PROJECT SUMMARY

Overview:

The main objective of this proposal is to lay the groundwork for transitioning the Generic Mapping Tools (GMT) from a PI-centric, high-cost maintenance model into a community-based, distributed-cost model. Our plan is to sustain the continued development and maintenance of GMT by building a broad community of practice around the project. This transition will shift the cost from funding a PI into funding the maintenance and growth of the community. The GMT team has traditionally been small (2-5 developers) and not much effort has gone into expanding the core team. To build a community around GMT, we must remove the barriers to entry and increase our efforts to recruit and mentor a diverse cohort of new developers. One of the ways we plan to accomplish this is through developer training workshops and social gatherings at large scientific meetings. To retain these new contributors, we will create a clear pathway for those interested in the project to get involved, climb the project hierarchy, and gain recognition for their efforts. In order to achieve this, we will create a governance model with rules, expectations, responsibilities, and mechanisms for progression and retirement. Alongside these community-building efforts, we will also work to decrease the large cognitive load required of new (and even experienced) developers when working on GMT. Based on the experience gained over the past 10 years, we will refactor the source code to reduce complexity, allow more thorough testing, and increase maintainability overall. Through these social and technical changes, we hope to distribute the development and maintenance burden of GMT to a broad and motivated group of users and developers. Our efforts will be focused on ensuring that the project is robust to changes in the development team (such as the retirement of the lead developer) and that the community can not only sustain itself but flourish.

Intellectual Merit:

The excellent engagement metrics of GMT testify that our tools are enabling a very broad range of cutting-edge geoscience research. GMT 6 (to be released this fall) will place "modern mode" into the hands of tens of thousands of users. Modern mode is to GMT what slicing is to bread. It is a lot easier to learn and most GMT users will want to take advantage of a greatly simplified interface and explore new unique features (subplots, insets, and especially animations for the masses). Classic mode remains the default so legacy GMT 4 and 5 scripts will remain compatible. Our sustainability plan will ensure that geoscientists currently relying on GMT to process data and communicate results can confidently continue to do so into the future. Furthermore, our three-part plan to simplify the code base, strengthen core functionality, and enlarge the community of developers will make GMT easier to maintain and less costly to support. The experience gained in undertaking these tasks can help other mature infrastructure projects transition to more sustainable models as well. The end goal of such activities is to preserve investments made and redistribute a larger portion of program allocations to core science priorities.

Broader Impacts:

GMT is the very definition of broader impact - it impacts thousands of scientists across a broad swath of disciplines well beyond geoscience and reaches the public at large: GMT maps abound on Wikipedia, "Science on a Sphere" displays, and in other public places. We seek to grow the size and diversity of the community that develops and maintains Open Source software for the geosciences. Emerging geoscientists are community-oriented and we will evolve our operations to tap into their enthusiasm and modern computational skills for the betterment of the project and hence the community itself. Involving a young post-doctoral researcher provides cutting-edge training in modern computational software development and data analysis, skillsets in high demand in both academia and industry. We will also engage undergraduate students in scientifically challenging tasks related to their major and beneficial to the GMT project.

D. A Sustainable Plan for the Future of the Generic Mapping Tools

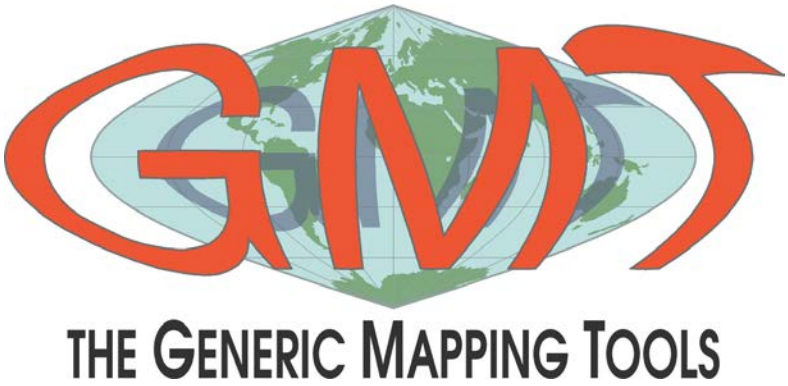

Fig. 1. The GMT logo is recognized around the world, with projected letters of its acronym on top of a Hammer projection world map. The *logo* module calls other GMT modules for plotting coastlines, filling polygons, and placing text via the GMT API to make the final illustration.

Preamble. The Generic Mapping Tools (GMT; Fig. 1) [*Wessel and Smith*, 1991, 1995, 1998; *Wessel et al.,* 2013] was born over 30 years ago. During that time span, GMT has seen continuous development and maintenance and become an essential tool set for Earth, ocean and planetary science. As the original founders and developers approach the twilight of their careers, the GMT stakeholders (i.e., current developers, users, funding agencies, our steering committee) must develop plans for how GMT will continue to serve its huge user base after the retirement of PI Paul Wessel from the project. Two weeklong developer summits in Faro, Portugal (January, 2019) and La Jolla, CA (July, 2019) were dedicated to discussing our sustainability plans; this proposal outlines what these are, who will be involved, the tasks that must be completed, and how governance and communications will evolve.

1. Background

The Generic Mapping Tools (GMT) was first released in 1988 at Lamont-Doherty Earth Observatory while Paul Wessel and Walter Smith were still graduate students (Fig. 2). Most users became familiar with it in 1991 when version 2.1.4 was released [*Wessel and Smith*, 1991]. Now at version 6.0.0, we have several tens of thousands of users across all continents. Generations of US scientists all (or at least the major marine and solid earth) R1 Research Universities, major government labs (e.g., USGS, NOAA, NASA), and many private institutions have used GMT for almost 30 years and it is ingrained in their mission-critical workflows. GMT is widely used in marine geology and geophysics [e.g., *Müller et al.,*2016], solid earth geophysics [e.g., *Ritsema et al.,*2011], geodesy [e.g., *McClusky et al.,*2000], geodynamics [e.g., *Steinberger et al.,*2004], oceanography [e.g., *Key et al.,*2004], and even for planetary geoscience applications [e.g., *Smith et al.,*1999]. GMT also provides the software foundation for higher-level tools used by the scientific community and funded by NSF, including MB-System [*Caress and Chayes*, 1996] for multibeam processing and GMTSAR [*Sandwell et al.,*2011] for radar interferometric applications. Citations of the GMT publications continue to increase unabated with over 1000 citations in 2018 alone, yet this is clearly an undercount; a perusal of AGU journals shows many articles with GMT illustrations that are not citing our works. Anecdotal evidence from seeing GMT plots in numerous AGU posters adds to the tally. As our success has increased, so has feedback and requests for new features. There is now more, rather than less maintenance. GMT still strives to be FAIR (Findable, Accessible, Interoperable, Reusable), and the data we do supply (coastlines, remote DEM grids) follow the FAIR

principles. We believe that, among geophysical toolsets, we are at the leading edge of modern cyberinfrastructure (we employ GitHub, Continuous Integration, Static Analyzers, Coverity, etc.)

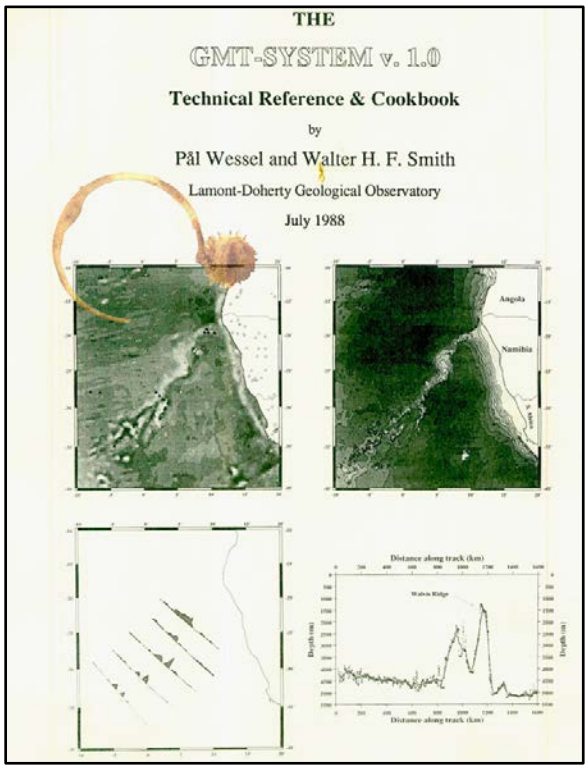

Fig. 2 *The cover page of GMT v1.0, an internal release only at LDEO. dated July 1988. Coding started in early 1987 (based on the earliest date written in the 1.0 source code).*

As founder and most active developer, Paul Wessel has designed and written the majority of the GMT code base, including the GMT 5 API that now allows the MATLAB and Octave [*Wessel and Luis*, 2017], Python [*Uieda and Wessel*, 2017], and Julia [*Luis and Wessel*, 2018] external interfaces to be developed. He has programmed in C since 1986 and has a deep understanding of the large GMT code base. Walter Smith was a "partner in crime" from the beginning but ceased making coding contributions more than 10 years ago. Yet, he remains an essential part of the GMT team due to his vast knowledge of data processing, statistics, and numerical methods. His participation has largely been underwritten by NOAA. GMT has also received tremendous assistance from volunteers residing around the world. A few of these volunteers have joined the GMT core developer team over the years (Fig. 3): Joaquim Luis, Remko Scharroo, and Florian Wobbe have all made fundamental contributions to GMT. As scientists mostly employed at various institutions outside the US, these contributions have come at no cost to NSF (except for modest travel support to our developer summits). In 2017 a CIF-21 grant with NSF/OCE matching allowed Dr. Leonardo Uieda to join the team with emphasis on developing a Python interface for GMT. Most recently, Dr. Dongdong Tian (currently a postdoc in seismology at Michigan State University) has joined the core development team as a volunteer. He is also responsible for gmt-china.org, a grassroots community with thousands of users whose bug-reports help us strengthen GMT. Table 1 shows a fact sheet for GMT, including funding, while Table 2 highlights the core GMT team members.

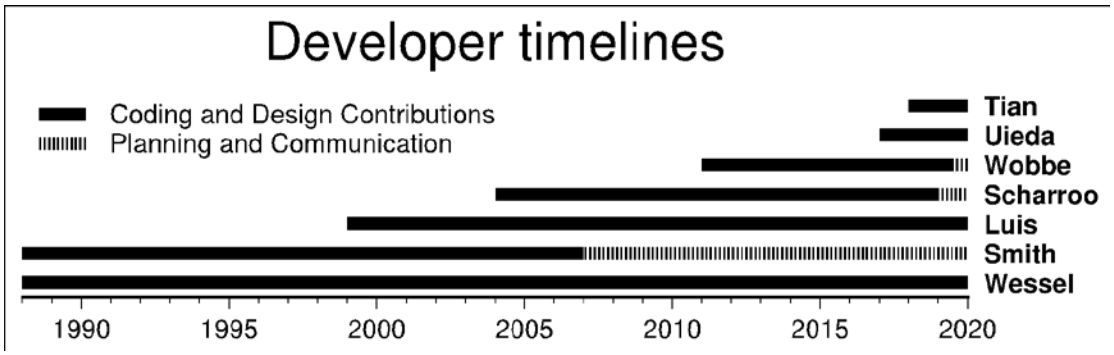

Fig. 3. Timeline for when our developers joined the GMT core team and when they made coding, design, planning and communication contributions.

Table 1. GMT fact sheet

Origin:	1988 at LDEO [Wessel/Smith while graduate students]
Initial release:	Oct 8, 1991, v 2.1.4 (via EOS article *Wessel and Smith* [1991])
Current release:	Jan. 1, 2019, v 5.4.5 [6.0.0 in development; 6.0.0rc3 released]
NSF funding:	Since Oct 1, 1993, a total of $1.5M (80% OCE, 20% EAR)
SOEST support:	~$100,000 in hardware support, plus IT support
UHM support:	~$1.5M: ~2 months/year of salary support to Wessel since 1991
International support:	Several employee-years at unknown cost

Table 2. The GMT core team members

Joaquim Luis is a Professor at the University of Algarve, Portugal, working on marine magnetics, plate kinematics, tsunami modeling and scientific software development. He is a GMT developer as well as the developer of Mirone [*Luis*, 2007], a GUI type interface for GMT on Windows, and developer behind the Julia interface [*Luis and Wessel*, 2018].
Remko Scharroo is a Remote Sensing Scientist at EUMETSAT, the European Organisation for the Exploitation of Meteorological Satellites. He works on definition and validation of altimeter data products and algorithms, with particular interest in monitoring of sea level rise. He developed and improved several elements of the GMT code, dealing with gridded data and the PostScript engine.
Walter H F Smith is a Geophysicist at NOAA. His current research focuses on satellite radar signal processing to enhance applications in marine geophysics, physical oceanography, cryosphere science, and weather and climate forecasting. He is the co-founder, with Paul Wessel, of GMT, and is co-author with David Sandwell of the "Smith and Sandwell" method of estimating ocean depth from satellite altimetry [e.g., *Smith and Sandwell*, 2004].
Dongdong Tian is a post-doctoral Researcher at Michigan State University. He is a seismologist who maintains the SAC tool in the seis GMT supplements and addresses issues related to the other tools in that supplement. He contributes with GitHub management, documentation, C code improvements as well as assisting Dr. Uieda with the Python interface.
Leonardo Uieda is a Lecturer at the University of Liverpool, UK (from August 2019). He is a geophysicist whose expertise covers geopotential field data processing, forward modeling, and inversion. He was the 2017–19 GMT/Python post-doctoral fellow at SOEST, UHM working on the PyGMT package [*Uieda and Wessel*, 2017].
Paul Wessel is a Professor at SOEST, University of Hawaii at Manoa working on plate tectonics, lithosphere geodynamics and scientific software development. He is the main developer and maintainer of GMT and contributor to GMTSAR [*Sandwell et al.,* 2011]. He designed the GMT API, built the MATLAB interface with Luis [*Wessel and Luis*, 2017], and supervised Dr. Uieda.
Florian Wobbe got his PhD from the Alfred Wegener Institute in Bremerhaven, Germany. He worked on plate-tectonic reconstructions of the Southern Ocean and joined the GMT developer team in 2011. He added CMake build system support, compression and tiling via netCDF-4 and integrated external FFT libraries. He now works for Sun and Sea Technology in Germany.

The School of Ocean and Earth Science and Technology (SOEST) and the University of Hawaii at Manoa (UHM) have both made significant contributions to GMT over the years, such as in computer infrastructure and salary support for Paul Wessel, matching NSF contributions over the same period. In addition, we have had access to SOEST IT assistance to help maintain the GMT servers, web pages, and more. Likewise, the GMT project has benefited tremendously from the generous support by numerous organizations in the US and abroad who have hosted the GMT mirrors over three decades. Finally, GMT would not be the juggernaut it is today without the volunteer contributions from our international collaborators and their long-term dedication to the project. These efforts have come at no salary expense to US funding agencies, meaning GMT has been an incredible investment with a high yield value. As an illustrative example, all GMT users on Windows can largely thank volunteer Joaquim Luis at the University of Algarve, Portugal for the fact that GMT runs superbly on Windows.

Relying on volunteers has been the GMT modus operandi since its inception, and this principle will remain at the core of GMT philosophy. However, there is now an urgent need to bring GMT to a place where it can be sustainably maintained once founder and main developer Paul Wessel retires in about three years. Deliberating about this transition has taken place over the last two years and this proposal is the culmination of a long road of planning and evaluation.

2. Intellectual Merit: The GMT Succession Planning

The goal of the three years of activities proposed herein is to transition GMT so it may remain a vital and sustainable set of computational infrastructure for the Earth, Ocean, and Planetary Sciences in the following decade or more. After the major endeavors detailed in this proposal, GMT should have a very good chance of needing significantly less support for maintenance. Longer-term initiatives requiring GMT enhancements may arise from the various science communities but is not the concern of this proposal. Should such funds be needed, they will be solicited elsewhere.

2.1 The Process of Planning

The GMT developers have been in contact with NSF program managers since 2016, when we indicated a succession plan will be needed and NSF suggested it be sustainable. Thus, we have been addressing some aspects of succession planning over the last several years. Recently, most of the core GMT developers met in Faro, Portugal, for a weeklong summit (Jan 21-27, 2019). Due to the US government shutdown, Walter Smith was unfortunately unable to attend. Also, Remko Scharroo could not attend due to work obligations. Attendees thus were Luis, Uieda, Wessel and Wobbe; Scharroo participated in one Skype video consulting session. The outcome of this meeting was a "white paper" shared with our stakeholders (NSF, the GMT Steering Committee). Our second weeklong summit took place at Scripps (July 29-Aug 2, 2019) and had participation by Luis, Uieda, Smith, Wessel, and Tian (new developer), with Scharroo and Wobbe unable to attend. This summit included the steering committee (present: Dave Sandwell, SIO (Chair), Dave Caress, MBARI, Steve Diggs, SIO) and an NSF representative. The white paper was discussed, tentative plans presented, and following a full day of discussion and another day for team debriefing, our prioritizations crystallized; the result of this lengthy process is the current proposal you are reading.

2.2 Executive Summary of our Sustainability Plan

There are three broad areas that this proposal will seek to address, develop or strengthen. Each are important to the project and its survival, for different reasons.

1. Recruitment and training of new developers
2. Building a broad and sustainable developer community
3. Modernization, simplification and strengthening of the GMT code base

Of these three tasks, two are critical social activities and only the last item is mostly a technical one along the lines of our previous proposals. Here, we will outline what each area entails and provide some of the details and justifications for each task.

2.2.1 Recruitment and Training of New Developers

Despite our success in creating and expanding the GMT project over the last three decades, we have not previously prioritized recruitment of new developers or even made it easy for the few brave souls who persisted in contributing to join the "Exalted Few" (i.e., the core team). Now, this lack of a clear pathway for new developers is our biggest Achilles heel. As Fig. 2 indicates, we have a relatively small core of dedicated GMT developers. However, as these age out of the project or leave for personal reasons (job and/or family changes), we will need a steady influx of new developers to even sustain the project, let alone allow it to grow. We believe there are many potential contributors in our fields who would want to be part of a broad group that seeks to strengthen the infrastructure that the community at large intimately depends on. Based on what other successful Open Source projects do, we realize that we must make a stronger effort to convert existing users into project contributors. The strategy that we have agreed upon is to offer user and, most importantly, *developer* workshops. These workshops will, to the extent possible, be run as short courses offered at (or in conjunction with) large scientific meetings, such as AGU and GSA. The main goal of these workshops is to lower the threshold for existing users to join the development team. As our history has shown that developers tend to arise from the larger group of advanced users, we will also begin placing more emphasis on the training of new users.

We propose a four-pronged approach: (1) We will continue to run GMT user short-courses, such as those offered by UNAVCO (we ran our inaugural GMT short course on July 22-23 2019, followed by a 3-day UNAVCO GMTSAR short course). (2) The materials we develop for these short courses will form the basis for updated tutorials that will be available from the GMT website. As these short courses are recorded, there will be both written and audiovisual training materials that can be reused by other instructors. (3) We will seek to identify experienced and motivated user and developer workshop participants and encourage them to run these short courses at their own institutions. We will provide support for these activities via materials, scripts, data and videos. Following the old "teach a man to fish" dictum, we will use our workshops to train new instructors so that we can reach a wider audience, based on the very successful "Software Carpentry" model. The training material will be available online with an open license, while promotional material will be handed out during the meetings (Fig. 4). We will also encourage some of our more frequent contributors to become active developers, such as Andreas Bjørnestad and Kristof Koch, who both have made significant, non-coding contributions to GMT.

2.2.2 Building a Broad and Sustainable Developer Community

After participants go through our developer workshops, they will require more information about the development process, how decisions are made within the project, how to access our communication channels, and more. They must also find a welcoming community and feel empowered to take responsibility within the project. Without these mechanisms in place, we stand no chance of retaining these new developers and our training efforts will be in vain.

We have researched what governance structures and procedures other successful open source projects have adopted to grow and sustain their developer communities. We found that they all have in common: (1) a thorough *Contribution Guide* that allows both experienced and beginning developers to learn how to make code and documentation contributions. (2) a *Code of Conduct* that is strictly enforced to ensure interactions are positive and actively encourage diverse contributions. (3) a *Governing Council* composed of a representative subset of the project community and documented mechanisms for current members to retire and new members to be appointed.

Clearly defining the Governance structure of GMT will be critical. It is important to acknowledge the varied contributions that users make and have a clear pathway for contributors to climb the hierarchy and join the Governing Council. We therefore need to develop our own set of governance documents based on successful projects, such as *Jupyter* [in fact, we had an online conversation with a postdoc on the Jupyter team to learn about their procedures]. Like many Open Source projects, it is likely that founder Paul Wessel will be made "Benevolent Dictator For Life" (BDFL) and serve on the Governing Council until the content of his speech is decoupled from the workings of his brain, or he is hit by a bus (whatever comes first). In fact, increasing a project's *bus factor* is frequently a common goal in Open Source development (google "Bus_factor"). The goal of these changes in governance is to encourage the development of a *sustainable* community that is able to continue the project beyond the tenure of the BDFL. For example, the Python BDFL has recently stepped down but the project has adapted and will likely continue to grow. It is expected that the Governing Council will be predominantly composed of active developers who have insight into the status and inner workings of GMT. In addition, we will restructure the current GMT Steering Committee into an *Advisory Committee*, whose function will be to monitor the sustainability progress, assist in advertising and arranging workshops, and advise the Governing Council on broad changes needed, alternate funding opportunities, and diversity and inclusion efforts.

Growing an engaged user, developer, and instructor cohort requires constant effort and attention. To this end and to further increase the transparency of our decision-making process, we will: (1) hold off-site developer meetings for cohort building and strategic planning. Invitations will be extended to current and aspiring project contributors and workshop attendees. These will be scheduled in conjunction with AGU or other large meetings. (2) host regular online developer meetings through video conferences to discuss progress and solicit feedback on ideas. The meetings will be announced through our communication channels (website, forum, chat) and will be recorded and made available online. Users and potential developers who are curious about how we operate can participate in these sessions, just watch and listen, or watch the recordings at a later time. The purpose of this open approach is to lower the barrier to participation and encourage a broader and more diverse developer cohort. Similarly, we want to increase the diversity of the Advisory Committee and Governing Council and will be proactive in recruiting good role models to the project. Finally, we will seek to have a social function at the large scientific meetings, such as a GMT luncheon or similar. We believe it is important to have a physical presence – not everything can be an online experience.

Fig. 4. *We are experimenting with coasters and stickers that will be distributed at our upcoming GMT user and developer workshops.*

2.2.3 Modernization, Simplification and Strengthening of the GMT Code Base

The GMT project has seen numerous technical paradigm changes over the decades: Transitions from 32-bit to 64-bit, the rise of architecture-independent data formats (XDR, then netCDF), moves from Makefiles via autotools to cross-platform CMake for building, increased dependency on prerequisite libraries (netCDF, GDAL, PCRE, Curl, FFTW, etc.), and frequent migration to new version control systems (git is our 4th such system). All of these transitions have involved considerable redesign, redevelopment, and reorganization efforts. Per Charles Lyell's inverse dictum that "the past is the key to the future" [*Doe*, 1983], we know that there is no such thing as cessation of development as we move forward. An analogy with house ownership suggests that, while building the house is an initial and costly investment, continued maintenance is required to prevent decay and damage. Along the way, the homeowner may need to rip out some old parts and get newer parts that are up to spec and work with the rest of the house infrastructure. So is the case with software. For instance, to ensure that GMT 6 (and later versions) remains compatible with GMT 4 (and 5), we have tests to check for compatibility and we encourage users to let us know if something that used to work in GMT 4 no longer works. We are on record stating that GMT must retain compatibility with GMT 4 options, even though some options have been modified and old-style usage may result in warnings (but not errors).

Apart from relative minor feature enhancement, bug fixes, optimization, code hardening, profiling, documentation improvements, website refresh, and a myriad of other routine maintenance tasks that will continue unabated until the project is no longer useful (Fig. 5), there are several larger undertakings that will require focused and sustained efforts to see them through. Here, we highlight these tasks, and why they are needed.

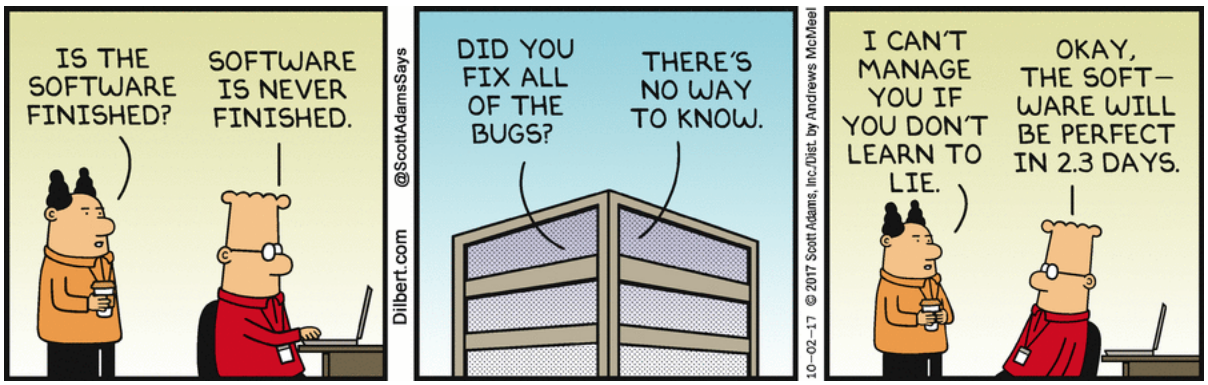

Fig. 5. We may laugh at the impedance mismatch between managers and developers, but it is irrefutable that vigilance will be required to prevent GMT "bit rot" and ensure compatibility with other computational infrastructure as they continue to evolve as well.

Refactoring of the C Library. It is 10 years since we began a major refactoring of the code that produced an API on which GMT 5 was built [*Wessel et al.,* 2013]. While a major milestone in our development, several compromises were made to make it possible in the required timeframe. For example, the workload was greatly reduced by the decision to retain the i/o (input and output) operations inside the new module functions that substituted the previous GMT standalone programs. This approach minimized code changes and was invisible to command-line users. However, as we have gained experience with using GMT from other environments (MATLAB, Octave, Python, Julia, GMTSAR, and MB-System), we have seen the limitations of the 2009 design decision: passing data in and out of the API functions is cumbersome, adding suitable automated unit tests is difficult as the i/o gets in the way, and there is a steep learning curve to understand how the codebase works internally and, hence, how to change it.

We will refactor the library to separate the i/o from the calculation and plotting algorithms and make functions smaller and easier to test in isolation (i.e., unit tests). While the refactoring will have little to no direct impact on the GMT command line, it is essential for developing robust unit tests with broad coverage as well as allowing new developers to make meaningful contributions to the code. It will also be instrumental in reducing the effort required to develop and maintain the external interfaces, such as Python, MATLAB, Julia, R, and others. We are convinced that this refactoring is the key to GMT remaining relevant with new generations of scientists.

To illustrate what we are proposing, consider two of the workhorses of GMT: the modules ***blockmedian*** and ***surface***. These are routinely used by scientists to decimate and grid sizeable data sets. For instance, the 15x15 arc second global Earth relief grid (86400 by 43200 nodes) is created from over half a billion of individual data points [*Tozer et al.,* 2019]. We are not aware of other tools that can handle these operations. However, the way the 2009-2013 refactoring worked was to convert the two stand-alone programs ***blockmedian*** and ***surface*** into two functions that could be called from the GMT library. Yet, because both programs used to perform their own i/o, these operations were still part of the API functions. For a developer with input data already in memory to call, ***blockmedian*** via the GMT API, they will need to set up "virtual files" pointing to the memory and pass these special filenames to ***blockmedian*** since it expects to read data from a "file". Output from that module is handled similarly and we may "write" into another virtual file (i.e., another memory matrix). While this design allowed GMT 5 to break away from just supporting the command line users, it is now causing design difficulties for the Python and Julia interfaces and projects like GMTSAR . We now realize it would be much better if ***blockmedian*** were written as one or more functions that accept input *x,y,z* data via arrays or a matrix, a few parameters that control the module's operations, and then returns its result via another matrix or set of arrays. Such a function could then be called in a variety of settings and it would be up to the developer to supply and receive the data, i.e., they would handle the i/o separately and let the function just do the calculations. Likewise, ***surface*** should also see a separation of i/o from the calculations and this would simplify our attempts to parallelize this finite difference gridding algorithm and speed up these calculations (which are becoming a bottleneck for InSAR processing with its very large grids; David Sandwell, pers. communication). A simplified and more robust C library would also be much more appealing for other software projects to call upon directly (GMTSAR and MB-System are just two prominent examples). We envision GMT living beyond the command line interface as the underlying library for higher level computations, much as LAPACK is now for linear algebra. Expanding GMT usage into beyond the command and into higher level languages is critical for recruitment of users and developers alike.

Modernizing the command-line interface. While GMT *modern mode* (released in GMT 6.0.0 this fall) greatly simplifies the task of learning GMT by eliminating the top three problems encountered by thousands of GMT users in *classic mode* (the PostScript cake-baking with **-O**, **-K**, **-P**; the redirection of Postscript output to specific files and eventual conversion to other more convenient graphics formats, and the constant repetition of region and projection settings for all commands), the options themselves remain terse and cryptic, giving novices a stark dose of fear and trepidation. The solution to this problem, like modern mode was to classic mode, is to offer an alternate set of options that relies on descriptive keywords instead of single letter option codes. For instance, pretend for a minute you are a GMT novice looking at a GMT script and compare the command string "-RFR -JM15 -G255/255/0" with a hypothetical modernized version of the same command, such as "--region=France --projection=mercator --width=15c --fillcolor=yellow". The latter is understandable regardless of any previous exposure to GMT (and we will use established ***proj4*** syntax for projection parameters). We propose to add such a modern option layer as an optional feature while retaining compatibility with the classic behavior. Thus, thousands of existing GMT 4–6 scripts will continue to work as they do now. Meanwhile, new scripts can be written using the

modern syntax, making them readable to new and experienced users alike. In particular, using this syntax in our documentation will make learning GMT as accessible as other competing science tools. Again, the ability to simplify GMT is intrinsically linked to the recruiting of both users and developers. If GMT starts to "show its age" it will be harder to recruit either, risking the investments already committed by funding agencies and developers.

Develop GSHHG version 3. The ease of making quality geographic maps with coastlines and filled land is integral to the GMT experience. However, maintaining a global high-resolution coastline database is taxing, especially when the source data are showing their age. GSHHG derives from WDBII from the 1970s [*Gorny*, 1977)] and WVS from the 1980s [*Soluri and Woodson*, 1990]; see *Wessel and Smith* [1996] for details. Yet, our GSHHG database remains popular, even outside of GMT (e.g., matplotlib). We receive numerous inquiries helpfully pointing out shortcomings in the data, while in some cases (such as NE Greenland) these are better labeled "longcomings". Our solution, as we have proposed earlier (see Fig. 6 for an example from our 2015 five-year proposal that was reduced to a two-year proposal and GSHHG 3 was put on hold), is to source the next generation GSHHG from the OpenStreetMap project. We will produce new sets of preprocessed tiles that allow auto-assembly into fillable polygons on the fly, as GMT has been known for doing. The task is both straightforward and formidable, with ~80 Gb of geographic data to simplify into something that plots near instantaneously. Especially the "dumbing down" to lower resolutions [e.g., *Douglas & Peucker*, 1973, *Pallero*, 2013] is fraught with issues, such as self-intersections and intersections between nearby features after line reduction. We believe manual editing performed by undergraduate students using a custom tool is the fastest and most cost-effective solution to fix these complicated artefacts.

Our whole process of building the coastline tiles, while briefly discussed in *Wessel and Smith* [1996], consists of a poorly document set of GMT4-era C programs. As such, the process is not reproducible or sustainable, requiring intimate knowledge that is not readily accessible. Tackling the OpenStreetMap reformatting task provides an opportunity to automate and document this process, allowing future efforts with improved data to be much more manageable.

Vector data interoperability. GMT has the ability to read almost any grid file format since we now rely on GDAL to handle formats we do not natively support (including *images*). Yet, the other major type of scientific data (vector data: points, lines, polygons) do not yet have the same level of support. Thanks to a contribution from user Brent Wood at NIWA (New Zealand), GDAL developer Frank Warmerdam was paid to implemented an "ASCII GMT Shapefile" format in GDAL's ***ogr2ogr*** tool, allowing users to convert between formats such as shapefiles and the new GMT/OGR format. This interaction allows us to indirectly handle shapefiles and other standard vector formats, but it is undeniably cumbersome. A better solution would be to replicate the design decision of using GDAL functions in GMT to read grids: Use GDAL's OGR functions internally in GMT to handle the i/o of vector formats we do not support natively. This enhancement will greatly improve GMT's *interoperability* with many modern GIS-like tools. We should note we have received considerable feedback from users wishing to use shapefiles with GMT and the detour via ***ogr2ogr***, while operational, is seen as a stopgap measure. Like the GDAL bridge before it, implementing the OGR bridge will greatly improve the interoperability of GMT, perhaps eliminate the need for the GMT/OGR transitional file format entirely, and simplify our own vector data i/o library codes. Consequently, the OGR bridge would strengthen GMT's position as a GIS Swiss Army knife immensely. The maintainer of GDAL is now Even Rouault and the GDAL community remains very active.

4. Progress already undertaken or on the way

This proposal builds on a large set of improvements already implemented or in the works. Here we discuss some of these since they are relevant to our proposed plans or augment them. We are of

course carrying out the usual day-to-day management of GMT (i.e., responding to bug reports, considering and researching feature requests from NSF-funded scientists, addressing GMT usage issues, and much more) as well as working on GMT 6 development and maintenance.

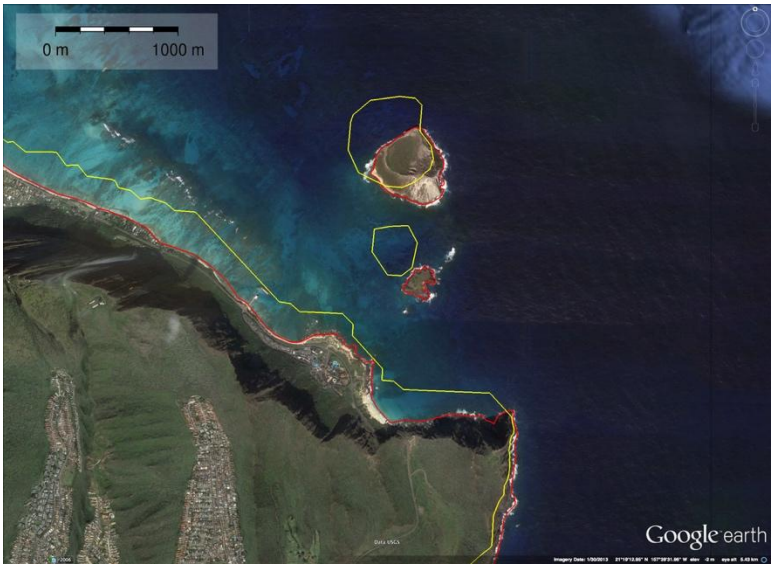

Fig. 6. *The limitations of GSHHG*. Easternmost point on Oahu, Hawaii, showing a Google Earth view of the uninhabited islets Manana and Kaohikaipu across from the tourist attraction Sea Life Park. The GSHHG full-resolution coastline is drawn in yellow, while the OpenStreetMap coastline is shown in red. Note the lower precision of GSHHG as well as its lack of accuracy.

Modern Mode in GMT 6. Thanks to a grant from the NSF EarthScope program, we have been able to take ideas of a better GMT (developed during our discussions at the 2016 Developer Summit) and implement them in GMT 6. The result is GMT *modern mode*, released as part of GMT 6 in September 2019 (estimated). Modern mode greatly simplifies GMT scripting and adds major new capabilities, such as new modules for creating subplots, placing map insets, and easily building animations. We cannot do justice to modern mode in this proposal but we have written a paper on GMT 6 for G^3 that is currently in press. Until GMT 6 is formally released, we maintain release candidates; the latest is GMT 6.0.0 rc4 updated Aug 15, 2019.

Google Earth and GMT. We have also improved GMT's ties to Google Earth by supplying a new tool (grd2kml) that builds quadtree images from a grid so that huge data sets can be seamlessly overlain in Google Earth. Since we already support vector data via gmt2kml this new utility brings the full force of GMT to a great platform for presenting and discussing science.

Migration to GitHub. We successfully transitioned our entire repository from subversion to git last summer. This transition took us several days to complete since the GMT history goes back decades and the repositories were full of references to user accounts and addresses no longer valid. To avoid an unwieldy repository, we decided to retain commits back to 1998 only. The last year has seen the GMT team, especially the elders with residual memory of SCCS, CVS, and SVN, learning how to operate with git branches, pull requests, and merging. The move to GitHub has also greatly improved how users can report bugs, request new features, while allowing us to keep track of progress on all ongoing undertakings. Bringing GMT to GitHub has also allowed new developers to make smaller contributions to documentations or code features, which was not easy to do with subversion. Case in point: new developer Dongdong Tian started making contributions as soon as our GitHub repository was up and running. Just being on GitHub is, in fact, a critical selling point to potential developers.

Continuous Integration. With the migration from subversion to GitHub we now have Continuous Integration (CI) running for Linux, Windows, and macOS. The CI greatly assists a software project by working in the following way: Each time there is a change (commit) or a pull request made (a bug fix, or new feature added), the CI system will attempt to compile and (ideally) test the updated code. The update must pass all specified tests before the pull request can be merged into the repository. This requirement (which is applied to the core development team as well) ensures that

we will never accidentally merge in a patch that breaks GMT compilation. We are considering how to create a subset of the many test scripts that will get checked along with compilation to minimize the introduction of new bugs. In addition, CI allows many routine tasks to be fully automated. For example, our CI systems already automatically build and update the GMT documentation once a day. We wish to add more CI scripts to completely automate the building of installers when a new version is to be released, freeing up valuable developer time.

Test suite. GMT is an infinitely customizable tool and can be run in a myriad of ways. While this is a big selling point, the downside is that it is difficult to test all the ways in which a program can be executed. As bugs are reported, we design test scripts to make sure they are fixed and add them to our test database. We are now pushing 850 test scripts and would like to have many more. However, these are not arranged with growth in mind and the current test suite already takes a considerable amount of time to run. As more tests are added (which is a good thing) it will also take even longer to finish the tests. Once a developer needs to run the test several times (to ensure a bug fix is not breaking other things - the whack-a-mole dilemma), having to wait for a long time for the test results decreases the likelihood that they will actually run the tests. Furthermore, if tests take too long then the CI may time out and not complete. We need to rethink how we organize the tests and sort them into different tiers so that not all tests need to be run by the CI, or perhaps automate by running subsets of tests via CI nightly. Once that is solved we can add more tests that are only run on demand for major checking. Finally, a key concept in testing is coverage: there is no point having 10,000 tests if they collectively only execute the same 10% of the code. Thus, designing tests that provide complete coverage is very difficult but will be considerably easier after the proposed C refactoring. The other, and perhaps obvious, reason for test suites is to make sure the results are correct. Lots of important science is being done with GMT and published in the scientific literature. It is crucial that the software produces correct results. Hence, we must be vigilant with testing, which is why we emphasize the need for refactoring and unit tests.

Website Update. The GMT server at SOEST ran the GMT site and wiki using the Redmine software inside a virtual machine running CentOS 5. Unfortunately, due to technical difficulties and inability to update the "reCaptcha" plugin within the VM server, we had to move everything off this server. In the process we have also studied how other successful projects present their materials on their webpages (e.g., matplotlib). The inescapable conclusion is that our documentation needs a major redesign both in content and structure, so that the index page with man pages for modules is not the first thing users see. These should be present but placed under a "Technical Reference" section. Instead we should have shorter "how-to" pages and tutorials up front. With GMT *modern mode* coming in 6.0.0 we need a better way to present both the classic and modern usages without duplicating documentation codes. We need a better outline of the material in the Cookbook, i.e., how to set pens, fills, fonts should be easy to find with one click. We hope to reduce the amount of questions on the GMT forum from inexperienced users by making these critical resources easier to find in a timely manner. We also need a *culture change* among the developers: If an issue is reported and the documentation is in fact lacking, then we should update documentation instead of explaining by private email or forum how to do things. To reduce maintenance and training, we want a new gallery of examples that is automatically built from scripts (by sphinx). Finally, we need to add complete documentation on how to add new examples to the docs (i.e., document the documentation). Since we are still operating out of SOEST, University of Hawaii, it is important that we may host a site that is not "hawaii.edu". Thus, we have acquired the domain name *generic-mapping-tools.org* and we are now using it to resolve the actual URLs of GMT web services and give the project a permanent address (we have registered and paid for the domain name for the next 10 years). We have applied for and secured permission from the UH President to host a non-"hawaii.edu" site directly on the UH network. This will give us options to migrate the site to run elsewhere, such as at UNAVCO or CIG, without noticeable changes to users.

Short-courses. We ran a two-day UNAVCO-sponsored GMT short course at Scripps in July 22-23. This workshop had a geodesy theme and was pitched to beginners. The short course relied on GMT modern mode and had ~25 participants (both on-site and on-line) and was well received. We expect these to be annual events. We have submitted a request to AGU for holding a pilot GMT developer workshop during the 2019 AGU Fall meeting. The intent is to attract GMT users who are eager to contribute to the project. As mentioned, developer training is a key pillar in our succession and sustainability plan.

Developer Update. A new core developer has joined the team: Dr. Dongdong Tian, a postdoc in seismology at Michigan State University. Dr. Tian has already made many contributions (over the last ~2 years) and we have met in person at the Fall AGU meeting twice; he also participated in the UNAVCO GMT workshop as well as the GMT summit we held this summer. Tian also maintains gmt-china.org and thus leverages efforts in China that benefit all GMT users. At the same time, it now seems inevitable that our long-time developers Scharroo and Wobbe will greatly curtail their involvement for the coming years: Both have new work responsibilities and a young family addition to support. These events drive home the need for us to accelerate the recruitment of "new blood".

GMT for Planetary Science. We have submitted a proposal to NASA specifically for the support of planetary-specific applications of GMT. This proposal, if funded, will add easy access to planetary gridded dataset from GMT via the remote-file mechanism we prototyped in GMT 5.4 and streamlined for GMT 6. The proposal was carefully designed not to step on our long-term development under the NSF umbrella, and should it not be funded then these specific tasks simply will not happen or be fully dependent on volunteer contributions.

User Installation. We are working to simplify the build and install process for users who need to build GMT from source code. A simple script that takes care of running the CMake commands and their options will be designed. This feature is somewhat urgent since most Linux users rely on repositories that are lagging seriously behind the GMT release cycle (e.g., Ubuntu), hence their only hope to get the current GMT version is often to install from source. For Windows and OS X users we are considering to set up nightly builds of installers (since the CI can do that for us) so that these users can run the very latest developer version and assist with bug detection and testing without having to build GMT themselves.

User Forum. Because our Redmine-driven Wiki with user forum on the SOEST server is partly broken, we have researched alternate solutions for a new user forum. We have settled on using Discourse as a forum replacement and are now working on setting this up at SOEST (but it will be accessible via an URL from our site so that the whole forum may be migrated to another server if required).

5. Work Plan

Given the considerable undertakings discussed above and the need to perform these tasks before PI Wessel retires, this proposal calls for the hiring of a postdoctoral researcher to replace the previous (and only) post-doctoral researcher that worked on GMT, Leonardo Uieda. He will move from Hawaii and start as a new Lecturer at the University of Liverpool, UK this August. While he will remain a vital force on the GMT team, his day-job will be to teach and research geophysics, not exclusively write code for GMT. Thus, in order to accomplish the large refactoring and other tasks proposed in Section 2.2.3, we need a steady (and salaried) contributor to ensure that we can reach our milestones for sustainability. Based on our previous two decades of development and deliverables, we have considerable experience with shepherding GMT development that depends to a major extent on the contributions from volunteers. Thus, the roles in the proposed project will be as follows:

- **Paul Wessel** as PI will lead the project and supervise all development. He will mentor the postdoctoral researcher (see mentoring plan) and oversee the C library refactoring that to a large extent will fall on the postdoc, with support from the GMT team. Wessel will lead the design of the keyword/value long-format option redesign and work with contributor Joaquim Luis on the GDAL OGR vector interface. He and the postdoc will design a workflow and oversee the undergraduate student and the work on GSHHG 3. Wessel will lead the development of user and developers course materials with assistance from the GMT team.
- **Leonardo Uieda** as co-I will be responsible for overseeing and coordinating the ongoing development of PyGMT, our Python interface, to which the post-doctoral researcher will contribute. Dr. Uieda remains an Affiliate Researcher at UHM to ensure continuity of this vital sub-project despite his move to the UK. He is also an essential maintainer of our GitHub account, the redesigned website, and (with Dr. Tian) maintains the growing list of Continued Integration scripts running on Travis and Azure Pipelines.
- **Postdoctoral researcher** (unnamed) will gain training in modern software development (GitHub, Continued Integration, static analyzers, profiling), all within a geophysical context. We expect the candidate to have development skills in both C and Python as our work overlaps both spheres of development. PI Wessel will oversee all C development while co-I Uieda will supervise the Python work. *Aside*: Back in 2016, we were not sure if a post-doctoral researcher would be a better choice than a paid hourly programmer. Our experience working with Uieda has made it obvious that the postdoc route is vastly better, since it recruits a life-long computational scientist into the upper echelon of the project leadership. The postdoc will also gain valuable experience and networking from interactions with renowned scientists in the GMT team and Advisory Committee.
- **Undergraduate student**(s) (unnamed) will assist the PI and postdoc with processing and editing of the GSHHG 3 data. We will of course try to automate this workflow to the extent possible, but past experiences dictate that there will be trade-offs between expensive developer time and relatively inexpensive manual interaction by an undergraduate student.
- **Joaquim Luis** and **Dongdong Tian,** as external collaborators and team members**,** will contribute as their time permits. Again, based on past experience these contributions are substantial and irreplaceable.
- The remaining team (**Smith**, **Scharroo**, **Wobbe**) will be engaged as their time allows in activities they are interested in, including online meetings and our informal meetings at AGU.

The SOEST-based GMT team will meet weekly and set minor and major milestones to monitor progress. These plans will be further reviewed in monthly online meetings with the full GMT developer team. We also expect to meet in person by taking advantage of GMT short courses at major scientific meetings as well as the UNAVCO summer short-courses in GMT and GMTSAR. Furthermore, we aim to have online meetings with the Advisory Committee at least once or twice a year, and will touch base with them at various meetings.

6. Time Table

This is a three-year project to address four major outstanding technical issues, create a governance and developer recruitment process similar to other Open Source projects, and nurture the community of GMT users from which developers will arise. We expect to be able to accomplish these tasks in the given time provided the accelerated progress that the involvement of a post-doctoral researcher with prior experience in C and Python coding can bring. In addition, PI Wessel will contribute considerable development time partly supported by UHM to bring GMT to a higher

and more sustainable level. Because the post-doctoral researcher provides dependable structures, we expect to follow these timelines:

Year 1: We will research, discuss, decide on, and document the GMT project governance structure (both for the Governing Council and Advisory Committee), finalize our draft code of conduct, contributor guide, and implement these (and post them on the website). We will design the user and developer short course material based on experience with those that were offered in 2019. The full OpenStreetMap data will be obtained and coastlines, including lakes, will be extracted and the processing into fillable tiles will commence [e.g., *Wessel and Smith*, 1996] which will involve the undergraduate student. The post-doctoral researcher will gain experience contributing to the C and Python codebase by being involved in the maintenance tasks, then transition to starting the work on C library refactoring in a new branch. Wessel and Luis start work on the GDAL OGR i/o design and implementation, while Wessel continues to design (but not yet implement) the keyword/value pairs long-option interface.
Year 2: Will see continued push to finalize the refactoring by the end of the second year. We will build unit tests and document the new API as the refactoring progresses. Start of the coordination of keyword/value pairs for all of core GMT to be consolidated with similar sets already used in PyGMT and GMT.jl. Draft implementation of OGR i/o functionality and first draft version of GSHHG 3 beta completed.
Year 3: Completion of C refactoring, the unit tests and API documentation. Completion of GDAL OGR i/o implementation. Completion of the implementation of long-format keyword/value alternate command options. Finalize and publish GSHHG 3. Merge development branch(es) into master and release GMT 7 stable release. Migrate PyGMT and GMT.jl to the new API in GMT 7.

It is the collective opinion of the GMT developers and steering committee that the completion of this plan will prepare GMT to be a reliable and easier-to-maintain computational infrastructure resource for the geoscience community. The project will require a more modest level of support (i.e., for future training and workshops) to maintain a vibrant and self-sustaining developer community.

7. Broader Impacts

As a critical software infrastructure, GMT has broader impact built in – we cannot avoid it! We already see lots of GMT maps on Wikipedia pages and in museum exhibits. Because GMT is used across so many scientific fields, but especially in the Earth and ocean sciences, the fundamental strengthening and improvements proposed herein will not only allow scientists to do more and better, but we expect will increase the number of users and developers. The refactoring may also have a broader impact since it will make it easier for other C programs to build on top of 30+ years of robust and performant GMT algorithms rather than starting from scratch. A strong and vibrant GMT community will be a huge benefit to the wider scientific community as we are only as great as we are together. The younger scientists are very community-oriented and we expect GMT to do very well in this regard. The project will provide essential training for a young post-doctoral researcher in modern computational software development, a skill set that is in high demand in and out of science. The close contact of the postdoc with the Governing Council and the Advisory Committee will provide valuable networking opportunities and insight into management and fundraising side of science. Finally, one or more undergraduate students will gain employment doing a scientifically challenging task related to their major.

Postscript: GMT was started by two students (Wessel and Smith) and over the next 30 years it grew into a huge community of users supported by a modest (6-7) group of core developers. Losing one out of 7 due to a pending retirement does not seem too dramatic. However, PI Wessel is the

most active of the developers (Fig. 7) and cannot easily be replaced by a single new recruit. It is essential that we cast a wide net to encourage many new developers so that tasks can be split across many instead of being focused on one person. That is the true challenge of this transition, which must take place against the backdrop of regular GMT maintenance and delivery; hence the urgency and focus of this proposal. Ideally, we hope some of the developers we will recruit will rise to take major leadership roles within GMT and guide the project into the future.

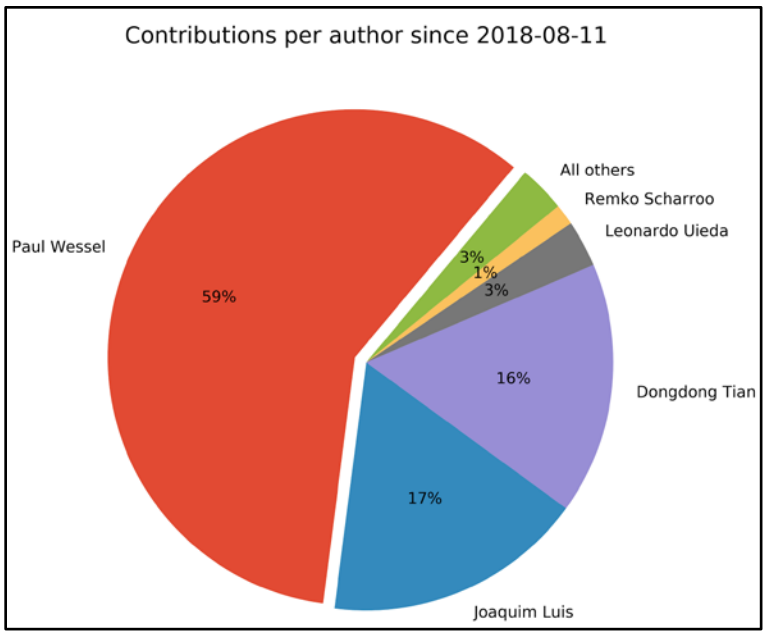

Fig.7. Number of changes made to the GMT source code (i.e., commits in the git repository) by each team member since the migration to GitHub on August 11, 2018. Note the display only examines the C code and the majority of Uieda's contributions are in the separate GitHub repository for PyGMT.

8. Results from Prior NSF Support

a) EAR 13-47184, $ 146,961, 5/1/14-4/30/18 incl. 1-year supplement [P. Wessel (UHM), D. Sandwell (UCSD), K. Feigl (U Wisconsin)].

b) Collaborative Research: Improving the Generic Mapping Tools for Seismology, Geodesy, Geodynamics and Geology.

c) Summary: Intellectual merit: Improved numerous GMT modules used in seismology, geodesy, and tectonics. We hardened computational aspects using OpenMP, added code verification via the Coverity service, added new modules for GPS vector gridding, ternary plots, and regression, and strengthened existing supplements for seismology, including adding support for plotting SAC files. Collaborated with Sandwell and Feigl to port GMTSAR to GMT 5 and positioned it to become an external GMT 6 supplement. We also participated in four GMTSAR workshops at UCSD, supported by UNAVCO. Finalized the GMT/MATLAB-Octave toolbox and made strides on the GMT/Python library (shown at 2017, 2018 Fall AGU), also supported via separate funding via an NSF CIF21 competition supplement to our OCE/MGG grant still active.

d) Broader impacts: The development and maintenance of GMT is all about broader impact since GMT is used across nearly all disciplines that fall under the Earth and Ocean Sciences. We expect the MATLAB and Python extensions will reach thousands of new users of GMT.

e) Publications:
Sandwell and Wessel [2016], *Wessel and Luis* [2017].

f) Products: Ongoing releases of GMT, the Generic Mapping Tools.

E. References

Caress, D. W., and D. N. Chayes (1996), Improved processing of Hydrosweep DS multibeam data on the R/V/ Maurice Ewing, *Mar. Geophys. Res.*, *18*, 631–650.

Doe, B. R. (1983), The past is the key to the future, *Geochimica et Cosmochimica Acta*, *47*(1341–1354), doi:10.1016/0016-7037(83)90293-4.

Douglas, D. H., and T. K. Peucker (1973), Algorithms for the reduction of the number of points required to represent a digitized line of its caricature, *The Canadian Cartograhper*, *10*(2), 112–122.

Gorny, A. J. (1977), World Data Bank II General User Guide, *Rep. PB 271869*, 10pp pp, Central Intelligence Agency, Washington, DC.

Key, R. M., A. Kozyr, C. L. Sabine, K. Lee, R. Wanninkhof, J. L. Bullister, R. A. Feely, F. J. Millero, C. Mordy, and T. H. Peng (2004), A global ocean carbon climatology: Results from Global Data Analysis Project (GLODAP), *Global Biogeochemical Cycles, 18*(4), doi:10.1029/2004gb002247.

Luis, J. F. (2007), Mirone: A multi-purpose tool for exploring grid data, *Computers & Geosciences*, *33*(1), 31–41.

Luis, J. F., and P. Wessel (2018), A Julia Wrapper for the Generic Mapping Tools, *Eos. Trans. AGU*, Fall Meeting, (Abstract NS53A-0563).

McClusky, S., Balassanian, S., Barka, A., Demir, C., Ergintav, S., Georgiev, I., Gurkan, O., Hamburger, M., Hurst, K., Kahle, H., Kastens, K., Kekelidze, G., King, R., Kotzev, V., Lenk, O., Mahmoud, S., Mishin, A., Nadariya, M., Ouzounis, A., Paradissis, D., Peter, Y., Prilepin, M., Reilinger, R., Sanli, I., Seeger, H.,m Tealeb, A., Toksöz, M. N., Veis, G.. (2000), Global Positioning System constraints on plate kinematics and dynamics in the eastern Mediterranean and Caucasus, *J. Geophys. Res., 105*(B3), 5695–5719, doi:10.1029/1999JB900351.

Müller, R. D., Seton, M., Zahirovic, S., Williams, S.E., Matthews, K.J., Wright, N.M., Shephard, G.E., Maloney, K.T., Barnett-Moore, N., Hosseinpour, M., Bower, D.J., Cannon, J. (2016), Ocean basin evolution and global-scale plate reorganization events since Pangea breakup, *Ann. Rev. Earth Planet. Sci., 44*(1), 107–138, doi:10.1146/annurev-earth-060115-012211.

Pallero, J. L. G. (2013), Robust line simplification on the plane, *Computers & Geosciences*, *61*(0), 152-159, doi: 10.1016/j.cageo.2013.08.011.

Ritsema, J., A. Deuss, H. J. van Heijst, and J. H. Woodhouse (2011), S40RTS: a degree-40 shear-velocity model for the mantle from new Rayleigh wave dispersion, teleseismic traveltime and normal-mode splitting function measurements, *Geophysical Journal International, 184*(3), 1223-1236, doi:10.1111/j.1365-246X.2010.04884.x.

Sandwell, D. T., and P. Wessel (2016), Interpolation of 2-D vector data using constraints from elasticity, *Geophys. Res. Lett.*, *43*, 10,703–710,709, doi:10.1002/2016GL070340.

Sandwell, D. T., R. Mellors, T. Xiaopeng, M. Wei, and P. Wessel (2011), Open radar interferometry software for mapping surface deformation, *Eos Trans. AGU, 92*(28), 234–235, doi:10.1029/2011EO280002.

Smith, D. E., Zuber, Maria T., Solomon, Sean C., Phillips, Roger J., Head, James W., Garvin, James B., Banerdt, W. Bruce, Muhleman, Duane O., Pettengill, Gordon H., Neumann, Gregory A., Lemoine, Frank G., Abshire, James B., Aharonson, Oded, David, C., Brown,, Hauck, Steven A., Ivanov, Anton B., McGovern, Patrick J., Zwally, H. Jay, Duxbury, Thomas C. (1999), The Global Topography of Mars and Implications for Surface Evolution, *Science, 284*, 1495-1503.

Smith, W. H. F., and D. T. Sandwell (2004), Conventional bathymetry, bathymetry from space, and geodetic altimetry, *Oceanography*, *17*(1), 8–23.

Smith, W. H. F., and P. Wessel (1990), Gridding with continuous curvature splines in tension, *Geophysics*, *55*(3), 293–305, doi:10.1190/1.1442837.

Soluri, E. A., and V. A. Woodson (1990), World Vector Shoreline, *Int. Hydrograph. Rev.*, *LXVII*(1), 27–35.

Steinberger, B., R. Sutherland, and R. J. O'Connell (2004), Prediction of Emperor-Hawaii seamount locations from a revised model of global plate motion and mantle flow, *Nature, 430*, 167–173, doi:10.1038/nature02660.

Tozer, B., D. T. Sandwell, W. H. F. Smith, C. Olsen, J. R. Beale, P. Wessel, and J. J. Becker (2019), Global bathymetry and topography at 15 arc seconds: SRTM15+V2.0, *Earth and Space Sciences*, in press.

Uieda, L., and P. Wessel (2017), A modern Python interface for the Generic Mapping Tools, *Eos. Trans. AGU*, Fall Meeting, (Abstract IN51B-0018).

Wessel, P., and J. F. Luis (2017), The GMT/MATLAB Toolbox, *Geochem. Geophys. Geosyst., 18*, 811–823, doi:10.1002/2016GC006723.

Wessel, P., and W. H. F. Smith (1991), Free software helps map and display data, *EOS Trans. AGU*, *72*(41), 441, doi:10.1029/90EO00319.

Wessel, P., and W. H. F. Smith (1995), New version of the generic mapping tools released, *Eos Trans. AGU*, *76*(33), 329, doi:10.1029/95EO00198.

Wessel, P., and W. H. F. Smith (1996), A global, self-consistent, hierarchical, high-resolution shoreline database, *J. Geophys. Res.*, *101*(B4), 8741–8743, doi:10.1029/96JB00104.

Wessel, P., and W. H. F. Smith (1998), New, improved version of Generic Mapping Tools released, *Eos Trans., AGU*, *79*(47), 579, doi:10.1029/98EO00426.

Wessel, P., W. H. F. Smith, R. Scharroo, J. F. Luis, and F. Wobbe (2013), Generic Mapping Tools: Improved version released, *Eos Trans. AGU*, *94*(45), 409–410, doi:10.1002/2013EO450001.