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. 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.
2. Identify domain. (Provide criteria on a candidate domain.)
3. Identify list of candidate software in the domain. (Section 4)
4. *Domain Expert*: Vet domain software list.
5. Domain Analysis. (Section 5)
6. *Domain Expert*: Vet domain analysis
7. Measure using "shallow" measurement template.
8. Empirical measures.
9. Use AHP process to rank the software. Tool for estimating AHP results? Tool for ranking and sorting by various criteria? Conditional formula to assess software within each measure, or just use opinion of measurer?
10. Identify short list for deeper exploration.
11. *Domain Expert*: vet AHP ranking and short list.
12. With short list:
    a. Survey developers
    b. Usability experiments
    c. Performance benchmarks
    d. Maintainability experiments
13. Compare shallow and deep measures and ranking
14. 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 Identify Candidate Software

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

5 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.

6 Empirical Measures

6.1 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.

6.2 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.

6.3 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

6.3.1 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.

6.3.2 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.

6.3.3 hercules

GitHub repo

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

6.3.4 git-repo-analysis

GitHub repo

6.3.5 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.

6.3.6 gitinspector

GitHub repo

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

7 User Experiments

7.1 Usability Experimental Procedure

7.2 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

7.3 Task selection criteria

Below are the criteria to be used in defining usability experiment tasks. The task selection will be determined with the aid of the domain expert attached to any of the selected projects. The domain expert will be asked to consider the below criteria when helping us to determine a task.

1. All tasks should take no more than 2 hours (in one sitting) for completion.
2. Common and popular tasks that users would perform (these apply to all selected projects collectively) with the software application.
3. Tasks related to what users will do in the real world
4. End to end tasks that require users to go through sequential or hierarchical steps.

**Domain experts will be asked to identify what background knowledge is necessary for the suggested tasks (Expert, Intermediate, Novice).

7.4 Usability Questionnaire

This questions need to be reworded to make them as open ended as possible
source https://www.usabilitest.com/sus-pdf-generator- 20-29

1. Overall, I am satisfied with how easy it is to use this system.
2. It was simple to use this system.

3. I could effectively complete the tasks and scenarios using this system.
4. I was able to complete the tasks and scenarios quickly using this system.
5. I was able to efficiently complete the tasks and scenarios using this system.
6. I felt comfortable using this system.
7. It was easy to learn to use this system.
8. I believe I could become productive quickly using this system.
9. The system gave error messages that clearly told me how to fix problems.
10. Whenever I made a mistake using the system, I could recover easily and quickly.
11. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.
12. It was easy to find the information I needed.
13. The information provided for the system was easy to understand.
14. The information was effective in helping me complete the tasks and scenarios.
15. The organization of information on the system screens was clear.
16. The interface of this system was pleasant.
17. I liked using the interface of this system.
18. This system has all the functions and capabilities I expect it to have.
19. Overall, I am satisfied with this system.
20. I think that I would like to use this system frequently.
21. I found the system unnecessarily complex.
22. I thought the system was easy to use.
23. I think that I would need the support of a technical person to be able to use this system.
24. I found the various functions in this system were well integrated.
25. I thought there was too much inconsistency in this system.
26. I would imagine that most people would learn to use this system very quickly.
27. I found the system very cumbersome to use.
28. I felt very confident using the system.
29. I needed to learn a lot of things before I could get going with this system.
30. **The below were added to this list by Olu based on Nielson's Heuristics**
31. The software provided me system's current state and what actions can be done or where to go or what change was made.
32. I was able reuse a previously defined value without having to memorize or jot it.
33. I was able to reverse an error and didn't have to restart a task due to error.
34. I was able to refer to help/documentation easily when I needed it.
35. I was pre-warned of errors based on the actions I performed

8 Analytic Hierarchy Process

Describe process. Domain expert review.

9 Quality Specific Measures

9.1 Installability [owner —OO]

9.2 Correctness [owner —OO]

9.3 Verifiability/Testability [owner —OO]

9.4 Validatability [owner —OO]

9.5 Reliability [owner —OO]

9.6 Robustness [owner —PM]

9.7 Performance [owner —PM]

9.8 Usability [owner —JC]

9.9 Maintainability [owner —PM]

9.10 Reusability [owner —PM]

9.11 Portability [owner —PM]

9.12 Understandability [owner —JC]

9.13 Interoperability [owner —AD]

9.14 Visibility/Transparency [owner —AD]

9.15 Reproducibility [owner —SS]

9.16 Productivity [owner —AD]

9.17 Sustainability [owner —SS]

9.18 Completeness [owner —AD]

9.19 Consistency [owner —AD]

9.20 Modifiability [owner —JC]

9.21 Traceability [owner —JC]

9.22 Unambiguity [owner —SS]

9.23 Verifiability [owner —SS]

9.24 Abstract [owner —SS]

10 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].