

Characteristics of a "Successful" Application.

Caroline Bahler, Meridian Software, Inc.®

Abstract

An application can be judged "successful" by two different sets of criteria. The first set of criteria belongs to the users - does the application perform as required. The second set of criteria belongs to Information Systems (IS) - was the application developed on time, within budget, and is it easy to maintain and support. A truly "successful" application needs to meet both sets of criteria. Applications that meet this goal have these characteristics in common: agreed upon requirements, good documentation, parameter driven code, reusable and self-generating code, programming efficiencies when possible, and testing. The object of this paper is to discuss these characteristics and how they contribute to a "successful" application.

Let Us Start at the Very Beginning...

Once upon a time a Marketing Manager (MM) for the Enormous Toy Company wandered into the IS department. This MM went to her friend Isp's (an information system programmer) office and said 'Isp, I need a way to track the sales of our biggest product the Gigantic IOU Doll. Could Isp create a tracking application?' Well, Isp was delighted because she had been trying to get MM to consolidate all of those ad hoc requests. That way Isp could make one manageable project instead of being bugged by MM whenever she needed information. So, Isp said, "let's talk to my manager " and off they went.

Isp's manager was I.S. Head, the Department Head of IS. He had been worried because Isp was spending so much of her time and other developer's time filling MM's requests. In fact, I.S. Head had been thinking about proposing to MM that it was time to move all of her ad hoc requests into a formal application with Marketing footing the bill instead of IS. So, when Isp and MM showed up in his office, he was happy to talk about developing a project for MM. And by the way did she have MONEY in HER budget for this? MM assured I.S. Head that the budget was there. When could they get started? So, everyone was happy and an application was born.

Characteristics

Characteristic 1 – Defined Requirements

Probably the hardest part of any project whether it is a "simple" program to create a report or a major

application distributed via an Intranet is the definition in "English" of what the application will do. The users may have a very clear idea of what they want individual reports to contain, but a fuzzy idea of how they want an application to act. The success or failure of an application or even a report depends upon being able to define the users' requirements and for the users to agree that the requirements defined will meet their needs.

The ART of requirement definition has been discussed and written about by many authors. However, the gist of what all authors say is that the person who ultimately will use the program, report, or application needs to invest time in helping to write down specifically what the report or application needs to accomplish. It does not matter if a very traditional approach or rapid application development techniques are used, the end user needs to not only have a stake in making the application a success, they need to participate in the requirements phase of the development process.

Once the "English" requirements are defined then technical details can be incorporated into an overall design and technical specifications. In turn these specifications need to be documented.

In our little story, Isp was able to convince MM of the importance of well-defined requirements and specifications. In fact, as MM went through the requirements process, she discovered that she had a better understanding of the type of information that would be needed to make informed marketing decisions. In addition, MM had a document that detailed what could be expected from the tracking application and what could not. In doing this, Isp and other members of IS ensured that everyone's expectations for the application were the same.

Characteristic 2 – Well Documented

Documentation can be the most time consuming and least rewarding part of any project. However, it can be one of the most important elements, starting as discussed above, in the creation of requirements and technical specifications. Documentation can come in many forms, but probably one of the most useful from a developer's perspective is in-line documentation or comments, otherwise known as the programmer's bane. Just remember the next time you silently cuss out programmer X for not documenting HER code thoroughly that programmer X may be thinking the same thoughts about you.

In general, there you should place a comment block at the beginning of each program that contains the following information:

- name of the program
- when it was created
- what it does and how it is used
- date and nature of any changes made to the program since it was written
- programmer's name and organization

Next, you should list any macro variables used within the program or passed to the program. This information can be contained within the comment block or as a separate block. Where macro variable information resides depends upon the standard practices of an individual, consulting group, or IS shop.

Comments within the body of the program, i.e. in-line documentation, can contain the following types of information:

- information about the program's flow
- changes to the program
- additional information that may be needed by someone who comes along behind you or even for you to maintain or enhance the program in years to come

Overall, in-line documentation benefits everyone in IS by making code easier to maintain and enhance. At least that is the answer I.S. Head gives every time a developer complains about how much time documentation takes.

Characteristic 3 - Parameter Driven Code

Why should you bother, you may ask, when time is of the essence during the development of this application, as in all applications.. Writing parameter driven code can increase the time it takes to develop an application why not hard code, especially if code already exists to handle a part of an application? The main answer is that hard coded programs break the minute something changes. By utilizing lookup tables or formats to house program parameters a single easily maintainable source for definitions will exist that all programs can utilize.

For example, during this fiscal year, MM has defined the northeast region as Maine, New Hampshire, Vermont, Connecticut, Rhode Island, and New York. She then indicates that next year this definition will most likely change. Region is one of the main grouping variables for sales tracking. In the past, Isp and other IS developers just asked MM for this year's definition and hard coded the region definition

in the programs that produced ad hoc reports. Isp realized that this practice could not continue within the application. Too many parts of the application were dependent on region. What was needed was a single source containing the region definitions that IS **DID NOT** have to maintain. Isp handled this by creating a region table that contained region id, name, and definition (state) that MM or one of her staff "owned" and maintained.

Characteristic 4 – Programming Efficiencies

Creating and utilizing efficient programming techniques can be classified as the "Holy Grail" of programming. They have been around for as long as computers. What is efficient programming? Efficient programming can be defined as using as little source code as possible to get the job done. The definition that will be used within this paper is not that simplistic. Efficient programming requires evaluating the following:

- I/O processing capabilities of the hardware
- amount of data storage/work space
- amount of memory.

Programming techniques can affect:

- CPU time
- Programming time

Optimizing I/O

I/O usage is affected by the number of times the SAS® System accesses the data storage device (disk, cd, or tape). Some methods for increasing I/O efficiency are:

1. Modify or plan your program to limit the number of times that the data is processed.
 - Combine multiple DATA steps into a single DATA step wherever possible.
 - Use the SQL procedure to consolidate code
2. Reduce the number of times you read the data by increasing the amount of data processed each time the data is accessed.
3. Reduce the amount of data processed by:
 - setting options that can affect this process (BUFNO, BUFFSIZE, CATCACHE, and COMPRESS)
 - reading only the data that is needed
 - using PROC DATASETS to copy data sets with indexes
 - performing data subsets early and at the same time

- keeping only the variables needed within each temporary data set (DROP or KEEP statements)
- using FIRSTOBS and OBS options to reduce the number of observations processed.

4. If possible, have the SAS WORK directory located on a different drive

Optimizing Data Storage/Work Space

Workspace can become a hurdle when you work with extremely large data sets or a limited amount of workspace. The main constraint is the amount of disk space required to sort a data set. SAS requires a workspace with a minimum of 2.1 times the size of a data set to sort a data set. Therefore, when working with very large data sets or data sets that will grow in size over time, design your program to take into account the sorting requirements.

- Retain only desired variables
- Use the LENGTH statement to reduce the variable size if possible
- Use data compression strategies
- Use DATA _NULL_ steps for processing
- Use the TAGSORT option within PROC SORT to reduce the amount of space required to sort a data set

Optimizing Memory Usage

The following techniques will reduce memory usage. However, there is a tradeoff in increased I/O processing or CPU usage that you will need to evaluate.

- Increase the MEMSIZE system option and increase the number and size of buffers used (reduces the impact on I/O)
- Read only the data that is needed
- Use WHERE statement when possible
- Use PROC DATASETS to copy data sets with indexes

Optimizing CPU Performance

CPU performance, memory usage, and I/O are all interrelated. Often decreasing one can positively or negatively affect another. The following techniques will increase CPU performance (reducing the amount of processing time):

- Utilize the Stored Program Facility – this facility is especially useful in situations that call for computation-intensive DATA steps or DATA steps that are called repetitively. This technique

is very effective for large Data Step jobs that are not I/O intensive.

- Reduce the search time for SAS executable files – the default CONFIG.SAS file specifies a default order for the directories containing SAS executable files. The directory order can be re-arranged within the PATH option so that the most commonly accessed directories are listed first.
- Specify Variable Lengths – the SAS System moves data in and out of the data vector in one large operation instead of single variables. Movement of data can occur very quickly (as low as 2 clock cycles) when data is properly aligned (in 8-byte boundaries). Unaligned data can severely increase the amount of time needed to read the data. To keep data aligned:
 1. Leave data at full width (8-bytes)
 2. Keep character data in multiples of 8 bytes in length (this wastes memory but keeps data aligned).
- Retain only the variables that are needed
- Create and use indexes with large data sets
- Utilize macros for redundant code
- Use If-Then-Else statements to process data
- Use PROC DATASETS to copy data sets with indexes
- Turn off the Macro Facility when it's not needed
- Avoid unnecessary sorting
- Use formats created from data sets as look-up tables to add descriptor information to another data set instead of joining look-up tables to base data sets
- Use a CLASS statement when available with procedures
- Use SQL joins with caution since they include an implicit sort

Optimizing Program Time

The following techniques can be used to decrease the amount of developing the source code for a specific task:

- Use the SQL procedure for code simplification
- Use procedures whenever possible
- Document programs and routines with comments
- Utilize macros for redundant code
- Code for unknown data values

- Assign descriptive and meaningful variable names
- Store formats and labels with the SAS data sets that use them
- Test program code using complete test data.

ISP and her fellow developers were fortunate, IS Head realized the importance of programming efficiency. Therefore, when the project was planned, time was allocated for benchmarking and other efficiency activities.

Characteristic 5 – Testing

To err is human but to really screw things up you need a computer. Testing is the one of crucial steps needed to deliver reports or applications that meet users' needs. A key aspect of testing is soliciting and actually getting user feedback.

During application/report development, the testing activity can be broken down into four (4) separate activities:

1. Unit testing – the testing that the developer does during program creation and validation.
2. System testing – a more formalized test that compares all of the requirements to the finished results. The tester is usually part of IS and is NOT the developer. The object of this testing is can the tester “break” the report or part of an application.
3. Integration testing – this can be combined into system testing and consists of testing how all of the components of an application interact with one another. The tester is usually part of IS and may be part of the development team.
4. User acceptance testing – testing done by the personnel who will be using the program or application. It can consist of the user reviewing a report that will be produced on a specific schedule or multiple users testing different parts of a complex application. The object of this testing is for the user(s) to interact with the application before it goes into production. This type of testing has also been called alpha and beta testing and user feedback is solicited.

The amount of testing required is dependent on the complexity of the report or application. All types of testing are not required in every situation. It is up to the developer and the end user to define the amount and nature of testing required for a given project.

In Conclusion

MM understood the need for user acceptance testing and assigned several people from her staff to the task. The feedback was excellent and the

project was finished to everyone's surprise on-time and within budget. IS Head was happy because he now had personnel to assign to other projects.

References

General

1. Gill, Paul. 1997. The Next Step: Integrating the Software Life Cycle with SAS Programming. SAS Institute, Inc. Cary, NC.
 - excellent discussion of Software Cycle and all of the characteristic listed within the paper
2. Whitney, C. Michael. 1999. Taming the Chaos: Managing Large SAS/AF Applications Using Programming Standards and the Source Control Manager of Version 7 of the SAS System. Proceeding of the Twenty-fourth Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 60-65.
 - good discussion of software development life cycle
 - examples of in-line documentation
 - examples of code indentation/ readability
3. Bessler, LeRoy. 1999. Strong Smart Systems: Software-Intelligent Development for Reliable, Reusable, Extendible, Maintainable Applications. Proceeding of the Twenty-fourth Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 80-88.
 - general concepts for robust application development
4. Newhouse, Russell. 1997. Validation and SAS Programming: Benefits of Using the System Life Cycle Method. Proceeding of the Twenty-second Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 104-108.
5. Some web sites that contain information on Software Development Life Cycle

System Development Approaches

<http://www.uwp.edu/academic/mis/baldwin/sysdelec.htm>

IEEE – helps to create standards for Software industry.

<http://www.ieee.org>

Software Research Institute

<http://www.soft.com/Institute/index.html>

Efficiency Considerations

6. Ma, J. MeiMei and Karp, Andrew H. 1997. Efficiency Ideas for Large Files. Proceeding of the Twenty-second Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 353-361.
7. Lafler, Kirk Paul. 1996. Gaining Efficiency with SAS Software. Proceeding of the Twenty-first Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 1577-1582.
8. Optimizing systems performance
<http://www.sas.com/partners/enterprise/ibm/optimize.html>.
 - discussion of options and other efficiency techniques
9. SAS Programming Tips: A Guide to Efficient SAS Processing. 1999. SAS Institute, Inc. Cary, NC.
10. Gilsen, Bruce. 1998. SAS Program Efficiency for Beginners. Proceeding of the Twenty-third Annual SAS Users Group International Conference. SAS Institute, INC. Cary, NC. pp. 310-319.

Trademarks

SAS® and all SAS products are trademarks or registered trademarks of SAS Institute Inc.

Meridian Software, Inc.® is a registered trademark of Meridian Software, Inc.

Contact Information

Caroline Bahler

Meridian Software, Inc.

12204 Old Creedmoor Road

Raleigh, NC 27613

(919) 518-1070

mercceb@meridian-software.com