Sparse Matrix Software Package Design Proposal Critique

Proposal Title: SPAM

Author: Kyle Lee

Critic: Terence Henriod

CS 365: Mathematics of Computer Science

11/14/2013

|  |  |  |
| --- | --- | --- |
| Category | Points  Received | Points  Possible |
| Write-Up: | 25 | 30 |
| Explanations & Descriptions: | 28 | 30 |
| Design and Plans: | 37 | 40 |
| Total | 90 | 100 |

1. **Write-Up**

In general, the largest concern for this proposal is that the writing needs better organization to help the reader identify what is important to know about the project. Keeping discussion of topics within sections of the proposal where they belong, utilizing appropriate paragraph structure, and keeping sentences in an order such that they lead to one another would all benefit the overall flow of the design proposal. Using low level language in order to make statements clear and simple was a good choice.

A prime example of the need for appropriate section partitioning would be the Error Handling and User Interface sections. The two should be (intuitively) separate, for the most part, but there is a large amount of crossover between the two topics.

There are many instances where paragraph structure is too large, combines too many ideas, or dilutes the idea currently being discussed (by mixing the subject at hand with other topics). Also, there was a lot of information that is just thrown in. It is unclear, sometimes, if information should be of concern to the reader (e.g. when FORTRAN is mentioned in the Abstract section. Why is FORTRAN relevant to the discussion if it has been decided that C++ will be used?).

The language was rough in some places, and smoother language and sentence structure can be utilized to improve the flow of the paper. For example, some paragraphs could be separated to improve the flow of the document, as well as help distinguish different topics from one another. Example: in the “Error Handling” and “User Interface” sections, each is only a single paragraph, yet multiple operations are described. Split the operations into separate paragraphs when necessary.

As far as punctuation and grammar though, there were only a few minor faux pas. Nothing that would constitute an actual error, but some minor changes would enhance readability and presentation. I did not detect any spelling errors.

1. **Explanations and Descriptions**

The descriptions of the coding implementation of SPAM was good, but take care in order to appropriately match the scope of the assignment. Specific coding techniques should likely not be discussed unless they are effective or unusual in some novel or significant way. Including a list of public interface functions does match the scope of the document and is a good design idea. Listing public interface functions helps to make it clear to the reader what actions SPAM is capable of.

The explanations, while specific in some parts, did leave something to be desired in other parts of the document. In some parts of the proposal, you were pedantic enough define a matrix or state coding techniques that might be used (e.g. that C++ iostream operations would be used), but in other parts possible important topics were neglected, like what format matrix data should be stored in when stored in a file.

Use of illustrations, tables, and equations was a good idea, but improving their implementation and layout is suggested to improve their appearance and better utilize the space of the page.

With the exception of description of the Yale matrix storage format, most descriptions and explanations are in good standing; they are in position to be easily refined into something ideal. The simplicity of the explanations is definitely a gift to the reader.

As a suggestion, when explaining the reasoning for a given choice, it may be beneficial to include any counter-arguments for a particular design choice and then address why the counter-argument either does not apply or is not significantly detrimental to the reasoning behind your selection. For example, when discussing the format that the data is stored in (the Yale format was chosen), discuss the advantages that other storage formats might have and then state why those advantages either did not matter for the goals of SPAM or did not provide a significant enough benefit over the Yale format to outweigh the advantages that the Yale format provides.

Certainly SPAM should be clearly explained/defined. It seems that you expected that the reader would just intuitively understand that SPAM utilized segments of SPArse Matrix (or something), which could result in confusion for some readers.

1. **Proposed Design and Plans**

The proposed design seems to be a solid start, but there are certainly conditions that still need to be addressed, conditions like what happens if a matrix is too large (even computers have their limits), or what the user should see as they are using the software. You discussed the package from a coding perspective more than a design one, but a very important aspect of the design of a software package is how the user will interact with it.

It was good to consider the Yale storage format for memory management perspectives, but will it translate well to performing computations? Or might another format be required for computations? Questions of this nature should be addressed in the design of the package.

The specifications of SPAM could be better defined, as well. For example, you say that SPAM should reject a matrix that is not sparse, but you do not define what is not sparse enough to be appropriately handled by SPAM.

In general, this design is a good start, but effort should be applied in order to tie up any loose ends or gaps in the design of the package. There are many conditions and situations that should be addressed. The pre- and post- conditions of the public interface functions should be clearly defined for the benefit of the reader/user.

As a suggestion, if you are going to take the time to mention possible improvements for the SPAM package, it would be of interest to the reader and those evaluating the proposal to see first, that more than one improvement is being considered (otherwise, why not include the feature in the first release?), as well as why the coming improvements are important enough to mention. The reader should be informed about why additions might be necessary or desired.

1. **Overall**

Overall this proposal is an acceptable rough draft with good ideas, and efforts to improve clarity, readability and presentation could make this software package design something viable. Take all editing/critique remarks into consideration in order to accomplish this.

You may want to affirm the scope/scale of the design. You may want to refer to the instructor, but I interpreted the assignment as a design that would discuss overall features and looks, and neglect the finer details such as programming language to be used, specific commands, etc. Keep that in mind as you read my remarks. You may feel differently, and the instructor may consider your take on the assignment perfectly acceptable.