Preparing Horizon Europe proposals in LaTeX with heria

Tristan Miller

Abstract

This article introduces heria, a LaTeX class to format funding proposals for the European Commission's Horizon Europe program. It provides a basic summary of the class's use; compares it to existing packages for funding proposals; discusses its motivations, design decisions, and limitations; and reports on its real-world use and plans for future development. Besides providing prospective Horizon Europe applicants with an overview of the class, this article may give prospective developers and users of classes for other proposal types some idea of the work involved and the potential pitfalls.

1 Introduction

Horizon Europe is a seven-year, €95.5 billion initiative of the European Commission (EC) that is intended to fund research and innovation projects in the European Union and its wider network of global partners. The EC earmarks portions of the total budget according to various topics and action types, issues calls for proposals of projects supporting those topics and action types, and then disburses the funds to applicants according to a competitive evaluation process. The types of projects solicited often require a large consortium of partners — potentially dozens — and the calls prescribe a specific, intricate structure for the proposals, which can run to hundreds of pages. While the EC does not require applicants to use any particular content authoring tool, the only templates it distributes are in Rich Text Format (RTF) — hardly the most convenient format for multiple authors to collaborate on producing a lengthy, heavily (cross-)referenced technical text.

This article introduces heria, a LaTeX class to format proposals for the Research and Innovation Actions (RIA) and Innovation Actions (IA) of Horizon Europe. Using heria and a networked source control system or collaborative online LaTeX editor, it becomes easier for a dozen or more authors to jointly produce an elegant, internally consistent proposal conforming to the EC's requirements. Unlike the default RTF template, the heria class manages the numbering of and references to project elements (participants, work packages, etc.) as they are added and removed, and programmatically regenerates and resums the requisite data tables (for staff effort, project costs, etc.). This helps ensure that data changed in

doi.org/10.47397/tb/45-1/tb139miller-horizon

one part of the proposal remains consistent with explicit and implicit references to it elsewhere in the proposal. The class also preserves the instructions from the original template, but allows users to toggle their visibility, either individually or *en masse*, so that instructions can be hidden once they are fulfilled, or once the proposal is ready to submit.

Besides providing a very basic summary of how `heria` is meant to be used, this article compares it to existing packages for funding proposals; discusses its motivation, design decisions, and limitations; and reports on its real-world use and plans for future development. This material should help prospective Horizon Europe applicants decide whether it makes sense to use `heria` for their proposal; perhaps equally importantly, it may inspire others to develop and publish their own packages for other types of funding proposals (whether for Horizon Europe or some other funding scheme) and it may give them some idea of the work involved and the potential pitfalls.

2 Previous proposal packages

Perhaps surprisingly, given the TEX ecosystem's popularity among academics and technologists, CTAN boasts only a handful of packages for typesetting research funding proposals. The `nih` package [2], last updated in 2005, provides a LATEX class to format grant applications to the US National Institutes of Health (NIH). The more recent `grant` package [7], last modified in 2019, also handles LATEX proposals for NIH, as well as five further American agencies, including the National Science Foundation and the Defense Advanced Research Projects Agency. Neither package provides much in the way of formal documentation, though `nih` at least comes with substantial example documents. The `mynsfc` package [10], last changed in 2020, provides a XƎLATEX class for proposals to the National Natural Science Foundation of China (NSFC). As with the previous two packages, there is little or no technical documentation concerning how to use the class; however, the package does reproduce the NSFC's instructions concerning the content and formatting of the proposal.

The `proposal` [8] and `h2020proposal` [6] packages are seemingly the only ones until now that specifically target the preparation of European grant proposals. The most recent CTAN release of `proposal` dates to 2016, but development has continued on the project's GitHub repository at `github.com/KWARC/LaTeX-proposal`. The package includes LATEX classes for proposals to the German Research Foundation and the EC's Framework Programme 7 (FP7), along with documentation and examples. The `h2020proposal` package, dating to 2015, contains classes for RIA proposals in the EC's Horizon 2020 program, the predecessor of Horizon Europe. The templates are set up for use with LATEX but contain guidance on adapting them for use with XƎLATEX. As with `proposal`, `h2020proposal` includes documentation and examples, and like `mynsfc`, the templates helpfully reproduce the funding agency's content- and formatting-related guidelines.

Although these last two packages are fairly elaborate, and (according to their authors) have even been used to prepare real proposals for submission to the EC, they share two significant shortcomings. First, neither package supports the proposal format used by Horizon Europe, the current EC framework program that runs from 2021 until 2027. (FP7 ran from 2007 to 2013, and Horizon 2020 from 2014 to 2020.) Second, both packages are admittedly incomplete: the documentation for `proposal` indicates that the package is "relatively early in its development", and the documentation for `h2020proposal` warns would-be users that it is "still in a beta-testing stage" and should not be distributed.

3 Motivation and design decisions

The principal motivation for producing a new proposal class, rather than extending an existing one, was to support applications to the current Horizon Europe RIA and IA actions. Although `proposal` and `h2020proposal` have many good ideas in terms of their implementation, the historic proposal types they support differ so much in terms of form and content from current ones that it would be very challenging to extend or even adapt these packages. The `heria` class was therefore written entirely from scratch, though it does draw some inspiration from the interface of `h2020proposal`.

The Unix philosophy was another influence on `heria`, or more specifically its maxim, "Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new 'features'." [9] The two aforementioned packages include some extra bells and whistles that are not strictly necessary for preparing a proposal. Besides supporting both European and German proposals, `proposal` includes functionality for preparing grant agreements and final project reports, which are significantly different in structure and formatting. And both packages provide a mechanism for generating Gantt charts for use in the proposal; while the official Horizon Europe application instructions require applicants to indicate the "timing of the different work packages and their components", it does not require this to be in the form of a Gantt chart. In any case, in the present author's experience, how best to style and structure a Gantt chart varies greatly from project to project, and so having heria provide its own implementation would pose a number of challenges. Either the implementation could be a simple one targeting the lowest common denominator, which many users would find limiting, or it could allow for great versatility in the design of the chart, in which case it could end up as little more than a wrapper for the pgfgantt package [12]. Since pgfgantt already exists and is fairly simple to use, heria assumes that proposal authors will simply use that if they want a Gantt chart.

It was also important that heria be easy to use. Like proposal and h2020proposal, the macros it provides are fairly simple and often store information for use later in the proposal, and like h2020proposal, it helpfully exposes the official application instructions to the user. With seven pages of prose, the documentation for heria is comparable in breadth and depth to that of h2020proposal; direct comparison with proposal is not practical owing to the latter's goal of supporting more funding schemes and non-proposal document types.

A final design decision that was of great importance to the developer was that the package should allow users to produce a proposal using only free software [4]. The heria package is therefore released under the terms of the LaTeX Project Public License. The packages it depends on, as well as LaTeX and TeX themselves, are also available under various licenses permitting free use, modification, and (re)distribution. While proposal co-authors may choose to collaboratively edit their heria-based proposal using a proprietary online service such as Overleaf, they could alternatively use an entirely free authoring pipeline with a source control system such as Git.

4 User interface

The intention of this section is not to recapitulate the complete package documentation, but rather to give a very general overview of heria's interface and features. This serves as an introduction to prospective users and as context for some of the observations and discussions found later in this article.

The heria package consists of a class file (heria.cls), a set of LaTeX files containing the application instructions, a skeleton proposal (heria-example.tex), and the package documentation (heria.pdf). Since the official Horizon Europe proposal template requires proposals to follow a fairly strict and detailed structure, the best way of starting a new proposal is to make a copy of the skeleton proposal and then adapt it by replacing its dummy data and supplying any missing information.

As in the official template, lengthy application instructions are interspersed throughout the skeleton proposal. Most of these instructions are emitted via an `\heinstructions` macro at the appropriate place in the proposal document; the argument to this macro specifies one of the aforementioned files containing the instruction text. Other instructions take the form of recommended page limits printed next to section headings; these limits are specified via an optional argument to the `\section`, `\subsection`, and `\subsubsection` macros. Instructions are printed only when the showinstructions option is passed to `\documentclass`; instructions can alternatively be omitted on a case-by-case basis by removing or commenting out the corresponding `\heinstructions` macro (or option to the section heading macro, as the case may be).

The class, which is derived from the standard LaTeX article class, takes care of setting the fonts, margins, etc. as mandated by the official template, and redefines some common commands (`\maketitle`, `\section`, etc.) to produce titles and headings in the prescribed format. This includes the so-called "tags" (cryptic identifiers such as "#§CON-MET-CM§#") that the official template places around certain section headings and warns applicants not to move, remove, or change in any way.

For many parts of the proposal, applicants can simply provide free-form text, along with whatever lists, tables, figures, etc. they think necessary. For other parts, the official template requires applicants to provide information in a fixed format, usually corresponding to one or more tables. Often these tables, and/or the rows or columns within them, must be printed in a particular order that is determined by information entered elsewhere in the proposal. For example, the table summarizing the staff effort has one row for each participant in the project, and these rows must be arranged in the same order as in the earlier table listing the participants; it also has one column for each work package in the project, and these columns must be arranged in the same order as the earlier table listing the work packages. Besides this, the staff effort table needs to include a final row that sums the numbers in each column, and a final column that sums the numbers in each row. To obviate the need for users to tediously re-arrange and re-sum such tables every time a participant or work package is added, removed, or re-ordered, heria provides (a) macros such as `\participant` and `\workpackage` for defining participants, work packages, and other project data in such a way that heria remembers their original order and that users can explicitly reference them in subsequent macros and environments, and (b) macros such as `\makeparticipantstable` and `\makeworkpackagestable` that automatically generate the data tables in the correct order, and with automatically computed sums, on the basis of the order and content of the previous definitions.

Perhaps the most typographically complex part of the official template is the "summary canvas", a tableau of framed text boxes that is spread across the whole of one or two pages with landscape orientation. The `heria` class provides `summarycanvas` and `summarybox` environments for typesetting these boxes, as well as a `SidewaysFigure` environment that takes care of rotating the page in a way that preserves readability in PDF viewers.

5 Limitations and workarounds

The official RTF template has a number of oddities and limitations, and in adapting it to LaTeX it was necessary to decide, on a case-by-case basis, whether to preserve or work around them. The following points discuss some of these decisions:

Vague instructions. Some of the official instructions for filling in the tables admit of more than one possible interpretation. Perhaps the only such ambiguity with bearing on `heria`'s behaviour concerns how purchase costs in a given category are to be listed in the associated table; it is not clear whether applicants should itemize these costs across separate rows or combine them into a single row. The `heria` class takes the latter interpretation, though future releases might include support for itemized costs.

Tables that aren't tables. Many of the proposal elements that the official template refers to as "tables" are not, typographically speaking, tables, nor even what some dismissively refer to as "tableaux" [3]. For example, the template's Table 3.1b, headed "Work package description", is actually a 2×2 tableau for entering the work package number and title, followed by two separate, framed, full-width paragraph boxes for entering the work package's objectives and description of work. These three elements are to be repeated, all under the same "Table 3.1b" caption, for every work package. In a typical proposal, Table 3.1b will have content running across several pages, and almost none of it will be tabular. All such "tables" in the official template are therefore adapted into `heria` as LaTeX subsections, with custom macros for the user to provide the "table" data and another custom macro to finally output it in the prescribed format, using an appropriate combination of `tabular`-style environments and framed boxes.

No provision for floats. The RTF format has little or no support for floating objects; the official template is therefore written with the expectation that all the required information will be presented in a linear fashion and in the prescribed order. This can pose problems when there is not enough room remaining on a page to typeset a table; the table would normally have to start at the top of the next page, leaving wasted space on the previous page. The `heria` class solves most such problems with measures that allow tables to gracefully break across pages. The one exception is the template's landscape-oriented "summary canvas" that forms the sole content of Section 2.3. The page rotation precludes any possibility of beginning or ending the tableau on the same page as the preceding or following material, respectively. Here the skeleton proposal distributed with `heria` makes an arguably justifiable departure from the official template by putting the summary canvas in a floating figure, and then adding a one-line reference to it under the Section 2.3 heading. While this may not be strictly in line with the official template, it at least averts the danger of having the canvas introduced by a page that, except for the section heading, is nearly or entirely blank.

Other wasted space. Besides the lack of floats, there are other cases in which the official template doesn't make or even allow for efficient use of space. For example, the 2×2 work package tableau mentioned above seems altogether gratuitous, since the information it contains could easily have been combined into a single line. In other cases, bona-fide tables are given needlessly verbose column headings that introduce extra line breaks or steal horizontal space from the other columns. The `heria` version of the template preserves the original's gratuitous structural elements (since the presence of these may be subject to formal checks by the funding agency) but takes some liberty in slightly abbreviating, or at least hyphenating, some words in table headings.

Besides these imposed limitations, the package has a few shortcomings that are down mostly to the rushed initial development. For one, the class does very little specialized error checking on its input. Usually passing an invalid argument to one of its macros, or neglecting to provide data necessary to generate a table, will result in some sort of compilation error, though the diagnostic message emitted may be somewhat obscure. Another issue is that the class generally expects users to supply numeric data (for person-months, costs, etc.) as integers. Though decimal arguments to certain macros may be correctly interpreted, and some provision has been made for the class to use floating-point arithmetic when calculating sums, the code that emits numbers in generated tables cannot be relied upon to produce elegant output. Solving both these issues is on the agenda for future development.

6 Reflections and case study

The process of developing `heria` proved to be rewarding and frustrating in equal measures. On the one hand, it presented the developer with the motive and opportunity to apply and extend his LaTeX programming skills, and bestowed upon him an intimate familiarity with the application requirements for an active proposal submission (described below). On the other hand, having to reproduce and stay within the aesthetic and structural limitations of the official RTF template felt unduly constraining, particularly when those limitations seemed to be the product of questionable or even deleterious design decisions. This agony would perhaps have been felt less acutely had the proposal co-authors decided to forgo the use of LaTeX in favour of a less capable tool.

Even still, LaTeX itself was also at times a source of consternation when developing `heria`. The sort of high-level programming required to *easily* automate the management of proposal data and the generation of data tables is not well supported by LaTeX: many of the basic data structures necessary for these tasks, and the basic algorithms for accessing, sorting, and iterating over them, are either not present in the language, or require obscurely named and relatively under-documented LaTeX3 macros, or are implemented only in third-party packages that must first be discovered and then learned. Of course, these criticisms of LaTeX are hardly new (see, for example, [1, 5, 11]). In hindsight, it may have been a better idea to write the class in LuaTeX, even at the cost of having to learn it (and Lua itself) from scratch.

On the whole, the initial development of `heria` probably took about as much time as it would have taken the coordinator of a large word-processed proposal to manually resolve all the edit conflicts, formatting problems, bibliographical inconsistencies, and outdated cross-references introduced over the entire writing process. Anyone considering developing a LaTeX class for proposals for another funding program should therefore consider whether it makes sense to invest the effort; if the template is unlikely to be used more than once, then it may be better to hold one's nose and use the official version.

`heria` saw its first real-world use case in 2023, for a highly interdisciplinary Horizon Europe RIA proposal co-authored by 19 people across 14 organizations in ten countries. The organizations included universities, an independent research institute, several small businesses, and branches of a multinational company. Many of the co-authors held degrees in computer science, but others came from the social sciences or humanities. Accordingly, they varied greatly in their prior knowledge of LaTeX, from none at all up to several decades' experience.

The proposal document was hosted on Overleaf, which allowed co-authors to edit it online or to check it out via Git for offline editing. According to the document's edit history, there were 21 403 distinct edits made, of which 20 631 (96%) were carried out online in Overleaf and 4% were committed through Git. It should be noted, however, that Overleaf's tracking of changes is considerably more fine-grained than Git's. Someone writing offline might produce several pages' worth of material and then submit it in a single commit to the Git repository, but had the same material been entered directly into Overleaf, the service may have recorded this as hundreds of distinct changes. It should also be borne in mind that the package developer was among the co-authors, and about 10% of the Git commits included updates to the `heria` class itself.

At the time, there was no formal documentation for `heria`; the other co-authors were provided only with a lightly commented skeleton proposal and a 300-word `README` explaining how to compile it, add citations and to-do notes, and toggle the visibility of the instructions. Nonetheless, the writing process proceeded smoothly, with the package developer receiving virtually no questions of a TeXnical nature, even from the LaTeX neophytes. The writing process exposed a few bugs in the package code, which were duly fixed, and also provided the impetus for a few optimizations and aesthetic improvements. The present author was among those who helped perform an internal consistency check of the final draft of the proposal, and found considerably fewer issues than in a past experience with a Horizon proposal written in Microsoft Word. In the end, the `heria`-formatted proposal was submitted on time; it passed all formal checks by the funding agency and so was duly forwarded to the reviewers. It is either great modesty or great shame that prevents the author from revealing the final accept/reject decision here, though for our purposes it suffices to say that the use of `heria` played no direct part in it.

After the proposal was submitted, the developer informally surveyed its other 18 co-authors for their feedback on the writing process insofar as it related to using LaTeX and Overleaf in general and `heria` in particular, and asked them to compare the experience with those for any past proposals collaboratively written with different tools. Eight responses were received and all indicated a positive experience with the heria-based workflow. Four respondents specifically highlighted heria's capacity to enforce consistency in the proposal's structure, formatting, and/or references; three respondents expressed appreciation or enjoyment at being able to leverage their existing LaTeX knowledge; and two thought that the package enhanced the group's ability to edit collaboratively.

Several responses described past experiences with Microsoft Word, Microsoft 365, and Google Docs as being inefficient or even "painful", indicating that "formatting will be a mess with many people collaborating". Nonetheless, they recognized that these tools are more familiar to those outside computer science, and so may have a lower barrier to entry. They also praised Google Docs's commenting facility, which allows co-authors to annotate documents with tasks, to assign them to individual collaborators, and to receive email notifications whenever tasks are created, replied to, or resolved. The heria-based workflow had no comparable commenting facility; co-authors used a mixture of comments in the LaTeX source code, in-document comments typeset with the todonotes package, and Overleaf's own commenting feature. This proved to be problematic, since the source code comments were not always visible to co-authors using Overleaf's visual editor and the Overleaf comments were not visible to co-authors who checked out the project with Git to edit it offline. Since providing a general-purpose issue tracking system is well beyond the scope of heria, anyone considering heria (or any other LaTeX-based workflow) for a large collaborative project should therefore consider how best to coordinate tasks and discussions during the writing process.

7 Availability and future development

The heria package has an official website at logological.org/heria that includes links to its documentation, source code repository, and bug tracker. The package saw its initial release to CTAN on December 4, 2023, and it was added to TeX Live the following day. By the time this article is published, heria may also be available in other TeX distributions and online editors.

From time to time, the EC revises the official template for Horizon Europe RIA and IA proposals, and it is intended that heria, over the course of its development, will track these revisions. Whether this intention is realized depends on the availability and motivation of its maintainers, a group that for the moment consists solely of the present author. Anyone interested in directly contributing to the further development of heria is welcome to get in touch. Failing that, the best possible impetus for continued improvement of the package is the opportunity for the author to participate in further Horizon Europe proposals. (Readers are welcome to take this as a coy solicitation to collaborate on high-quality research projects.)

References

[1] N.H.F. Beebe. 25 Years of TeX and METAFONT: Looking back and looking forward — TUG 2003 keynote address. *TUGboat* 25(1):7–30, 2004. tug.org/TUGboat/tb25-1/beebe-2003keynote.pdf

[2] B. Donald. The nih package, 2005-06-01. ctan.org/pkg/nih

[3] D. Els, S. Fear. The booktabs package, version 1.61803398, 2020. ctan.org/pkg/booktabs

[4] Free Software Foundation. What is free software?, Feb. 2021. www.gnu.org/philosophy/free-sw.en.html

[5] H. Hagen. LuaTeX: Howling to the moon. *TUGboat* 26(2):152–157, 2005. tug.org/TUGboat/tb26-2/hagen.pdf

[6] G. Indiveri. The h2020proposal package, version 1.0, 2015-09-20. ctan.org/pkg/h2020proposal

[7] J. Karr. The grant package, version 0.0.5, 2019-02-26. ctan.org/pkg/grant

[8] M. Kohlhase. The proposal package, version 1.5, 2016-04-15. ctan.org/pkg/proposal

[9] D. McIlroy, E.N. Pinson, B.A. Tague. Unix time-sharing system: Foreword. *The Bell System Technical Journal*, 57(6):1899–1904, July–Aug. 1978. archive.org/details/bstj57-6-1899

[10] F. Qi. The mynsfc package, version 1.30, 2020-08-18. ctan.org/pkg/mynsfc

[11] R. Reich. Does TeX/LaTeX give a headstart with other programming languages? [answer], Jan. 2012. tex.stackexchange.com/a/42749/22603

[12] W. Skala. The pgfgantt package, version 5.0, 2018. ctan.org/pkg/pgfgantt

◇ Tristan Miller
Department of Computer Science
University of Manitoba
Tristan.Miller (at) umanitoba dot ca
https://logological.org
ORCID 0000-0002-0749-1100