Heliyon 10 (2024) e26408

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

A case study of medical image software evolution and its impact in the medical imaging community

David Viar-Hernández, Borja Rodriguez-Vila, Mario Gil-Correa, Norberto Malpica, Ángel Torrado-Carvajal *

Medical Image Analysis and Biometry Laboratory, Universidad Rey Juan Carlos, Tulipán s/n, 28933, Madrid, Spain

A R T I C L E I N F O

Keywords:
Medical imaging computing
Open-source software
Software evolution

A B S T R A C T

Objective: We present the evolution of medical imaging software and its impact on the medical imaging community through the study of four open-source image analysis software platforms: 3D Slicer, FreeSurfer, FSL, and SPM. **Materials and methods:** We have studied the impact of these software tools over time, measured by the number of scientific citations. Additionally, we have also studied the source code evolution by measuring the lines of code and the tarball size of the stable releases and the changes in programming languages. **Results and discussion:** The rising number of related scientific publications confirms the popularity of these software tools in the research community, albeit some differences can be observed in the popularity of the tools. Moreover, we demonstrate that source code has evolved to modernize and optimize, at least partially thanks to the collaboration and code sharing with the user community. Furthermore, this evolution reveals an increased use of higher-level programming languages and meta-languages. **Conclusions:** The study of four open-source packages has revealed certain patterns in the evolution of medical imaging software and their impact on the medical image community. Further analyses and complementary metrics are suggested.

1. Background and significance

Medical image computing and computer-assisted intervention software platforms have experienced a huge development in different areas of diagnostic imaging such as image processing, visualization, and interaction, imaging, and analysis methods for image-guided therapies, and the evaluation of surgical and radiotherapeutic procedures [1,2]. The reason for such an amazing development is the inherent multidisciplinary nature of this field, where computational and mathematical methods are developed to solve problems related to medical images and their use for biomedical research and clinical care [3].

Several open source tools and applications such as 3D Slicer [4], Freesurfer [5], FSL [6], or SPM [7] have appeared in the last decades. These software tools are a complex combination of systems providing all the tools from the user interface to computation through data management. Additionally, large software systems must evolve, or they risk losing market share to competitors. Due to their complexity and the changing needs of their end-users, their source code has been continuously adapted to new needs and requirements, adding new features, system tuning, and fixing bugs.

* Corresponding author.
E-mail address: angel.torrado@urjc.es (Á. Torrado-Carvajal).

https://doi.org/10.1016/j.heliyon.2024.e26408
Received 30 March 2023; Received in revised form 12 February 2024; Accepted 13 February 2024

Available online 22 February 2024

However, maintaining such complex software platforms is extremely complicated, challenging, and time-demanding [13–15]. The efforts of including new features, adding support for new formats and platforms, system tuning, and debugging become more challenging as a software develops and grows. In this situation, community building is vital to the long-term viability of an open-source project, evolving and adapting to the advances in medical imaging modalities and image processing techniques.

2. Objective

Our objective is to study the evolution of medical imaging software and its impact on the medical imaging community through the evaluation of the four aforementioned and widely used image analysis software platforms: 3D Slicer, FreeSurfer, FSL, and SPM. We have studied how these platforms have influenced the medical imaging community by searching related publications in PubMed, Scopus, Web of Science, and Google Scholar. Moreover, the aforementioned platforms follow an open-source development approach that makes it easier to study the actual growth and changes in the code. We have measured the lines of code (LOC) as well as the tarball source code size growth of the stable releases, and the changes in programming languages over time. Our aim is to explain the behavior of the source code change and to compare these results with the point of view and acceptance of developers and users.

3. Materials and methods

3.1. Data under study

3.1.1. 3D Slicer

3D Slicer was born as a master's thesis project between the Surgical Planning Laboratory at the Brigham and Women's Hospital and the Massachusetts Institute of Technology (MIT) Artificial Intelligence Laboratory, US, in 1998. Currently, 3D Slicer is a free open-source software project developed and maintained by a wide community of experts in medical image analysis [4] and it is available for macOS, Windows, and Linux operating systems.

3D Slicer source code is released under the "3D Slicer Software License", a BSD-style open-source license compatible with the Open Source Definition by The Open Source Initiative. The 3D Slicer project has migrated to new control version systems several times, so different Slicer versions are fragmented in different repositories:

- Slicer1 and Slicer2 source codes were located in a Concurrent Versions System (CVS) repository.[1]
- Slicer3 source codes were located in a Subversion (SVN) repository[1].
- Slicer4 source codes are located in a web[2], while the developing version can be found in its official GitHub repository[3].

For this study, we have examined the 21 major and minor stable releases shown in Table 1 of Annex I of the supplementary material, from 2001 to 2021.

3.1.2. FreeSurfer

Freesurfer was created by the Laboratory for Computational Neuroimaging at the Athinoula A. Martinos Center for Biomedical Imaging (Massachusetts Institute of Technology, Massachusetts General Hospital, and Harvard Medical School). Currently, Freesurfer is an open-source software project developed by a community of developers and, mainly, by the Laboratory for Computational Neuroimaging, which manages the project supported by Massachusetts General Hospital, Boston MA, USA [5].

FreeSurfer software is released under the "FreeSurfer Software License Agreement" and it is available for macOS and Linux operating systems. Additionally, a not-fully-tested option exists for Windows 10 through the bash shell command-line tool. Nonetheless, this option lacks official support. All versions can be found online on the FreeSurfer webpage[4], while the current version is maintained on its official GitHub repository[5]. For this study, we have examined a total of 27 major and minor stable releases shown in Table 1 of Annex I of the supplementary material, from 2003 to 2021.

3.1.3. FSL

FSL The FMRIB Software Library (FSL) was born as an in-house developed set of tools that performed most of the steps in the fMRI analysis pipeline at the Functional Magnetic Resonance Imaging of the Brain (FMRIB) Center, Oxford University, UK, and first ever released in 2000. Currently, FSL is developed and maintained mainly by members of the FMRIB Analysis Group [6].

FSL is released under the GNU GPLv2. FSLView sources are released under the terms of the GNU General Public Licence version 2 (GNU GPLv2). FSL is available for download as binaries for Mac OS X and Linux (Centos or Debian/Ubuntu) -with Windows computers being supported with a Linux Virtual Machine. Source code is also available for those users who run an OS not directly supported by the FSL team. The different FSL versions are located on a webpage[6].

For this study, we have examined a total of 41 major and minor stable releases, as shown in Table 1 of Annex I of the supplementary material, from 2000 to 2021. FSL2 releases are not available for download, as they were internal development versions.

[1] These versions are deprecated and currently not available.

3.1.4. SPM

The Statistical Parametric Mapping software (SPM) was born at the MRC Cyclotron Unit, at the Hammersmith Hospital, London, to extract maps of t statistics in Positron Emission Tomography (PET) in what is known as SPM91 or SPMclassic. Currently, the SPM software is written and maintained by the Wellcome Department of Imaging Neuroscience at University College London, UK [7].

SPM is released under the GNU GPLv2. SPM is available for download for free for macOS, Windows, and Linux operating systems but requires a Matlab (The MathWorks) core version newer than 7.4 (R2007a) to run. There exists a standalone SPM version avoiding this requirement, but comes with limited functionality. Source codes for the different SPM versions can be downloaded from the SPM webpage or from its official GitHub repository[7]. For this study, we have examined 18 major and minor stable releases from 2000 to 2020, as shown in Table 1 of Annex I of the supplementary material. SPM96 and earlier versions are no longer available for download.

3.2. Methodology

3.2.1. Software impact evaluation

The impact of a specific software can be assessed by measuring the number of software sales or downloads. However, in the case of software aimed at research, its use in actual scientific studies may be a more accurate metric. A variety of publications were enabled by the release of these software tools. Thus, we have assessed the impact of these medical imaging software tools by searching the number of publications that mention them, either because they were used to visualize/analyze data or because a new module/extension was developed.

This impact can be partially assessed by searching with the different tools used in biomedical electronic research. We searched the official home pages of PubMed [8], Scopus [9], Web of Science [10], and Google Scholar [11] to identify and extract information regarding the number of publications citing the above-mentioned software suites and divided the results by year of publication. We used all these databases due to remarkable differences between them [12]. PubMed Central is an optimal database for full-text journal articles in biomedicine. Scopus covers a wider journal range, as well as the Web of Science. Google Scholar offers results of inconsistent accuracy but has no limits on the languages covered or the type of publication.

The aim of this work is not to develop a deep bibliographic research or a systematic review of the state of the art, but rather to represent general trends for the citation of the different software packages. Thus, we have defined some search terms and studied the obtained results, but we have not performed a manual purge of the results. The search terms used in the different databases are specified in Annex II.

3.2.2. Software source code metrics

The functional content of a software should be augmented over time in order to maintain user satisfaction. This increase in functionality will presumably be directly correlated with the additional amount of code included in the sources to improve or include new features, as stated by Lehman Laws of software evolution [13–15]. Thus, we have measured the total number of source lines of code (LOC). Moreover, we were also interested in the evolution of the dominant programming languages in each software and release. As programming language evolution continues, in both academia and industry, each project adopts the tools that are most convenient for its development. Particularly, specific languages, platforms, and frameworks have claimed to be the easiest and most efficient to use in certain scenarios, changing the usage trends [16]. To measure these two magnitudes, we have made use of the CLOC tool [17] and developed an ad-hoc script to iterate through the different software packages and releases. CLOC counts blank lines, comment lines, and physical lines of source code in many different programming languages and source files.

Additionally, as in many other classic studies about free software [18,19], we considered the size of each release tarball size as a complementary metric of the complexity evolution, due to its simplicity and immediacy. Thus, for each software and release in the study, we have studied the evolution of the tarball file size of the source code, downloaded from the available repositories, assuming these tarball files sometimes include documentation, images, and other utilities that may introduce artifacts in the final size.

4. Results

4.1. Software impact

Fig. 1 shows how the number of publications citing these tools has grown considerably (Figs. 1.A, 1.C, 1.E, and 1.G) and maintained an increasing rate over the last years (Figs. 1.B, 1.D, 1.F, and 1.H). These trends can be easily observed in the publication rate graphs, where values larger than 1 imply an increment in the annual number of publications, while values lower than 1 imply the annual number of publications is decreasing. Increase rates around 1 are associated with a stagnation phase in the number of publications.

3D Slicer and FreeSurfer present a similar number of publications in PubMed, Scopus, and Web of Science -around several hundreds per year-, while FSL and SPM show an order of magnitude less. However, these differences disappear when Google Scholar is used, where the four packages studied show similar values of the number of publications, around several thousands per year.

Analysis of the historical usage of 3D Slicer revealed a steady increase in scientific publications over the past two decades without hints of deceleration in all four databases. In the early 2000s, there were relatively few scientific publications citing the software, reflecting its nascent stage. However, over the past two decades, 3D Slicer has witnessed a steady increase in adoption and recognition. Notably, the growth rate slightly accelerated in the last five years, suggesting increased adoption and recognition within

the medical imaging community. This upward trend aligns with the software's evolving capabilities and its responsiveness to the evolving needs of the medical imaging field. As an open-source platform, 3D Slicer has also fostered a collaborative environment, attracting a diverse community of developers and users, which has likely contributed to its sustained growth and impact.

Analysis of FreeSurfer's historical usage revealed a substantial and enduring presence in the medical imaging landscape. While there was a modest uptake in scientific publications citing FreeSurfer in the early 2000s, the software experienced a notable surge in recognition from the mid-2000s onwards, but starting to show a small slowdown in the evolution of its increasing rate and the associated stabilization of the number of publications per year. This surge coincides with FreeSurfer's continuous development, which has expanded its capabilities for structural neuroimaging analysis. The medical imaging community has increasingly turned to FreeSurfer for its robust tools in brain segmentation, cortical surface reconstruction, and volume measurements. Its enduring significance underscores its reliability as a fundamental tool for researchers working in the realm of neuroimaging.

Analysis of the historical usage of FSL unveiled a compelling story of consistent growth and influence. Since its inception, FSL has been on an upward trajectory in terms of scientific publications, but starting to show a small slowdown in the evolution of its increasing rate and the associated stabilization of the number of publications per year, as it is happening with FreeSurfer's. This trajectory reflects FSL's adaptability and versatility, with an array of tools for functional, structural, and diffusion MR imaging analysis. Its appeal extends to researchers across various subfields of medical imaging, contributing to its broad and sustained impact. The software's comprehensive documentation and active user community have further facilitated its integration into research pipelines, making it a go-to choice for many in the medical imaging community.

Analysis of SPM's historical usage reaffirmed its enduring status as a cornerstone tool in the field of medical imaging. Over the past two decades, SPM has consistently maintained a high number of scientific publications, despite showing a maintained constant publication rate of 1 since early 2000s. Its appeal lies in its robust statistical methods for the analysis of neuroimaging data, enabling researchers to identify subtle brain activation patterns and structural changes. The widespread adoption of SPM is a testament to its reliability and effectiveness, making it an indispensable resource for studies involving functional MR, PET, and other imaging modalities.

Overall, these results confirm the acceptance of such tools by the medical imaging community, although some of them may have reached or are reaching their peak of popularity while others still present a clear increment. This fact could be, in part, due to the appearance of new software packages and tools for the same specific purpose (neuroimaging mostly), while 3D Slicer covers a wider spectrum of applications and there are no similar open-source alternatives in its niche of application.

Collectively, our findings underscore the substantial and enduring impact of these four key medical image software packages within the medical imaging community. The consistent growth in scientific publications for each software package reflects their pivotal role in advancing the field. Furthermore, the accelerated adoption and recognition observed in recent years suggest that these tools continue to shape and influence research in medical imaging, with wide-ranging implications for the broader healthcare and scientific communities.

4.2. Continuing growth

As described in the materials and methods section, we used the total number of LOC and the tarball file size for each release of each software to measure the continuous growth of the softwares. Fig. 2 shows the evolution of the number of source LOC (Figs. 2.A, 2.C, 2.E, and 2.G) and the size evolution of the compressed tarball files (Figs. 2.B, 2.D, 2.F, and 2.H) over time for the different stable releases.

The two metrics, LOC and tarball size, show positive but very different trends over the different software lifetimes. The positive trend in both growth metrics is due to the fact that the functional content of a software must be constantly changed and augmented to preserve user satisfaction over its lifetime, not only to adjust to varying circumstances but also to increase with completely new capabilities that address new demands from the community. It is interesting that, for all of them, LOC measurements seem to show that releases within the same version (and sometimes even between different versions) are growing at a very linear rate over time. This challenges Lehman laws, stating growth would be either linear or most likely sub-linear in its revisited version, by showing how medical imaging software continues growing and adapting to new developments in the field.

It is important to also note that most of the analyzed software tools present abrupt changes in LOC when upgrading to a newer version, e.g.: Slicer4, FreeSurfer5, FreeSurfer6, and FSL6. These changes may be subject to another important Lehman Law stating that continuous growth increases the complexity of the code, also linking with another Law that considers uncontrolled continuous change as a potential source of declining quality of the software. As open-source projects mostly developed by a community, the complexity of the different software packages studied greatly increased over time, leading to reengineering processes and changes to support new source code architecture, rather than defect fixing.

These general trends can be observed in more detail in the interactions of developers in the respective GitHub repositories. Graphs of lines of code added and removed over time (see Fig. 3) allow us to observe the moments in which large structural changes have been carried out in software packages that have a more open developer community, like 3D Slicer (Fig. 3.A) and FreeSurfer (Fig. 3.B). On the other hand, packages with more closed developer communities do not use GitHub as a development repository, but instead upload the different versions once they are finalized and debugged (like SPM, in Fig. 3.C) or do not have an official GitHub repository (FSL).

Slicer, in its fourth version, represents a special case because the tarball size decreases dramatically in both LOC and tarball size; this is due to a new structure where the release mainly includes the backbone, while the rest of the extensions, add-ons, images,

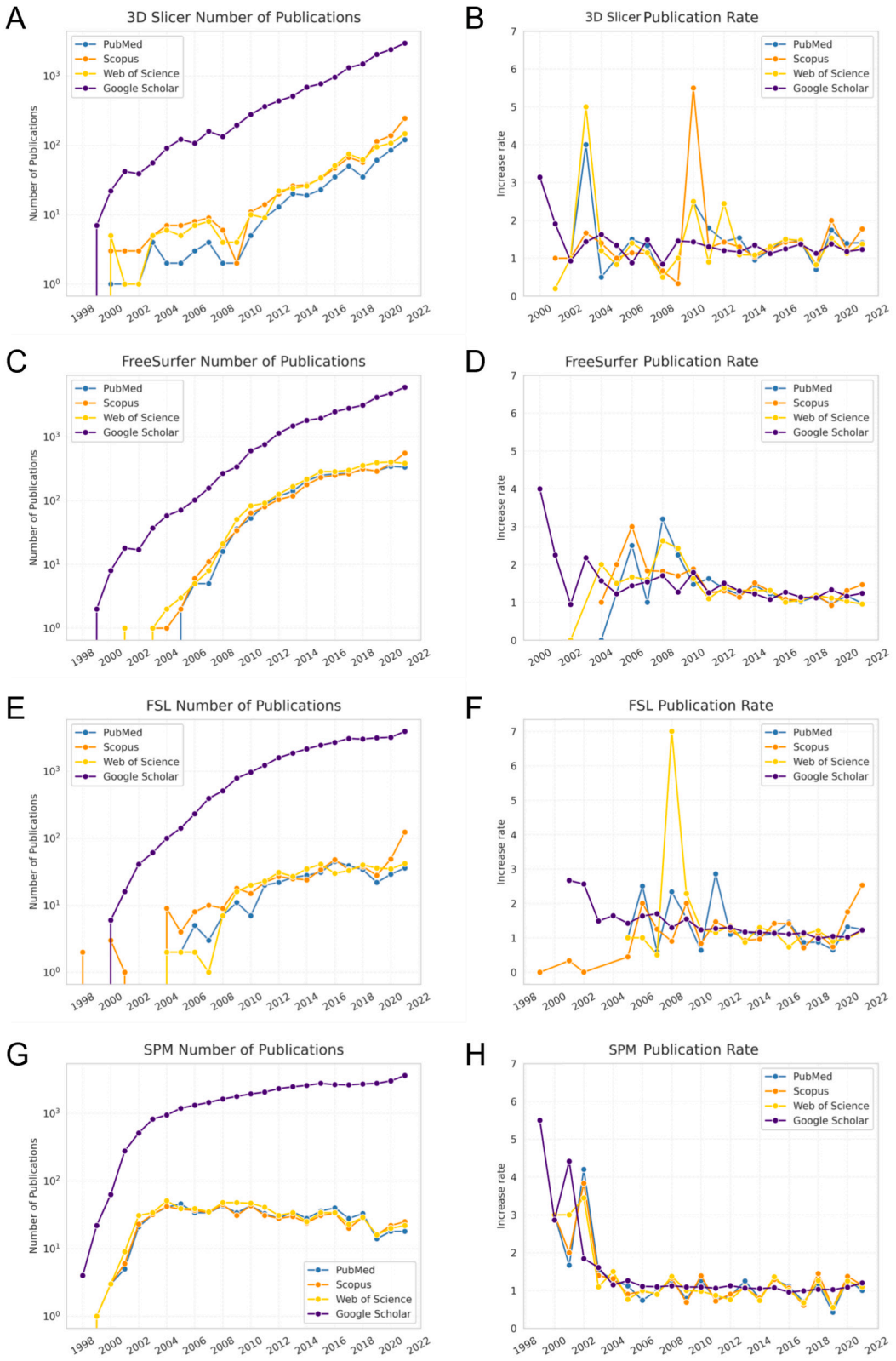

Fig. 1. Absolute number of publications per year (A, C, E, and G) and relative publication rate compared to the previous year (B, D, F, and H) as reported by PubMed, Scopus, Web of Science, and Google Scholar for 3D Slicer (A and B)), Freesurfer (C and D), FSL (E and F) and SPM (G and H).

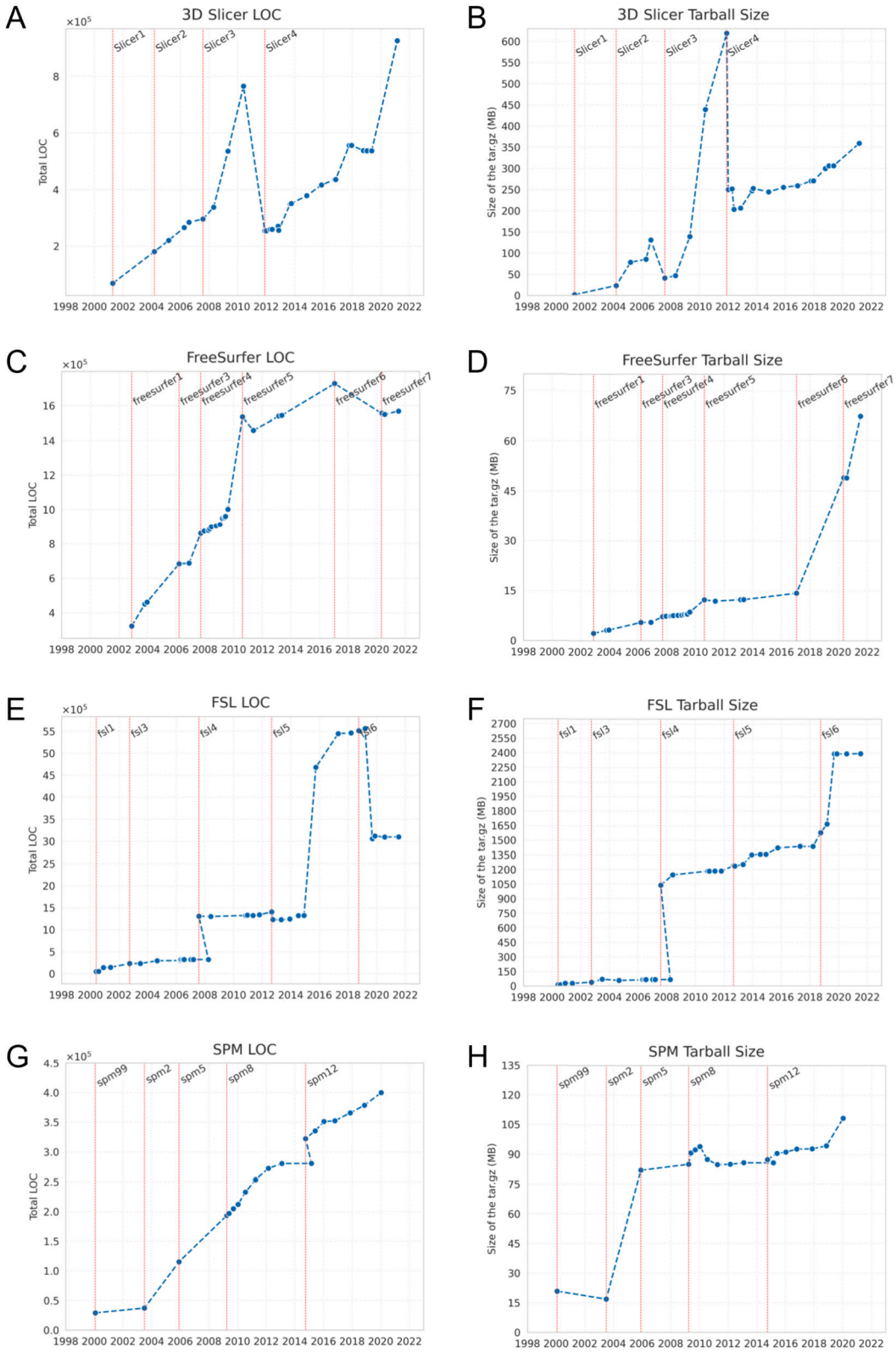

Fig. 2. Lines of code (A, C, E, and G) and tarball size (B, D, F, and H) metrics for 3D Slicer (A and B), Freesurfer (C and D), FSL (E and F) and SPM (G and H).

etc., can be downloaded as needed. However, the tarball size of newer versions is still around half the size of version 4.0, while the number of LOC has significantly exceeded the previous maximum.

The exponential growth of 3D Slicer is clearly represented in Fig. 1 and Fig. 2. Furthermore, Fig. 4 shows how the number of 3D Slicer extensions has continuously grown since its beginning in Slicer 4.1.1. Changes of release have implied the discontinuation of some of these extensions, but most of them have remained available and new extensions are included on each version. If we consider all the Slicer extensions that have arisen in the last years, available in a repository hosted in Github,[2] this value should be much larger. However, we have not studied the total LOC associated with the extensions for the different releases.

4.3. Source code evolution

As we mentioned before, we were also interested in the evolution and changes of the dominant programming languages in each software and release. Fig. 5 shows the evolution of the different programming languages in the source files case of study.

Most of the software tools analyzed showed one dominant programming language, that maintained its relevance throughout all versions. However, as programming language evolution continues, each project adopts the tools that are most convenient for its adaptation and growth. If we focus on the different platforms, we can see similar trends and changes supporting Lehman Laws of continuing change.

In 3D Slicer (Fig. 5.A), we can see how Tcl has been losing its dominance against C++. The most remarkable change is due to the reorganization of the Slicer architecture from Slicer2 to Slicer3. During that period C++ reached about 90% of dominance over 3D Slicer code, becoming the leading language. Slicer3 also allowed the use of Python to write scripted modules. The changes in Slicer4 have promoted the use of other languages. In this version, the GUI modules are described in XML, while the program logic is written in Python; hence, these languages have significantly increased their presence with respect to previous versions.

In Freesurfer (Fig. 5.B), C & C++ represented almost 100% of the code originally; however, that changed when released as Freesurfer2, including more C shell and Tcl scripts, as well as Matlab code. In this case, C & C++ have maintained their supremacy over the years, showing a stabilized value of around 80%. Nevertheless, we can appreciate how other programming languages have very slowly been gaining relevance over time, although none of them has reached 10%. This seems to have had an effect on decreasing the LOC with the introduction of FreeSurfer6 in 2017, which is not visible in the tarball size metric that shows a significant increment.

As for 3D Slicer and FreeSurfer, in FSL (Fig. 5.C) there is a clear predominance of C/C++ usage, although presenting lower values than in the other cases. In this case, it is interesting to emphasize the great variability of some secondary languages such as Tcl, decreasing from 30% to almost disappearing, or HTML, with significant positive and negative variations along the project, due to its role in the creation and visualization of the reports offered after the processing. The increment and posterior decrement in the use of HTML seem to have a reflection in the LOC metric, but these trends do not appear in the tarball size evolution.

In SPM (Fig. 5.D), Matlab has been the predominant programming language since the beginning of the project, and its leading position is maintained along the software lifetime and stabilized slightly below 90%. The usage of C/C++ started to decrease with the introduction of SPM5 and, in general, the use of secondary programming languages shows negative trends.

5. Discussion

In this work, we present a case study of medical image software evolution and its impact on the medical imaging community through the analysis of the 3D Slicer, Freesurfer, FSL, and SPM softwares. Specifically, we analyzed the aforementioned softwares evolution in light of software impact as well as data regarding their continuing growth and source code evolution.

We measured the impact of each software package by searching the number of publications that mention them in PubMed, Scopus, Web of Science, and Google Scholar. The number of scientific papers published every year and for each software package is very similar for PubMed, Scopus, and Web of Science, but significantly higher for Google Scholar. Moreover, the general trend also seems slightly different between Google Scholar and the other databases. For example, the slowdown, or even the reduction, in the number of publications per year that can be observed in the first three databases is not seen in the same way in the graphs associated with Google Scholar. However, the increase rate shows a better correlation between all cases, although with smoother variations for Google Scholar, associated with a much higher number of articles (two orders of magnitude of difference for SPM and FSL).

The graphs of the number of publications per year show that 3D Slicer is still in a phase of exponential increment and there is no sign of deceleration yet. Meanwhile, FreeSurfer and FSL seem to be reaching a phase of stabilization while SPM reached its peak around 2004 and it is starting a negative trend. These differences may be linked to the inherent characteristics of each of the platforms. While FreeSurfer, FSL, and SPM are oriented to neuroimaging, 3D Slicer is a multi-purpose platform offering general image visualization, analysis, and processing tools useful in several different applications. Also, the need for a Matlab license for using SPM may be a negative factor compared with the other software packages, which do not need any previous expenditure.

In order to evaluate the continuing change and growth we considered the size of each version as the measured total number of source LOC, as well as the tarball file size. This double metric was explored to assess the real change and evolution of medical imaging software. In this specific case, we can assure that measuring the tarball size is not a good parameter to observe the growth of this specific type of software. Different releases have included processing algorithms such as atlas-based segmentation, which require

[2] https://github.com/Slicer/ExtensionsIndex.

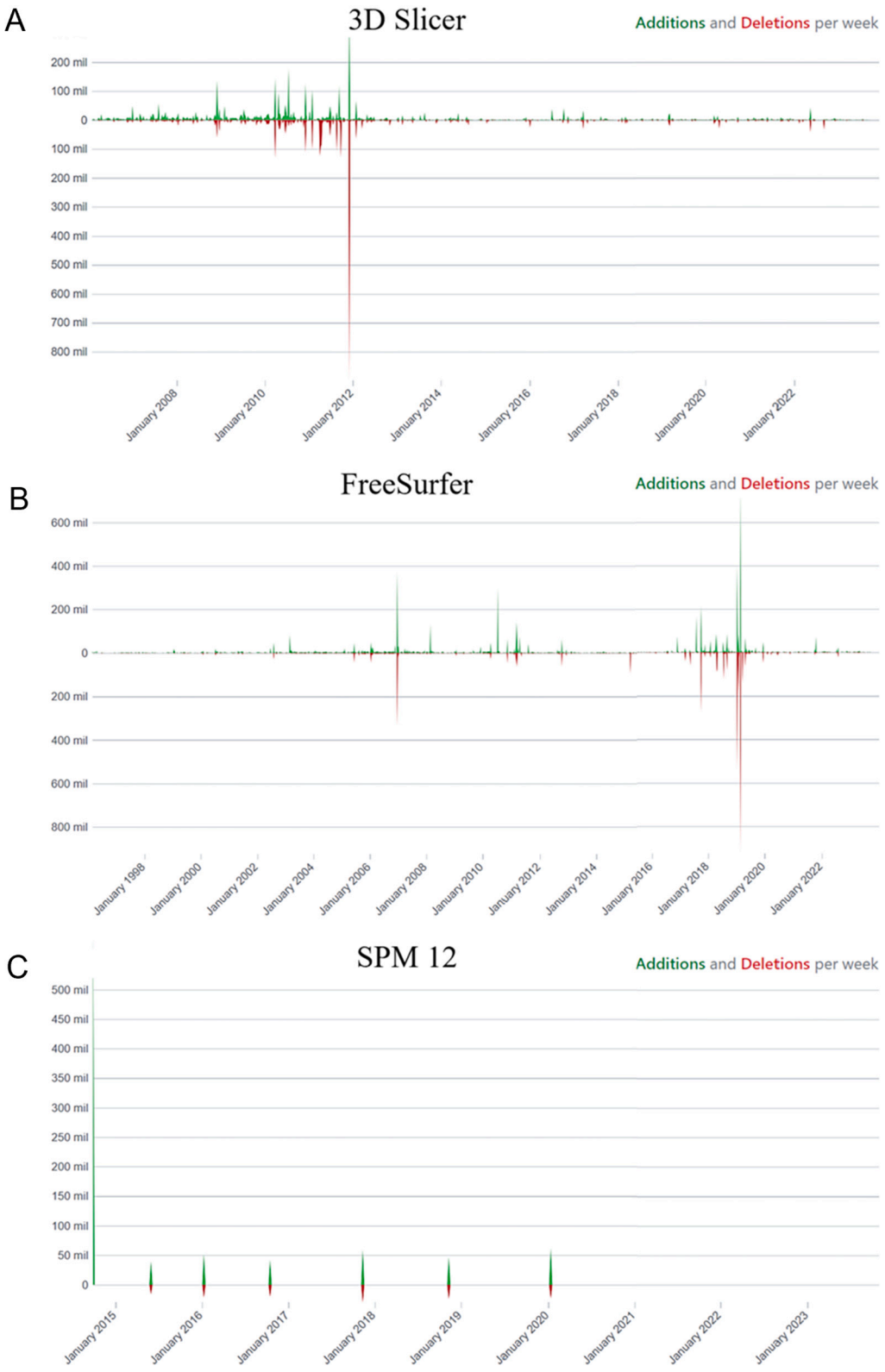

Fig. 3. Evolution of the additions and deletions of the GitHub repositories of A) 3D Slicer, B) FreeSurfer, and C) SPM.

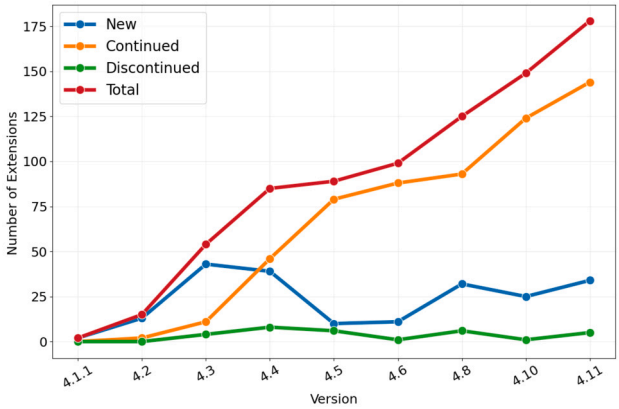

Fig. 4. Number of extensions of 3D Slicer along releases of Slicer4.

atlases containing images or reference volumes included in the source directory, implying abrupt increments in the file size (e.g.: Slicer3, FreeSurfer6, FSL4, and SPM2) and hiding changes in the source code. This implies that increments or decrements in LOC, associated with architectural or design changes, do not reflect or are hidden in the tarball size metric.

Thus, we consider that a better measure of growth is shown by the number of LOC, as this metric focuses on source code and eliminates the bias related to the inclusion of images and templates for image processing. However, even this metric presents drastic changes in 3 out of 4 of the analyzed software packages. These changes could be explained by the need for a reengineering process and changes in the architecture of the source codes. For example, HTML usage in FSL influenced dramatically in the total number of LOC. Also, Slicer4 changed the GUI description and implementation from previous versions, and turned into a modular architecture, including an extension mechanism that decouples modules from the main source code, to allow better modularity and third-party plug-ins.

We also noticed that in early versions, changes over time in 3D Slicer and Freesurfer were more noticeable than in FSL and SPM. This could be due to the "openness" of the projects, as FSL and SPM are "copyleft" software (meaning they are free but copyright software), while 3D Slicer and Freesurfer have created a whole collaborative community that contributes to the development of the different packages, in the shape of extensions and/or scripts. Thus, the rapid rise in the popularity of medical imaging software has resulted in a large amount of contributed stable code. In recent years, FSL has also performed an optimization of the source code, reducing the weight of HTML in the source code and resulting in an abrupt reduction of the number of LOC.

In both metrics, tarball file size, and LOC, some retracements can be observed for FSL4 and SPM12, which makes little sense in a time series. This is because, in those periods of time, two versions of the software coexist until the older version is no longer supported.

Programming language distribution has been constantly changing to adapt and help in increasing functionality while avoiding increasing complexity and declining quality. Taking into account the LOC and the Language Distribution together demonstrates the distribution of languages changes more dramatically when the total LOC increases more significantly. These changes show and support the need for reengineering processes and changes to support a new source code architecture and include more functionality. Additionally, fluctuations and preferences towards certain programming languages over time show an increase in higher-level programming languages (e.g., Python) as well as meta-languages (e.g., XML, HTML, TeX), which is directly correlated with the trends stated by the TIOBE programming community index [20].

Several limitations of this study should be pointed out. As an exploratory study, we focused on the study of software impact, continuing growth as well as source code evolution; however, additional metrics (i.e., number of functions, McCabe's index, Oman's index, added, deleted, grown, or shrunk files) could be calculated and further analysis could be performed. Future research should take into consideration these new metrics, whose inclusion would give insights into the actual complexity, maintainability, stability, and actual change of the different software packages analyzed in this study. Additionally, we studied the source code as a single unit, while a more detailed study could provide views on medical imaging algorithm tendencies (i.e., registration, segmentation). Furthermore, studying the number of publications citing these softwares is just a portion of information regarding the impact of these softwares in the community. The broader inclusion activity of the community in the repositories (i.e., contributions, commits), the activity of the users (i.e., number of searches, downloads), or their geographical distribution could provide a more detailed understanding of their importance, reach, and impact. In addition, it would be interesting to include cloud computing and transitioning of medical imaging to the cloud as a covariate interacting with these softwares in recent years. However, despite these limitations, our preliminary results suggest several common trends in medical imaging software evolution and their impact on the medical imaging community.

6. Conclusions

The study of 3D Slicer, Freesurfer, FSL, and SPM has revealed certain patterns in the evolution of medical imaging software and their impact on the medical image community. On one hand, the evolution of the number of scientific papers citing the studied

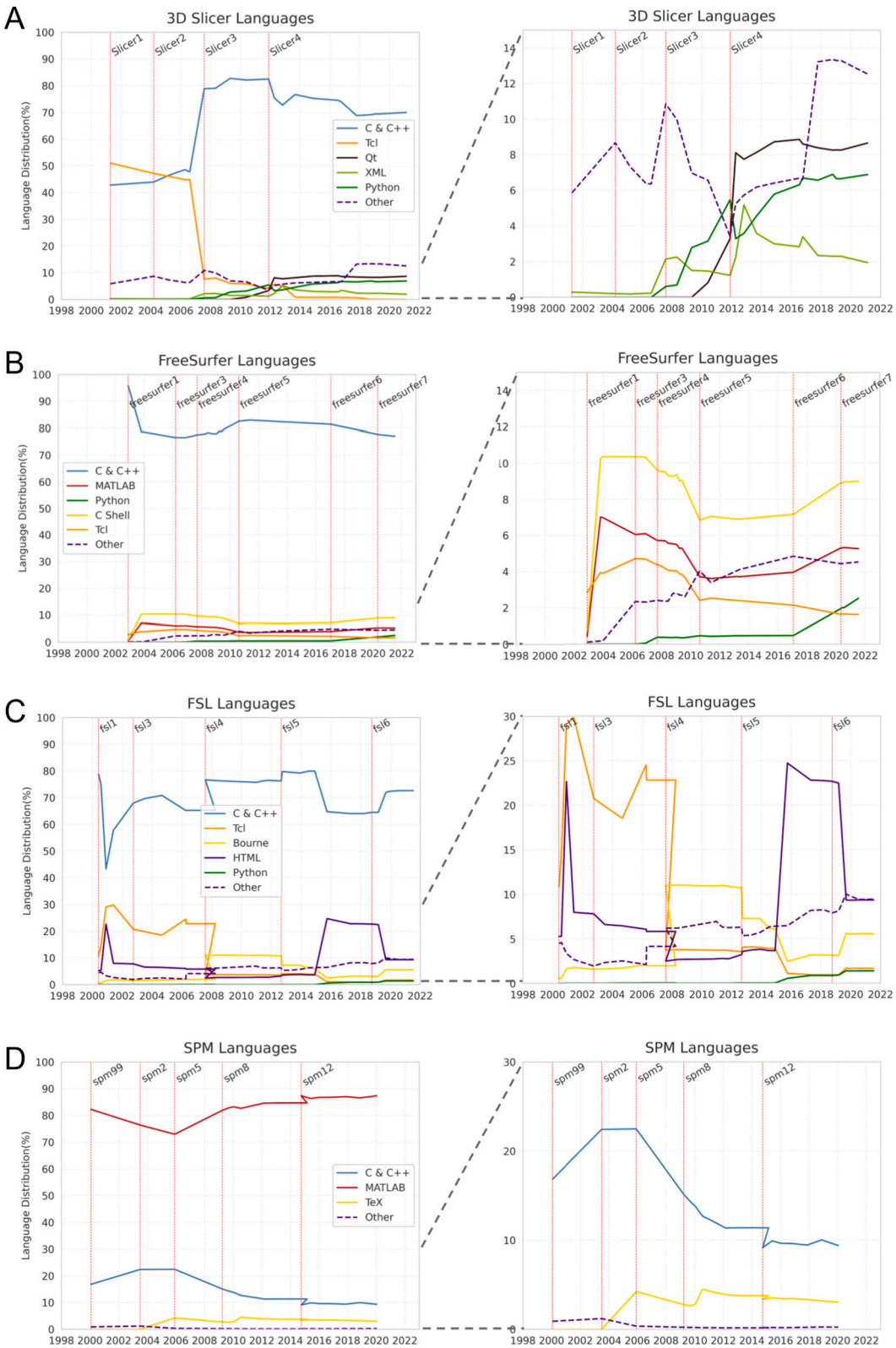

Fig. 5. Percentage distribution of the different programming languages evolution through releases for A) 3D Slicer, B) Freesurfer, C) FSL, and D) SPM. The right column shows an expanded visualization of the secondary programming languages for better visualization.

softwares confirms the acceptance of these packages by the medical imaging community, albeit some differences can be observed in the popularity of the tools. Additionally, we have proven that Google Scholar should not be used as a reliable source for this type of study. On the other hand, these findings show how the software packages' evolution through the different versions, and their constant and deep architectural changes between major versions, are the variables that offer more interesting information. Additionally, the development and evolution of these softwares have been accompanied by their use. Source code has changed over time to modernize, optimize, introduce new technologies and algorithms, and occasionally slim down. The study of the source code shows how this evolution has been possible due to the improvement in collaboration and code sharing with the medical imaging research community. This evolution also reveals that tendencies in programming languages are changing and other high-level programming languages are becoming more popular.

A further software engineering analysis including a study of the complexity, maintainability, stability, and actual change among versions can be performed. A deeper analysis of the different kinds of modules may include valuable information about medical imaging algorithm trends. Future work may also support the community findings by including the activity of the community in the repositories, the activity of the users, or their geographical distribution.

CRediT authorship contribution statement

David Viar-Hernández: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Borja Rodriguez-Vila:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Mario Gil-Correa:** Software. **Norberto Malpica:** Supervision, Project administration, Methodology, Conceptualization. **Ángel Torrado-Carvajal:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.heliyon.2024.e26408.

References

[1] K. Doi, Diagnostic imaging over the last 50 years: research and development in medical imaging science and technology, Phys. Med. Biol. 51 (13) (2006) R5–R27.
[2] K. Doi, Computer-aided diagnosis in medical imaging: historical review, current status and future potential, Comput. Med. Imaging Graph. 31 (4) (2007) 198–211.
[3] J.S. Duncan, N. Ayache, Medical image analysis: progress over two decades and the challenges ahead, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 85–106.
[4] R. Kikinis, S.D. Pieper, K.G. Vosburgh, 3D Slicer: a platform for subject-specific image analysis, visualization, and clinical support, in: Intraoperative Imaging and Image-Guided Therapy, Springer, New York, 2014, pp. 277–289.
[5] Fischl B. FreeSurfer, Neuroimage 62 (2) (2012) 774–781.
[6] M. Jenkinson, C.F. Beckmann, T.E. Behrens, et al., FSL, NeuroImage 62 (2012) 782–790.
[7] K.J. Friston, J.T. Ashburner, S.J. Kiebel, et al., Statistical Parametric Mapping: the Analysis of Functional Brain Images, Elsevier/Academic Press, 2007.
[8] NCBI: PubMed, http://www.ncbi.nlm.nih.gov/pubmed. (Accessed April 2022).
[9] Scopus: scopus database, http://www.scopus.com. (Accessed April 2022).
[10] Thomson Reuters: web of science, http://scientific.thomson.com/products/wos. (Accessed April 2022).
[11] Google Inc.: Google Scholar http://scholar.google.com. (Accessed April 2022).
[12] M.E. Falagas, E.I. Pitsouni, G.A. Malietzis, et al., Comparison of PubMed, scopus, web of science, and Google scholar: strengths and weaknesses, FASEB J. 22 (2) (2008) 338–342.
[13] M.M. Lehman, L.A. Belady, Program Evolution: Processes of Software Change, Academic Press Professional, 1985.
[14] M.M. Lehman, Laws of software evolution revisited, in: European Workshop on Software Process Technology, Springer, 1996, pp. 108–124.
[15] M.M. Lehman, J.F. Ramil, P.D. Wernick, et al., Metrics and laws of software evolution-the nineties view, in: IEEE Software Metrics Symposium, 1997, pp. 20–32.
[16] A. Daniel, CLOC count lines of code, https://github.com/AlDanial/cloc, Release: v1.74 (January 14, 2017).
[17] S. Cass, The 2017 top ten programming languages, IEEE Spectr. (July 18, 2017).
[18] M. Godfrey, Q. Tu, Evolution in open source software: a case, in: Proceedings of the 2000 International Conference on Software Maintenance (ICSM'00), 2000.
[19] M.M. Simmons, P. Vercellone-Smith, P.A. Laplante, Understanding open source software through software archaeology: the case of nethack, in: 2006 30th Annual IEEE/NASA, Software Engineering Workshop, Columbia, MD, USA, 2006, pp. 47–58.
[20] https://www.tiobe.com/tiobe-index/. (Accessed April 2022).