Methodology for Assessing the State of the Practice for Domain X

Spencer Smith
McMaster University, Canada
smiths@mcmaster.ca

Jacques Carette
McMaster University, Canada
carette@mcmaster.ca

Olu Owojaiye
McMaster University, Canada
owojaiyo@mcmaster.ca

Peter Michalski
McMaster University, Canada
michap@mcmaster.ca

Ao Dong
McMaster University, Canada
donga9@mcmaster.ca

Abstract

...

2012 ACM Subject Classification Author: Please fill in 1 or more \ccsdesc macro

Keywords and phrases Author: Please fill in \keywords macro

Contents

1 Introduction

Purpose and scope of the document. [Needs to be filled in. Should reference the overall research proposal, and the "state of the practice" exercise in particular. —SS]

2 Overview of Steps in Assessing Quality of the Domain Software

1. Identify domain. (Provide criteria on a candidate domain.)
2.

3 Identify Candidate Software

1. Must be open source.
2. Must have GitHub repository.

4 Domain Analysis

Commonality analysis. Follow as for mesh generator (likely with less detail).

Commonality analysis document Steps:

1. Introduction
2. Overview of Domain
3. Add Commonalities - Split into simulation, input, output, and nonfunctional requirements
4. Add Variabilites - Split into simulation, input, output, system constraints, and nonfunctional requirements
5. Add Parameters of Variation - Split into simulation, input, output, system constraints, and nonfunctional requirements
6. Add Terminology, Definitions, Acronyms

Commonality analysis for Lattice Boltzmann Solvers can be found here.

5 Empirical Measures

5.1 Raw Data

Measures that can be extracted from on-line repos.

[Still at brainstorm stage. —AD]

- number of contributors
- number of watches
- number of stars
- number of forks
- number of clones
- number of commits
- number of total/code/document files
- lines of total/logical/comment code
- lines/pages of documents (can pdf be extracted?)
- number of total/open/closed/merged pull requests
- number of total/open/closed issues
- number of total/open/closed issues with assignees

Instead of only focus on the current status of the above numbers, we may find the time history of them to be more valuable. For example, the number of contributors over time, the number of lines of code over time, the number of open issues over time, etc.

5.2 Processed Data

Metrics that can be calculated from the raw data.

[Still at brainstorm stage. —AD]

- percentage of total/open/closed issues with assignees - Visibility/Transparency
- lines of new code produced per person-month - Productivity
- lines/pages of new documents produced per person-month - Productivity

- number of issues closed per person-month - Productivity
- percentage of comment lines in the code - maintainability [Not Ao's qualities —AD]

In the above calculations, a month can be determined to be 30 days.

5.3 Tool Tests

[This section is currently a note of unorganized contents. Most parts will beremoved or relocated. —AD]

[This citation needs to be deleted later. It's here because my compiler doesn't work with 0 citations —AD] Emms [2019]

Most tests were done targeting to the repo of 3D Slicer GitHub repo

5.3.1 git-stats

GitHub repo

Test results: http://git-stats-slicer.ao9.io/ the results are output as webpages, so I hosted for you to check. Data can be downloaded as spreadsheets.

5.3.2 git-of-theseus

GitHub repo

Test results: It took about 100 minutes for one repo on a 8 core 16G ram Linux machine. It only outputs graphs.

5.3.3 hercules

GitHub repo

Test results: this one seems to be promising, but the installation is complicated with various errors.

5.3.4 git-repo-analysis

GitHub repo

5.3.5 HubListener

GitHub repo

The data that HubListener can extract.

Raw:

- Number of Files
- Number of Lines
- Number of Logical Lines
- Number of Comments

Cyclomatic: Intro

- Cyclomatic Complexity

Halstead: Intro

- Halstead Effort
- Halstead Bugs
- Halstead Length

- Halstead Difficulty
- Halstead Time
- Halstead Vocabulary
- Halstead Volume

Test results: HubListener works well on the repo of itself, but it did not work well on some other repos.

5.3.6 gitinspector

GitHub repo

Test results: it doesn't work well. Instead of creating output results, it prints the results directly in the console.

6 User Experiments

6.1 Usability Experiment

Steps:

1. Write research questions
2. Prepare logistic details(location, time table, recording setup, moderators etc)
3. Prepare survey questions
4. Select a list of projects
5. Design the tasks for the study subjects to perform (tasks can be defined based on user category, at least one task to modify the software)
6. Select participants
7. Conduct usability session
8. Survey the study subject to collect pre-experiment data
9. Perform tasks
10. Observe the study subjects (take notes, record sessions, watch out for body language and verbal cues)
11. Survey the study subjects to collect final feedback
12. Prepare experiment report
13. Perform pairwise comparison analysis
14. Prepare analysis report

6.2 Empirical Measures

1. Qualitative/Quantitative
2. Pairwise Comparison methodology needs to be described
3. Measure time to complete a task and compare actual with expected time of completion
4. From a list of given tasks, measure how many tasks were completed correctly

6.3 Tools

1. Downloaded OBS to record experiment sessions

6.4 To do

1. Detail plan

2. Describe Pairwise methodology

3. Define tasks

4. prepare survey questions

7 Analytic Hierarchy Process

Describe process. Domain expert review.

8 Quality Specific Measures

8.1 Installability [owner —OO]

8.2 Correctness [owner —OO]

8.3 Verifiability/Testability [owner —OO]

8.4 Validatability [owner —OO]

8.5 Reliability [owner —OO]

8.6 Robustness [owner —PM]

8.7 Performance [owner —PM]

8.8 Usability [owner —JC]

8.9 Maintainability [owner —PM]

8.10 Reusability [owner —PM]

8.11 Portability [owner —PM]

8.12 Understandability [owner —JC]

8.13 Interoperability [owner —AD]

8.14 Visibility/Transparency [owner —AD]

8.15 Reproducibility [owner —SS]

8.16 Productivity [owner —AD]

8.17 Sustainability [owner —SS]

8.18 Completeness [owner —AD]

8.19 Consistency [owner —AD]

8.20 Modifiability [owner —JC]

8.21 Traceability [owner —JC]

8.22 Unambiguity [owner —SS]

8.23 Verifiability [owner —SS]

8.24 Abstract [owner —SS]

9 Using Data to Rank Family Members

Describe AHP process (or similar).

A Appendix

A.1 Survey for the Selected Projects

[Several questions are borrowed from Jegatheesan2016, and needed to be cited later. —AD]

A.1.1 Information about the developers and users

1. Interviewees' current position/title? degrees?
2. Interviewees' contribution to/relationship with the software?
3. Length of time the interviewee has been involved with this software?
4. How large is the development group?
5. What is the typical background of a developer?
6. How large is the user group?
7. What is the typical background of a user?

A.1.2 Information about the software

1. [General —AD] What is the most important software quality(ies) to your work? (set of selected qualities plus "else")
2. [General —AD] Are there any examples where the documentation helped? If yes, how it helped. (yes*, no)
3. [General —AD] Is there any documentation you feel you should produce and do not? If yes, what is it and why? (yes*, no)
4. [Completeness —AD] Do you address any of your quality concerns using documentation? If yes, what are the qualities and the documents. (yes*, no)
5. [Visibility/Transparency —AD] Is there a certain type of development methodologies used during the development? ({Waterfall, Scrum, Kanban, else})
6. [Visibility/Transparency —AD] Is there a clearly defined development process? If yes, what is it. ({yes*, no})
7. [Visibility/Transparency —AD] Are there any project management tools used during the development? If yes, what are they. ({yes*, no})
8. [Visibility/Transparency —AD] Going forward, will your approach to documentation of requirements and design change? If not, why not. ({yes, no*})
9. [Correctness and Verifiability —AD] During the process of development, what tools or techniques are used to build confidence of correctness? (string)
10. [Correctness and Verifiability —AD] Do you use any tools to support testing? If yes, what are they. (e.g. unit testing tools, regression testing suites) ({yes*, no})
11. [Correctness and Verifiability —AD] Is there any document about the requirements specifications of the program? If yes, what is it. ({yes*, no})
12. [Portability —AD] Do you think that portability has been achieved? If yes, how? ({yes*, no})
13. [Maintainability —AD] How was maintainability considered in the design? (string)
14. [Maintainability —AD] What is the maintenance type? (set of {corrective, adaptive, perfective, unclear})
15. [Reusability —AD] How was reusability considered in the design? (string)
16. [Reusability —AD] Are any portions of the software used by another package? If yes, how they are used. (yes*, no)
17. [Reproducibility —AD] Is reproducibility important to you? (yes*, no)
18. [Reproducibility —AD] Do you use tools to help reproduce previous software results? If yes, what are they. (e.g. version control, configuration management) (yes*, no)
19. [Completeness —AD] Is any of the following documents used during the development? (yes*, no)

20. [General —AD] Will this experience influence how you develop software? Do you see yourself maintaining the same level of documentation, tool support as you go forward? (string)

- Module Guide
- Module Interface Specification
- Verification and Validation Plan
- Verification and Validation Report

References

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