t need to change anything in order to be supported by Inishell. If a consuming model would come with its own extensions, then either this would have to be implemented in Inishell or converted to one of the extensions supported by Inishell in the consuming model itself.

> 1) improve the literature summary/gap and discussion to position this manuscript around reproducible research and expert-user/non-modeller usage to clearly meet GMD contribution guidelines;

A new section on the topic of reproducible research has been created with some associated literature. The expert-user/non-modeller topic is now discussed in the introduction (as we saw little difference between the two groups besides a few individuals).

>2) more clearly articulate how the internal IniShell data model translates the input XML file into a GUI and could allow for generalization to different configuration formats;

The rewritten “General architecture” has been expanded to cover this topic alongside a new UML (simplified) class diagram that is also commented along these lines.

> and 3) to clearly articulate the INI spec extension to ensure generalizability.

This is covered in the new “Supported INI file syntax” appendix.

Answer to Anonymous Referee #4

>The paper describes a solution to the problem, but fails to clearly identify the problem and the scope of the paper. Much relevant information comes implicitly throughout the paper, but the paper as such would benefit from a clearer structure focusing on the problem/use case, methodology/approach/architecture, implementation, discussion and conclusion/summary. As it reads now it is difficult to follow the authors through this process. The impact of the paper would be greater if this is clarified. In the methodology/approach/architecture, usage of interaction and sequence diagrams would be beneficial along with a clear description of the data model supporting this interaction. The illustrations that are provided are nice, but needs more support.

The paper has been restructured in order to bring more clarity: the introduction now has a new subsection “Graphical User Interfaces”, a “methodology” section contains a “requirements” subsection (with an activity diagram) as well as a “General principles of Inishell” section and a “Reproducible science” subsection. A new “Implementation” section contains the overview of the interface, a description of the general implementation (with a class diagram and the description of how the data that is parsed leads to the creation of widgets in the application) followed by a mode detailed description of the XML / widgets / INI mappings grouped by types of elements.

>Section 1: I would like to see the introduction split to a more general background on generic GUI and a separate section on problem description which also explains the nature of numerical models and how they are configured and operated in more detail. As it is now, the sub section on requirements appears to lack justification. Additional references on the problem description would be useful.

This split has been implemented in the introduction section (see above). More details about the category of users are also given alongside many more references, including on the benefits and shortcomings of GUI and some examples of numerical models users.

>Section 1.2 – last paragraph: this is a very bold statement which I can't see is justified in the current text. Either more references or a better problem description and analysis would be necessary.

This has been rephrased and is now part of the new "Reproducible Science" section. This also contains new references.

>Section 2: Should be renamed to methodology or similar preceeded by a proper problem description and justification of selected approach.

This has been done. The justification of the selected approach is found in section 1.3.

>Section 2: In order to justify inishell I miss a more thorough problem description and analysis of potential solutions. There are good reasons to choose inishell, but it doesn't mean that there aren't alternatives that appears more modern.

The alternatives are discussed in section 1.3 "Graphical User Interfaces (GUI)". Basically, there are traditional, manually designed GUI that involve a significant time (and skill) investment or automatically generated GUI that require very deep knowledge (as well as high complexity). Inishell aims at being a middle way that remains easy yet requires very little time investment (the limitations being that the GUI is quite primitive).

>Referring to INI as the informal standard is not justified, references are needed or a more thorough analysis of approaches.

We have slightly rephrased this sentence: INI is not "the" standard, but one standard although it is informal (i.e. not formally defined). It is used (often as derivative forms such as under the name "namelist") for multiple numerical models in our community, such as the Crocus environment for weather forecasting at MeteoFrance (cited above), Amundsen (cited above although a new version is coming that now uses YAML), GeoTop (cited above), WRF (Skamarock et al.[5]), our models Alpine3D, Snowpack, MeteoIO, AlpineFlow among others.

>Section 2.1 – paragraph 7: Referring to the XML file as the data model for the configuration files won't do. A more generic approach would be necessary using e.g. UML or similar, representing the various sections identified in the GUI presented later.

A new class diagram has been created and commented to show how the GUI is built (section 3.2, figure 5).

5 Skamarock, W. C., Klemp, J. B., Dudhia, J., Gill, D. O., Barker, D. M., Wang, W., & Powers, J. G. (2005). A description of the advanced research WRF version 2. National Center For Atmospheric Research Boulder Co Mesoscale and Microscale Meteorology Div.

>Section 2.2 – first paragraph: What is meant by "...numerical model profile and specific configuration profile...". I am not sure a "specific configuration file" is the same as "a model with loaded settings ready to run". A better explanation is required.

This has been rephrased in order to be clearer.

>Section 2.2. figure 3: The GUI representation is jumped without a proper model describing the purpose of the different sections in the GUI. A logical workflow, potentially represented through an interaction diagram or similar would be useful to understand the context.

This figure (now figure 4) sits in section 3.1 that describes it in details (more details have been added as well as linked to the new UML activity diagram that is now figure 3). As it is a floating figure, it can be come just before the descriptive text, in the middle of it or just before. If this is an issue, this should be addressed at proof reading (as the two columns layout will change the positioning of figures anyway).

>Section 2.2 – first paragraph: To my knowledge MeteoIO isn't a numerical model but rather a pre- processor of input data. Clarification would be welcome.

The mentions of MeteoIO have been rephrased and supported by a new reference as for a few years many users would rely on a provided example code to call MeteoIO standalone to prepare data files. For the last 18 months, this has been made more official by providing a proper executable for data preparation as part of the source tree (and it is now compiled by default). In this respect, MeteoIO's online documentation is obsolete when it refers to MeteoIO being "only" a library.

>Section 3: As far as I can see both section 2 and 3 are implementation. A better separation of content would be welcome. Again referring to INI as the informal standard for configuration files has to be justified. In general I would claim that YAML is at least as widespread as INI, but there can be differences between communities that for legacy reasons stick with INI. References would help in this part.

Sections 2 and 3 have been heavily reworked and several paragraphs have been moved to a different location. Regarding the INI standard, see our reply above. So far, we wanted to avoid being too specific by giving model names that rely on INI files in order to avoid being too "cryospheric sciences community" focused.

>Section 3.2: I am sure I would agree on this being the architecture. As mentioned earlier, to simplify reading a general introduction followed by a proper problem description and analysis, then a section on methodology or approach that contains the architecture, leading up to the implementation and a discussion of this would simplify the reading of the paper. As sections 2 and 3 are somewhat interleaved and not clearly defined scope wise a restructuring would be welcome.

The paper has been restructured accordingly.

> Section 3.3: Not sure I understand what is meant by semantic names, seems repetitive.

This has been replaced by "symbolic names". These names are not directly color names, but function names (such as "warning", "error"…) so this leads to more consistency and can easily be styled differently without having to change the colors everywhere.

>Section 3.7: The work flow description would benefit from interaction or sequence diagrams. It is not clear if GUI and model has to run in the same environment or if remote control is possible, This

would be interesting to know. Also is this more than a configuration system? Is it also a execution environment?

The supported execution environment has been described in the “Workflows” section: local execution by spawning a new process, no support for batch schedulers.

> Section 3.8: I can’t see that a numerical model (nor multiple) is loaded into INIShell. Rephrasing would be welcome to clarify what a numerical model is.

This has been rephrased.

>Section 4 – second paragraph: For the time being running numerical models is commonly done in HPC than cloud environments, but could depend on the community and type of model. Referring to the cloud without addressing HPC would be oversimplifying the complexity of numerical models. Also referring to my comment on section 3.7, a clarification of execution environment would be beneficial up front, not indirectly addressed in discussion.

This has been rephrased as “directly on the web” for clarity. The supported execution environment has been described in the “Workflows” section: local execution by spawning a new process, no support for batch schedulers.

>Section 4 – third paragraph: file formats for what?

This has been clarified in the text (file formats for the configuration files, namely the INI files).

>Section 5: I can’t see that the statement that a GUI gives better control over the configuration than a text based solution. Justification would be required, also in the discussion. And the justification should also refer back to the problem description that is the foundation for the paper.

This justification has been brought in the new section 1.3 “Graphical User Interfaces” with two references that are based on (technically minded) user surveys. The better control over the configuration is a consequence of the reduced learning curve and improved educational value (even for quite advanced users). The link to section 1.3 has explicitly been made.