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 **2**

2 Research Questions **3**

3 Overview of Steps in Assessing Quality of the Domain Software **3**

4 How to Identify the Domain **4**

5 How to Identify Candidate Software **4**

6 How to Initially Filter the Software List **4**

7 Domain Analysis **4**

8 Empirical Measures **5**
8.1 Raw Data . . . 5
8.2 Processed Data . . . 5
8.3 Tool Tests . . . 5
8.3.1 git-stats . . . 6
8.3.2 git-of-theseus . . . 6
8.3.3 hercules . . . 6
8.3.4 git-repo-analysis . . . 6
8.3.5 HubListener . . . 6
8.3.6 gitinspector . . . 6

9 Measure Using Shallow Measurement Template **7**

10 User Experiments **7**
10.1 Usability Experimental Procedure . . . 7
10.2 Procedure . . . 7
10.3 Task selection criteria . . . 7
10.4 Usability Questionnaire . . . 7

11 Analytic Hierarchy Process **7**

12 Rank Short List **7**

13 Quality Specific Measures **8**
13.1 Installability [owner —OO] . . . 8
13.2 Correctness [owner —OO] . . . 8
13.3 Verifiability/Testability [owner —OO] . . . 8
13.4 Validatability [owner —OO] . . . 8
13.5 Reliability [owner —OO] . . . 8
13.6 Robustness [owner —PM] . . . 8
13.7 Performance [owner —PM] . . . 8
13.8 Usability [owner —JC] . . . 8
13.9 Maintainability [owner —PM] . . . 8
13.10 Reusability [owner —PM] . . . 8
13.11 Portability [owner —PM] . . . 8
13.12 Understandability [owner —JC] . . . 8
13.13 Interoperability [owner —AD] . . . 8
13.14 Visibility/Transparency [owner —AD] . . . 8
13.15 Reproducibility [owner —SS] . . . 8
13.16 Productivity [owner —AD] . . . 8
13.17 Sustainability [owner —SS] . . . 8
13.18 Completeness [owner —AD] . . . 8
13.19 Consistency [owner —AD] . . . 8
13.20 Modifiability [owner —JC] . . . 8
13.21 Traceability [owner —JC] . . . 8
13.22 Unambiguity [owner —SS] . . . 8
13.23 Verifiability [owner —SS] . . . 8
13.24 Abstract [owner —SS] . . . 8

14 Using Data to Rank Family Members **8**

A Appendix **8**
A.1 Survey for the Selected Projects . . . 8
A.1.1 Information about the developers and users . . . 9
A.1.2 Information about the software . . . 9

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. Reference questions we are trying to answer. —SS]

2 Research Questions

In general questions:

1. Comparison between domains
2. How to measure qualities
3. How does the quality compare for projects with the most resources to those with the fewest?
4. What skills/knowledge are needed by future developers?
5. How can the development process be improved?
6. What are the common pain points?

For each domain questions"

1. Best examples within the domain
2. What software artifacts?
3. What are the pain points?
4. Any advice on what can be done about the pain points?

Measure the effort invested and the reward. Related to sustainability.

Collect the data and see what conclusions follow. For an individual domain, between domains. The process isn't so much about ranking the software as it is about looking at the software closely and see what conclusions arise. The measurements are intended to force scrutiny, from different perspectives.

3 Overview of Steps in Assessing Quality of the Domain Software

1. Start with state of practice research questions. (Section 2) [To be completed —PM]
2. Identify the domain. (Section 4) [To be completed —PM]
3. *Domain Experts*: Create a top ten list of software packages in the domain. (Meeting Agenda with Domain Experts)
4. Brief the Domain Experts on the overall objective, research proposal, research questions, measurement template, survey for short list projects, usability tests, performance benchmarks, maintainability experiments. (Meeting Agenda with Domain Experts)
5. Identify broad list of candidate software packages in the domain. (Section 5) [To be completed —PM]
6. Preliminary filter of software packages list. (Section 6) [To be completed —PM]
7. *Domain Experts*: Review domain software list. (Meeting Agenda with Domain Experts)
8. Domain Analysis. (Section 7)[To be completed —PM]
9. *Domain Experts*: Vet domain analysis. (Meeting Agenda with Domain Experts) [This was part of the original Steps in Assessing Quality list. We need to add it to the Meeting Agenda —PM]
10. Gather source code and documentation for each prospective software package.
11. Collect empirical measures. (Section 8) [To be completed —PM]
12. Measure using "shallow" measurement template. (Section 9) [To be completed —PM]
13. Use AHP process to rank the software packages. (Section 11) [To be completed —PM]
14. Identify a short list of top software packages, typically four to six, for deeper exploration according to the AHP rankings of the shallow measurements.

15. *Domain Experts*: Vet AHP ranking and short list. (Meeting Agenda with Domain Experts) [This was part of the original Steps in Assessing Quality list. We need to add it to the Meeting Agenda —PM]
16. With short list:
    a. Survey developers [To do: Link to survey —PM]
    b. Usability experiments [Move the below usability section to a new doc, and link to it. —PM]
    c. Performance benchmarks[note: this is still in consideration —PM]
    d. Maintainability experiments[note: this is still in consideration —PM]
17. Rank short list. (Section 12) [To be completed —PM]
18. Document answers for research questions.

[The domain expert is involved in multiple steps in the process. How best to get their feedback? The domain experts are busy and are unlikely to devote significant time to the project. We need to quickly get to the point. Maybe something around task based inspection? Directed interview? —SS]

4 How to Identify the Domain

1. Needs to be "a community of"/many people using and studying this domain
2. There needs to be "many" software solutions
3. There needs to be a user community using these solutions
4. Must be in the research software scope
5. Specific enough scope
6. Must be a well understood domain / mature

5 How to Identify Candidate Software

[outline here how to find such a list —PM]

1. Must have viewable source code.
2. Ideally have a git repository. [Not all of mine had this.. —PM]

6 How to Initially Filter the Software List

The person doing this must keep track of both the initial and filtered lists. The reasons for this is traceability bc..

1. Professionalism..
2. Available documentation..
3. Status..
4. Availability of source code..

7 Domain Analysis

Commonality analysis. Follow as for mesh generator (likely with less detail). Final result will be tables of commonalities, variabilities and parameters of variation.

Commonality analysis document Steps:

1. Introduction
2. Overview of Domain
3. Add Commonalities - Split into simulation, input, output, and nonfunctional requirements

4. Add Variabilities - 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.

8 Empirical Measures

8.1 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.

8.2 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.

8.3 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

8.3.1 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.

8.3.2 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.

8.3.3 hercules

GitHub repo

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

8.3.4 git-repo-analysis

GitHub repo

8.3.5 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.

8.3.6 gitinspector

GitHub repo

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

9 Measure Using Shallow Measurement Template

This step... The measurement template can be found here.

1. Step 1..

10 User Experiments

10.1 Usability Experimental Procedure

10.2 Procedure

1. Survey participants to collect pre-experiment data
2. Participants perform tasks
3. Observe the study subjects (take notes, record sessions(OBS screen recorder), watch out for body languages and verbal cues)
4. Survey the study subjects to collect feedback (post experiment interview)
5. Prepare experiment report
6. Perform pairwise comparison analysis
7. Prepare analysis report

10.3 Task selection criteria

**The task selection will be determined with the aid of the domain experts attached to any of the selected projects.
**The domain experts will be asked to consider the below criteria when defining a task.
**Domain experts will also be asked to identify what background knowledge is necessary for the suggested tasks - Novice, Intermediate, Advanced

1. Collectively all tasks should not take no more than 2 hours.
2. Selected tasks should reflect common use cases of the software.
3. Include tasks that require a set of sequential or hierarchical steps to be completed

10.4 Usability Questionnaire

Two sources of standardized usability questionnaire we could use.
- https://www.usabilitest.com/sus-pdf-generator- 20-29 - SUS.
- https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/ - PSSUQ

11 Analytic Hierarchy Process

Describe process. Domain experts review. Outline the tool we used. The README file of the tool can be found here.

12 Rank Short List

Rank using pairwise comparison of short list results.

1. Compare with respect to usability
2. ...

13 Quality Specific Measures

13.1 Installability [owner —OO]

13.2 Correctness [owner —OO]

13.3 Verifiability/Testability [owner —OO]

13.4 Validatability [owner —OO]

13.5 Reliability [owner —OO]

13.6 Robustness [owner —PM]

13.7 Performance [owner —PM]

13.8 Usability [owner —JC]

13.9 Maintainability [owner —PM]

13.10 Reusability [owner —PM]

13.11 Portability [owner —PM]

13.12 Understandability [owner —JC]

13.13 Interoperability [owner —AD]

13.14 Visibility/Transparency [owner —AD]

13.15 Reproducibility [owner —SS]

13.16 Productivity [owner —AD]

13.17 Sustainability [owner —SS]

13.18 Completeness [owner —AD]

13.19 Consistency [owner —AD]

13.20 Modifiability [owner —JC]

13.21 Traceability [owner —JC]

13.22 Unambiguity [owner —SS]

13.23 Verifiability [owner —SS]

13.24 Abstract [owner —SS]

14 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].