Template for Grading Scientific Computing Software

Spencer Smith, Adam Lazzarato, Yue Sun, Zheng Zeng,
Vasudha Kapil and Jacques Carette

May 11, 2020

The table below lists the full set of measures that are assessed for each software product. The measures are grouped under headings for each quality, an done for summary information. Following each measure, the type for a valid result is given in brackets. Many of the types are given as enumerated sets. For instance, the response on many of the questions is one of "yes," "no," or "unclear." The type "number" means natural number, a positive integer. The types for date and url are not explicitly defined, but they are what one would expect from their names. In some cases the response for a given question is not necessarily limited to one answer, such as the question on what platforms are supported by the software product. Case like this are indicated by "set of" preceding the type of an individual answer. The type in these cases are then the power set of the individual response type. In some cases a superscript *is used to indicate that a response of this type should be accompanied by explanatory text. For instance, if problems were caused by uninstall, the reviewer should note what problems were caused.

Table 1: Grading Template

Summary Information
Software name? (string)
URL? (url)
Educational institution (string)
Software purpose (string)
Number of developers (number - use repo commit logs)
How is the project funded (string)
Number of downloads for current version (number)
Release date (date)
Last updated (date)
Status ({alive, dead, unclear})
License ({GNU GPL, BSD, MIT, terms of use, trial, none, unclear})
Platforms (set of {Windows, Linux, OS X, Android, Other OS})
Category ({concept, public, private})
Development model ({open source, freeware, commercial})
Publications using the software (set of url)
Publications about the software (set of url)
Is source code available? ({yes, no})
Source code URL ({set of url, n/a, unclear})
Programming language(s) (set of {FORTRAN, Matlab, C, C++, Java, R, Ruby, Python, Cython, BASIC, Pascal, IDL, unclear})
Installability (Measured via installation on a virtual machine.)
Are there installation instructions? ({yes, no})
Are the installation instructions in one document? ({yes, no, n/a})
Are the installation instructions linear? ({yes, no, n/a})
Do the installation instructions begin with a fresh operating system? ({yes, no, unclear})

Are compatible operating system versions listed? ({yes, no})
Is there something in place to automate the installation (makefile, script, installer, etc)? ({yes*, no})
Is there a specified way to validate the installation, such as a test suite? ({yes*, no})
How many steps were involved in the installation? What OS was used? (number, set of {Windows, Linux, OS X, Android, Other OS})
How many software packages need to be installed before or during installation?(number)
Are required package versions listed? ({yes, no, n/a})
Are there instructions for the installation of required packages? ({yes, no, n/a})
Run uninstall, if available. Were any obvious problems caused? ({unavail, yes*, no})
Overall impression? ({1 .. 10})

Correctness and Verifiability

Are external libraries used? ({yes*, no, unclear})
Are versions of external libraries specified? ({yes, no, n/a})
Does the community have confidence in this library? ({yes, no, unclear})
Any reference to the requirements specifications of the program? ({yes*, no, unclear})
What tools or techniques are used to build confidence of correctness? (string)
If there is a getting started tutorial? ({yes, no*})
Are the tutorial instructions linear? ({yes, no, n/a})
Does the getting started tutorial provide an expected output? ({yes, no, n/a})
Does running the tutorial create output that matches the expected output? ({yes, no*, n/a})
Overall impression? ({1 .. 10})

Surface Reliability

Did the software "break" during installation? ({yes*, no})
If the software installation broke, was a descriptive error message displayed? ({yes, no, n/a})
If the software installation broke, was the installation recoverable? ({yes, no, n/a})
Did the software "break" during the initial tutorial testing? ({yes*, no, n/a})
If the tutorial testing broke, was a descriptive error message displayed? ({yes, no, n/a})
If the tutorial testing broke, was the tutorial testing recoverable? ({yes, no, n/a})
Overall impression? ({1 .. 10})

Surface Robustness

Does the software handle unexpected/unanticipated input (like data of the wrong type, empty input, missing files or links) reasonably? (a reasonable response can include an appropriate error message.) ({yes, no*})
For any plain text input files, if all new lines are replaced with new lines and carriage returns, will the software handle this gracefully? ({yes, no*, n/a})
Overall impression? ({1 .. 10})

Surface Performance

Is there evidence that performance was considered? ({yes*, no})
Does the software allow the use of a GPU for processing? ({yes, no, unclear})
Overall impression? ({1 .. 10})

Surface Usability

Is there a getting started tutorial? ({yes, no})
Is there a standard example that is explained? ({yes, no})
Is there a user manual? ({yes, no})
Does the application have the usual "look and feel" for the platform it is on? ({yes, no*})
Are there any features that show a lack of visibility? ({yes, no*})
Are expected user characteristics documented? ({yes, no})
What is the user support model? (string)
Overall impression? ({1 .. 10})

Maintainability

Is there a history of multiple versions of the software? ({yes, no, unclear})
Is there any information on how code is reviewed, or how to contribute? ({yes*, no})
Is there a changelog? ({yes, no})
Are technical documents available (requirements, architecture, code, algorithms, interfaces, APIs)? ({yes, no, unclear})
What is the maintenance type? (set of {corrective, adaptive, perfective, unclear})
What issue tracking tool is employed? (set of {Trac, JIRA, Redmine, e-mail, discussion board, sourceforge, google code, git, BitBucket, none, unclear})
How many known bugs have been fixed? ({number})
How many open known bugs remain? ({number})
Are the majority of identified bugs fixed? ({yes, no*, unclear})
What is the percentage of identified issues that are closed? (percentage)
Are files commented well? ({yes, no, unclear})
Which version control system is in use? ({svn, cvs, git, github, unclear})
Is there evidence that maintainability was considered in the design? ({yes*, no})
Are there code clones? ({yes*, no, unclear})
Overall impression? ({1 .. 10})

Reusability

Are any portions of the software used by another package? ({yes*, no})
Is there evidence that reusability was considered in the design? (API documented, web service, command line tools, ...) ({yes*, no, unclear})
Is the system modularized? ({yes, no, unclear})
Are module interfaces clearly defined? ({yes, no, n/a})
Are modules described in the user guide? ({yes, no, n/a})
Overall impression? ({1 .. 10})

Portability

What platforms is the software advertised to work on? (set of {Windows, Linux, OS X, Android, Other OS})
Are special steps taken in the source code to handle portability? ({yes*, no, n/a})
Is portability explicitly identified as NOT being important? ({yes, no})
Convincing evidence that portability has been achieved? ({yes*, no})
Overall impression? ({1 .. 10})

Surface Understandability (Based on 10 random source files)

Consistent indentation and formatting style? ({yes, no, n/a})
Explicit identification of a coding standard? ({yes*, no, n/a})
Are the code identifiers consistent, distinctive, and meaningful? ({yes, no*, n/a})
Are constants (other than 0 and 1) hard coded into the program? ({yes, no*, n/a})
Comments are clear, indicate what is being done, not how? ({yes, no*, n/a})
Is the name/URL of any algorithms used mentioned? ({yes, no*, n/a})
Parameters are in the same order for all functions? ({yes, no*, n/a})
Is code modularized? ({yes, no*, n/a})
Descriptive names of source code files? ({yes, no*, n/a})
Is a design document (data structures, architecture, interface, procedures) provided? ({yes*, no, n/a})
Overall impression? ({1 .. 10})

Interoperability

Does the software interoperate with external systems? ({yes*, no})
Is there a workflow that uses other softwares? ({yes*, no})
If there are external interactions, is the API clearly defined? ({yes*, no, n/a})
Could the software output be used by other software as input? ({yes*, no, unclear})
Could the software take output from other software as input? ({yes*, no, unclear})
Does the software generate files with uncommon formats? (require uncommon software to open) ({yes*, no, unclear})
Could the software work with customized plug-ins? ({yes*, no, unclear})
Overall impression? ({1 .. 10})

Visibility/Transparency

Is the development process defined? If yes, what process is used. ({yes*, no, n/a})
Ease of external examination relative to other products considered? ({1 .. 10})
Are there any documents recording the development process and status? ({yes*, no})
Is the development environment documented? ({yes*, no, unclear})
Are there release notes? ({yes*, no})
In the issue tracking system, what percentage of issues are assigned to specific developers? ({number, n/a})
In the issue tracking system, are some issues tagged with types and/or priorities? ({yes, no, n/a})

Were there any project management tools used during development? ({yes, no})
Overall impression? ({1 .. 10})

Reproducibility
Is there a record of the environment used for their development and testing? ({yes*, no})
Is test data available for verification? ({yes, no})
Are automated tools used to capture experimental context? ({yes*, no})
Overall impression? ({1 .. 10})