STATE OF PRACTICE FOR MEDICAL IMAGING SOFTWARE

ASSESSING THE CURRENT STATE OF THE PRACTICE FOR MEDICAL IMAGING SOFTWARE

By

AO DONG.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the degree

Master of Engineering in Computing and Software

McMaster University

MASTER OF ENGINEERING (2021) McMaster University

(Computing and Software) Hamilton, Ontario

TITLE: Assessing the Current State of the Practice for Medical Imaging Software

AUTHOR: Ao Dong

SUPERVISOR: Dr. Spencer Smith

NUMBER OF PAGES: viii, **??**

Abstract

Abstract here

Acknowledgments

acknowledgements here

Contents

Chapter 1

Introduction

introduction

scientific computing (SC), also known as scientific computation or computational science

1.1 Motivation

1.2 Purpose

1.3 Scope

Chapter 2

Background

2.1 Software Categories

In this section, we discuss three common software categories that are mentioned in Section 3.2, and also SC software.

2.1.1 Open Source Software

For Open Source software (OSS), its source code is openly accessible, and users have the right to study, change and distribute it under a license granted by the copyright holder. For many OSS projects, the development process is based on the collaboration of different contributors worldwide [3].

2.1.2 Freeware

Freeware is software that can be used free of charge. Unlike with OSS, the authors of freeware typically do not allow users to access or modify the source code of the software [11]. The term *freeware* should not be confused with *free software*, which is similar to OSS but with a few differences.

2.1.3 Commercial Software

"Commercial software is software developed by a business as part of its business" [5]. Typically speaking, the users are required to pay to access all of the features of commercial software, excluding access to the source code. However, some commercial software is also free of charge [5].

2.1.4 Scientific Computing Software

Software development in Scientific Computing (SC) depends on the knowledge of three areas - the knowledge of a specific engineering or science domain, the ability to mathematically build models and applying algorithms, and the capability to implement theoretical models and algorithms with computational tools. SC software is built with mathematical and computational tools to serve the purpose of solving scientific problems in a domain [9].

2.2 Software Quality Definitions

The definitions of software qualities are from Smith et al. [14]. The order of the qualities follows the grading template in Appendix A.

- **Installability** The effort required for the installation, uninstallation, or reinstallation of a software or product in a specified environment.

- **Correctness & Verifiability** A program is correct if it behaves according to its stated. Verifiability is the extent to which a set of tests can be written and executed, to demonstrate that the delivered system meets the specification.

- **Reliability** The probability of failure-free operation of a computer program in a specified environment for a specified time, i.e. the average time interval between two failures also known as the mean time to failure (MTTF).

- **Robustness** Software possesses the characteristic of robustness if it behaves "reasonably" in two situations: i) when it encounters circumstances not anticipated in the requirements specification, and ii) when the assumptions in its requirements specification are violated.

- **Usability** The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

- **Maintainability** The effort with which a software system or component can be modified to i) correct faults; ii) improve performance or other attributes; iii) satisfy new requirements.

- **Reusability** The extent to which a software component can be used with or without adaptation in a problem solution other than the one for which it was originally developed.

- **Understandability** (To be completed)

- **Visibility & Transparency** The extent to which all of the steps of a software development process and the current status of it are conveyed clearly.

2.3 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) was developed by Thomas L. Saaty, and it has been widely used to make and analyze multiple criteria decisions [16]. The AHP organizes multiple criteria factors in a hierarchical structure and pairwise compares the alternatives to calculate relative ratios [12].

For a project with m criteria, we can use a $m \times m$ matrix A to record the relative importance between factors. By pairwise compare criterion i and criterion j, the value of A_{ij} is decided as follows, and the value of A_{ji} is $1/A_{ij}$ [12],

- $A_{ij} = 1$ if criterion i and criterion j are equally important;
- $A_{ij} = 9$ if criterion i is extremely more important than criterion j;
- A_{ij} equals to an integer value between 1 and 9 according the the relative importance of criterion i and criterion j.

The above process assumes that criterion i is not less important than criterion j, otherwise, we need to reverse i and j and determine A_{ji} first, then $A_{ij} = 1/A_{ji}$.

The priority vecotr w can be calculated by solving the following equation [12],

$$Aw = \lambda_{max} w, \tag{2.1}$$

where λ_{max} is the maximal eigenvalue of A.

In this project, w is approximated with the approach classic *mean of normalized values* [8],

$$w_i = \frac{1}{m} \sum_{j=1}^{m} \frac{A_{ij}}{\sum_{k=1}^{m} A_{kj}} \tag{2.2}$$

Suppose there are n alternatives, for criterion $i = 1, 2, ..., m$, we can create an $n \times n$ matrix B_i to record the relative preferences between these choices. The way of generating B_i is similar to the one for A. However, unlike comparing the importance between criteria, we pairwise decide how much one alternative is more favored than the other. The same method is used to calculate the local priority vector for each B_i.

In this project, the 9 software qualities mentioned above are the criteria ($m = 9$), while 29 software packages ($n = 29$) are compared. The software are evaluated with the grading template in Appendix A and a subjective score is given for each quality. For a pair of qualities or software, i and j, such that i is not less significant than j, the pairwise comparison result of i versus j is converted from $min((score_i - score_j) + 1, 9)$.

Chapter 3

Methods

[Where to include the domain experts? I plan to include them in Domain Selection and Software Product Selection, and also mention that they'll participate in interviews. —AD]

In this project, we aim to study and research the SC software within scientific domains. However, we try to design the whole process of this project in a more general way, with the hope that it should not be limited to only one or several specific software categories.

The way how we choose a domain is listed in Section 3.1. Within a specific domain, we use the process documented in Section 3.2 to collect and filter the software packages for our study. Then, all the software is measured by using the grading template in Section 3.3 and the empirical measurement method in Section . Section 3.5 presents more details about how we apply the above process.

3.1 Domain Selection

Although the methods of this project may not have such limitations, we limit our selection within scientific domains to fulfill the objective of our research.

One major factor in properly choosing a candidate domain for the study is the ease to select software packages within it. As described in Section 3.2, the software product selection process is most likely to have a screening step, and the quantity of final-decision packages may not meet our initial expectation if we do not have enough software candidates to choose from. Section 3.2 also explains why our process prioritize the OSS. Consequently, a scientific domain with a large number of active OSS is proffered. This also often indicates that the domain has an active community developing and using SC software, making it easier to conduct interviews mentioned in Section 3.6.

Even if we can find an adequate number of software packages in a domain, we may still find the software products are developed to solve various problems within the domain and should be categorized into different sub-groups. So one question needs to be asked - do we prefer a group of software all providing similar functions and features, or do we aim to cross-compare several sub-sections within the same domain? With that answered, it should be easier to determine a favored domain.

Another aspect to consider is the team carrying out the research, more specifically, the domain experts - if there is any - in the team. In our team, the projects are often led by researchers in the software engineering field and supported by experts working in other

scientific domains. Having domain experts in the team provides significant benefits in selecting software packages and designing interview questions.

In this project, Medical Imaging (MI) domain is selected. Numerous software products can be found in this domain, and a great number of domain experts use - and even develop - such software. We decide to focus on MI software with the viewing function, and more details about this filtering are in Section 3.2. Being able to include MI domain experts in our team also makes this domain more preferred.

3.2 Software Product Selection

The process of selecting software packages contains two steps: i) identify software candidates in the chosen domain, ii) filter the list according to needs [15].

3.2.1 Identify Software Candidates

The candidate software can be found from publications in the domain. Another source is to search various websites, such as GitHub, swMATH and the Google search results for software recommendation articles. Meanwhile, we should also include the suggested ones from domain experts [15].

As for this project, 48 MI software projects are identified as the candidates, and they are found from publications [1] [2] [6], online articles related to the domain [4] [7] [10],

forum discussions related to the domain [13], etc.

3.2.2 Filter the Software List

The goal is to build a software list with a length of about 30 [15].

The only mandatory requirement is that the software must be open source, as defined in Section 2.1.1. This is due to the grading process defined int Section 3.3. In order to evaluate all aspects of some software qualities, the source code must be accessible.

The other factors to filter the list can be optional and should be considered according to the number of software candidates and the objectives of a research project.

One of the factors is the functions and purposes of the software. For example, we can choose a group of software with similar functions, so that the cross-comparison is between each individual of them. On the other hand, if our objective is comparing sub-categories in the domain, we should select from candidates in each of the categories.

The empirical measurement tools listed in Section 3.4 are limited to projects using Git as the version control tool, so software with Git is preferred. Some manual steps in empirical measurement depend on a few metrics of GitHub, which makes projects hold on GitHub more favored [15].

Some of the OSS projects may experience a lack of recent maintenance. So packages that have not been updated for a long time can be eliminated, unless they are still popular and highly recommended by the users in the domain [15].

In this project, we focus on the MI software providing imaging viewing function. [TBC —AD]

3.3 Grading Template

[The grading template for manual measurements —AD]

3.4 Empirical Measurements

[Explain the empirical tools and how they affect the scores. —AD]

3.5 Measuring Qualities

[e.g. virtual machines, time spent per software, where to look for docs, etc. —AD]

3.6 Interview Methods

3.6.1 Interviewee Selection

[Generally speaking, ask all the teams which we can find contacts, and continue with the ones who are willing to participate. —AD]

3.6.2 Interview Question Selection

[The aspects we focused on. —AD]

3.6.3 Interview Methods

[The ethics approvals, the way to ask questions, the way to transcript answers, and the technologies used. —AD]

Chapter 4

Measurement Results

For some qualities, it might be a good idea to cross-compare with the empirical scores.

4.1 Selected Software List

4.2 Installability

4.3 Correctness & Verifiability

4.4 Reliability

4.5 Robustness

4.6 Usability

4.7 Maintainability

4.8 Reusability

4.9 Understandability

4.10 Visibility & Transparency

Chapter 5

Interviews with Developers

5.1 Summary of Answers

- Start with one by one, with commonalities and interesting special cases.
- Shorten and summarize later.

5.2 Discussions

Any conclusions?

Chapter 6

Threat to Validity

Chapter 7

Recommendations

I think the recommendations can originate from both parts - measurements and interviews.

Chapter 8

Conclusions

No clues yet. Should be started at a later stage.

Bibliography

[1] Kari Björn. Evaluation of open source medical imaging software: A case study on health technology student learning experience. *Procedia Computer Science*, 121:724–731, 01 2017.

[2] Andreas Brühschwein, Julius Klever, Anne-Sophie Hoffmann, Denise Huber, Elisabeth Kaufmann, Sven Reese, and Andrea Meyer-Lindenberg. Free dicom-viewers for veterinary medicine: Survey and comparison of functionality and user-friendliness of medical imaging pacs-dicom-viewer freeware for specific use in veterinary medicine practices. *Journal of Digital Imaging*, 03 2019.

[3] James Edward Corbly. The free software alternative: Freeware, open source software, and libraries. *Information Technology and Libraries*, 33(3):65–75, Sep. 2014.

[4] Steve Emms. 16 best free linux medical imaging software. `https://www.linuxlinks.com/medicalimaging/`, 2019. [Online; accessed 02-February-2020].

[5] GNU. Categories of free and nonfree software. https://www.gnu.org/philosophy/categories.html, 2019. [Online; accessed 20-May-2021].

[6] Daniel Haak, Charles-E Page, and Thomas Deserno. A survey of dicom viewer software to integrate clinical research and medical imaging. *Journal of digital imaging*, 29, 10 2015.

[7] Mehedi Hasan. Top 25 best free medical imaging software for linux system. https://www.ubuntupit.com/top-25-best-free-medical-imaging-software-for-linux-system/, 2020. [Online; accessed 30-January-2020].

[8] Alessio Ishizaka and Markus Lusti. How to derive priorities in ahp: A comparative study. *Central European Journal of Operations Research*, 14:387–400, 12 2006.

[9] Hemant Kumar Mehta. *Mastering Python scientific computing: a complete guide for Python programmers to master scientific computing using Python APIs and tools*. Packt Publishing, 2015.

[10] Hamza Mu. 20 free & open source dicom viewers for windows. https://medevel.com/free-dicom-viewers-for-windows/, 2019. [Online; accessed 31-January-2020].

[11] The Linux Information Project. Freeware definition. http://www.linfo.org/freeware.html, 2006. [Online; accessed 20-May-2021].

[12] Thomas L. Saaty. How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, 48(1):9–26, 1990. Desicion making by the analytic hierarchy process: Theory and applications.

[13] Ravi Samala. Can anyone suggest free software for medical images segmentation and volume? https://www.researchgate.net/post/Can_anyone_suggest_free_software_for_medical_images_segmentation_and_volume, 03 2014. [Online; accessed 31-January-2020].

[14] Spencer Smith, Jacques Carette, Olu Owojaiye, Peter Michalski, and Ao Dong. Quality definitions of qualities. Manuscript in preparation, 2020.

[15] Spencer Smith, Jacques Carette, Olu Owojaiye, Peter Michalski, and Ao Dong. Methodology for assessing the state of the practice for domain x. Manuscript in preparation, 2021.

[16] Omkarprasad S. Vaidya and Sushil Kumar. Analytic hierarchy process: An overview of applications. *European Journal of Operational Research*, 169(1):1–29, 2006.

Appendix A

Full Grading Template

appendix here

Appendix B

Summary of Measurements

appendix here

Appendix C

Interview Answers

appendix here

Appendix D

Ethics Approval

appendix here