Choosing an Application Development Tool for Your CANDA

Barry R. Cohen, Planning Data Systems

ABSTRACT

Many pharmaceutical companies choose SAS™ Software for CANDA development, but there are, of course, other application development tools to consider. A company must consider which tools produce effective graphic user interfaces (GUIs) and which tools produce systems which are easily maintained and upgraded. Many decisions are based upon the historical use of the software within the pharmaceutical industry, the programmers' levels of skill with the software, and the current software used for analytical clinical databases.

These and other issues might be included in your decision-making. Some of the general issues this paper addresses are:

- Does the tool speed up design prototyping and production coding?
- Does the tool facilitate long term life cycle applications, avoiding the need to constantly re-write the system as needs change?
- Does the tool provide optimal functionality?
- How does the tool ultimately effect regulatory agency review?
- How important is the choice of tool compared to the choice of system design?

This paper will be of value to anyone wrestling with the choice of software environment for CANDA development, and especially to anyone whose analytical clinical database is SAS-based.

INTRODUCTION

The choice of a CANDA development tool is not simple. There are many issues to consider, especially when CANDA use is viewed more broadly than just a system used externally for agency review. And, of course, the more broad the view, the more complex the choice is.

I do view the subject of CANDA use quite broadly in this paper. As such, I will raise many issues, but I cannot cover every issue of application development tool choice in depth. However, I will do the following:

- Present the broad context of CANDA use
- Present and discuss a representative variety of issues relevant to the application development tool choice
- Discuss my experience using SAS Software as a CANDA development tool

BACKGROUND

The early concept of CANDA systems was that of a tool to be used by the review agency, at the agency, to review the New Drug Application (NDA). The intent of the tool has been to reduce the time required for the review process. Today, the concept of CANDA use in the pharmaceutical industry is becoming more broad. The new focus is on internal uses for the CANDA system, prior to, during, and after the agency review process. This broader focus has evolved naturally because whether one is developing or reviewing an NDA with the aid of a computer system, the data are about the same, the documents are about the same, and the programs are about the same.

CANDA system users need access to data, documents, and programs, both to use them as is, and to modify, save, and use new versions of these items. Hence, people have begun to consider that the same system can be used internally before the NDA filing as well as for the NDA filing. And this thinking has also extended to system use to field questions from the agency during the review, and after marketing approval from a variety of sources.

In essence, the idea has evolved to develop the CANDA system first for internal use to help prepare the NDA, and then to spin off a subset system for external use at the review agency. The expectation is that the internal system will be of value during the NDA development, and that the spun off external system can then be developed at much lower cost than if built from scratch.

And as this thinking has evolved, many people have also begun to feel that in spite of a CANDA’s (hopeful) value in speeding up the agency review, that its greater potential value is its internal use. This is because the effort and cost to manage and analyze clinical data for an
NDA is so large, and any good system that can support this process will be a very valuable asset.

A large part of this cost is associated with ad hoc custom program development which supports data coordination, medical program coordination, medical data review and statistical analysis. A CANDA system can turn this ongoing custom program development, by costly programmer staff, into a process where the data and medical coordinators, the physicians, and the statisticians support themselves with a pre-programmed system. That is, the CANDA can be the self-guided work station of the future for these varied staff who are all involved in the NDA development.

Further, since preparation of an external CANDA system is itself a costly internal task, an approach which makes this task easier to do is also seen to have much internal value.

One last area of internal value comes from the fact that when the external CANDA is essentially the internal CANDA, then the agency reviewers are using exactly the same data that was used internally. This minimizes those "little" discrepancies between the company’s written findings and the reviewer’s findings that crop up during the review and are anything but little to resolve.

Many pharmaceutical industry people believe that the potential savings from use of the CANDA system internally are actually quite large, precisely because the ongoing custom program development cost is quite large. And because they feel that the way to realize this savings is to integrate the CANDA into the drug development process much earlier that at agency review time, the acronym of CINDA has begun to appear as the successor to CANDA. CANDA, of course, means Computer Assisted New Drug Application, and emphasis is on the time of application to the agency. The new acronym, CINDA, means Computer Integrated New Drug Application, and emphasis is on early integration of the system into the NDA development process. Some people view this as part of the topical subject of “re-engineering the drug development process”.

More on the Broad View

Table 1 summarizes the broadened view of CANDA use today. But note that all my comments so far about CANDA use have concerned when in the NDA development process the CANDA is used. It is now time to expand and consider, for any given point in the process, what part of the work is being supported. And that leads to a discussion of the various types of CANDA’s. I am aware of three types: (1) the data access or data review CANDA, for data review via ad hoc queries and via count and descriptive statistics reports; (2) the statistical CANDA for review of the statistical analysis; and (3) the document CANDA for review of all the documents of the NDA.

One or more among these three CANDA types can be developed for the pre-clinical phase (pharmacology/toxicology), the chemistry and manufacturing phase, the clinical phase, and the combined safety assessment. It is interesting to note that, to date, most CANDA work outside of document CANDA’s has been in the clinical phase, and all three CANDA types tend to be produced for this phase. And it is not uncommon for the clinical data access and clinical statistical CANDA’s to be combined in one system.

This grouping of CANDA’s by type and loosely by work phase is not perfect; in reality the boundaries are not always clean. However, I believe it is useful to view the CANDA world this way when considering what application development tool to use. It should help you to keep the big picture clear as you consider the details of CANDA development at any given point in the process or any given part of the work.

Narrowing the Focus

Obviously, this big picture is quite big. I cannot cover all aspects of it as I move on to discuss the issues of selecting an application development tool in this paper. My focus will be on the clinical data access CANDA and the clinical statistical CANDA, considering their use at four points in the process stated above:

- internal, during NDA development
- external, by the review agency
- internal, during agency review
- internal, post marketing

But it is important to note that several of the comments I will make about development tool selection for CANDA’s of this type will be relevant to development of other types of CANDA’s, too.
DEVELOPMENT TOOL SELECTION ISSUES

The tool selection issues presented here are quite varied and you will probably not find them all to be of equal importance in your situation. But each one will probably be of enough import to some of you to factor it into your decision of a software tool for CANDA development.

As I discuss each issue, I will make one or two comments about how SAS Software performed for me in this regard during two major CANDA development projects. I hope this will be useful information to the CANDA development community. Perhaps other CANDA developers will follow suit and evaluate their experience with the tools they have used for CANDA development. This way, we can soon have a larger body of knowledge to draw upon for our subsequent work.

1. GUI Functionality: The development tool should provide a full set of GUI objects, from the functional standpoint, for building screen interfaces with the user. Most development tools will probably do this, but I mention this issue to distinguish interface functionality from visual appearance. Some tools will allow you to build interfaces that appear a little niftier or slicker than others, but this has little to do with meeting the challenge of CANDA development.

The challenge in CANDA development concerns an intelligent system design, one that is intuitive to use for data analysis and review. You need a system that does not require a complete, in-depth knowledge of the underlying database before the analysis or review begins. Such a system is not an easy thing to achieve, but it is imperative to do because a CANDA database tends to be large and complex, especially for a clinical CANDA. A user can easily become overwhelmed and bogged down in an ill-conceived system to analyze or review such data. This is true regardless of how slick the screens may appear. In short, slickness in the interface will never overcome...
complexity in the database. Only an intelligent design will do this. Choose a development tool that is fully GUI functional, and then concentrate on your system design. Forget slickness.

I found the SAS Software to be a full-featured GUI tool, about equally functional with other tools available, but certainly not the slickest screen maker on the market. I also found that what it lacked in slickness did not matter.

2. **Compliant with GUI Standards**: It is desirable to choose a development tool that allows you to build screen interfaces whose GUI objects behave in a way that is standard for the given environment. If you are developing on a PC platform this would mean being Windows compliant. And on a UNIX-based work station, this would mean being OSF/MOTIF compliant. This may be a bit more important than at first appears to be the case. The reason is that your users, possibly both internal and external, will begin work untrained in use of your CANADA system, but they may be familiar with products that are compliant with the given environment. If so, you can shorten their learning curve with your system, and lower their frustration level, by building a compliant interface.

I found with SAS Software that I was able to produce an interface that was somewhat Windows compliant, but not totally so. The SAS Institute actually increased Windows compliance in the release I switched to between my first and second CANADA projects (release 6.08 to 6.10), but I do not know what more is expected in this regard in future releases. I cannot comment yet on SAS’s GUI compliance in a UNIX-based environment.

3. **Data Querying and Reporting Power**: Your system development is not done once you build a functional (or even slick) and GUI-compliant interface to your data. You still have to do something with the data after a great interface is made to it. Important value in a CANADA stems from analysis conducted on data subsets created from queries. This includes both summary count tables and statistical procedures. Many application development tools do not provide even simple count and basic descriptive statistics report tools. They instead require access to external products, such as spreadsheets, that do. There are two issues to consider here.

First, when more products are involved in your system, you will have to build more interfaces between the various products. Thus, it is going to be harder for you to develop the system. And, it will be easier for users to get into trouble when using the system, because they will be leaving an environment which you totally control for one which you do not. Granted, today's desktop software products certainly do make data exporting and importing easier than ever, but I still feel this is an important consideration. Hence, the more analytic and reporting capability that is built into the development tool, the better off you will be as a developer, and the more trouble-free your users will be when using these features.

Second, and more important, is the need to produce reports in a clinical CANADA that can count data by either patients or incidents (e.g., visits, adverse events) when both are present in the same data table. Very few products, if any, provide this capability as a built-in feature. Further, most products do not allow you to easily provide this, or at all provide this, by using some custom programming feature associated with the product. Nonetheless, this feature is extremely important for the type of data analysis and review conducted with a clinical CANADA.

I found that SAS Software provides an exhaustive set of reporting features built into their product. These are features that you can provide easy and reliable access to in any application you build with SAS. Further, I was able to handle the multiple-level counting issue (patients and incidents) in my SAS-based CANADA's through a combination of database design, custom programming using the SAS procedural language, and the SAS SQL procedure.

It is probably true that no other application development tool available today is as strong as SAS at combining general application development features and native analytic and reporting features. This is because SAS Software has added an application development tool (SAS/AF) to a long-established strong analytic and reporting product, while many of today's popular development tools have no such history. These products may have come out of a database environment, but they are still coming from the mold of more generalized application development.

4. **Associating Different Types of Data**: CANADA databases often have many data tables and complex relationships among them. An example of this, in a clinical CANADA, would be the relationship between a concomitant medication table based upon start and stop
day and an adverse event table based upon start and stop day.

The ad hoc combining of data from multiple tables such as these during querying is central to effective use of the CANDA system. But this combining must be based upon strictly defined relationships between the data tables, such as is done in a relational database model, to avoid both excessive processing time and overly large (and often useless) output. Hence, you need a development tool that is closely associated with a database querying system, so that you can pre-define these complex relationships. The key to easy and successful use of the CANDA system in this regard comes from this pre-definition, which removes the burden from the user at query time.

The SAS System is sometimes described as inferior to a robust database system because it is not a true database system. I found the SAS System, nonetheless, to be quite powerful and flexible for building a system that allows the user to easily associate the different types of data in a clinical database while querying the data.

5. Creating Accurate Count Reports: If data table relationships are pre-defined in a clinical CANDA database, then tables resulting from combining two tables during querying will be accurate at the table level. For example, if you make rules to join your patients' concomitant medication data and their adverse experience data by relative study day, then the concomitant medications and the AE's will be joined in a meaningful way during querying. So, if you look across any row of an output table from combining these types of data during querying, you will only see concomitant medications associated with AE's when they truly occurred in the same time frame. This data is said to be accurate at the table level, and this is a standard feature of most general query tools.

However, data can still be inaccurately repeated within a given column of a data table produced by combining data during querying. For example, if when querying the data, a patient has multiple concomitant therapies at a given point in time, then the AE's that occurred in the same time frame will appear multiple times in the joined table, once for each concomitant therapy occurring in the time frame. This creates an inaccurate repeat of the AE's in the AE column of the output table. This problem is actually a common occurrence when querying data with complex relationships, such as in clinical CANDA's.

The repeated data must be removed before the output is accurate, and it must be removed by the system behind the scenes, without any burden to the user. This is not an easy thing to do. It is not a standard feature of any general data querying tool. Rather, it requires custom programming.

SAS-based CANDA's will provide maximum flexibility, compared to other application development tools, to accomplish this type of processing. In contrast, most other application development tools do not provide this capability, and those that do tend to do so only with greater difficulty for you than is the case with SAS.

This subject might seem to be a finer technical point of data table combining while processing queries, and one without great consequence in the grand scheme of things. But, in fact, in the clinical CANDA world, nothing is further from the truth. Agency reviewers will try to replicate your published NDA results using your CANDA system, both to confirm that they know how to use your system, and to confirm that you have supplied the same data in the CANDA that you used internally to develop the NDA. Probably nothing causes more discrepancies between their findings and yours than their unexpected results when combining data tables during querying. It turns into a major headache (for you and them) to track down such discrepancies. At best, the result is a slower review process, and at worst, the result is a lack of faith on their part in your data and findings.

6. Ability to Switch Hardware Platforms: As we all know by now, nothing is as constant as change in our industry. The hardware platform you choose today for your CANDA may not be the best one for tomorrow. And this is all the more true when you consider that the CANDA system may have both internal and external use. Thus, it is highly desirable to choose a development tool that can build applications that can be ported between platforms.

One of the strongest benefits of using SAS Software as your development tool is its ability to run on all of the major hardware platforms in the industry. I believe that no other major application development tool can make this claim. Further, some of the hottest new application development tools may be restricted to only the PC platform running Windows. In contrast, your SAS-based application can run within Windows or OS/2 on the PC platform, on a UNIX-based work station, and on a VAX system.
You can even run your SAS-based applications on a mainframe (MVS or CMS). This is probably not too relevant given the GUI nature of the CANDA system and the need to deliver it externally to the review agency. But it does underscore the SAS Institute’s continuing commitment to provide a product that can be run on all the major industry platforms.

7. Cost to Develop Programmer Skills: Programmer skills are costly in time and money both to develop and maintain. If you are truly interested in building systems using skill sets that already exist in your staff whenever possible. It is also desirable to use industry standard development tools in case you must either contract, or hire additional permanent staff, to meet additional need.

Some of the hottest development tools available today are quite new and do not yet have broad-based industry use. Others are beginning to exhibit such use. Your internal staff may very well not be experienced in these tools. The interesting thing about SAS Software in this regard is its long history of high profile use in the pharmaceutical industry. This may offer your company a valuable advantage. Some of your existing staff may already be familiar with the SAS/AF product for application development. Others may find it easier (read that as faster and less costly to you) to add the SAS/AF skill to their existing SAS skill set.

8. Ease to Build, Maintain, and Enhance: Obviously, we are all looking for development tools that are easy and fast to use, and which can build systems that are easy to maintain and enhance. Several of the application development tool available today are based on an object oriented approach. Perhaps the primary benefit offered by object oriented development tools is their ability to design and implement systems that are easily maintained and upgraded.

And several application development tools are delivered in what is called an integrated development environment. While there is no one definition of an integrated development environment, the concept is one of integrating into one place a series of tools that make the developer’s job easier and faster to do. Some key components of such an environment include:

- use of an object oriented approach;
- easy on-screen access to the attributes of any given object of the application;
- provision of objects that pertain to the underlying database (in addition to the more traditional provision of those that pertain to the graphic user interface);
- control over versions of the code, with check-in and check-out, to support a multiple developer environment;
- on-screen visual editing which allows the developer to more quickly specify object behavior and relationships;
- easy on-screen access to an object’s method code.

SAS Software is not today a leading product regarding either an object oriented approach to development, or an integrated development environment. The product is object oriented regarding the graphical user interface, including the ability to define custom screen objects and custom methods for these objects. But SAS is only beginning to become object oriented regarding the underlying database (with release 6.11). Thus, you are not yet able to fully gain the object oriented benefit of more easily-maintained and easily-enhanced CANDA systems when developing with SAS Software. Further, SAS Software is not yet a full integrated development environment. My discussions with the SAS Institute indicate that improvements are planned in this area; however, none are present in release (6.10), and whatever is planned for a full integrated environment is not yet available in release 6.11.

9. Software Licensing Cost: There are two different issues to consider here. The first is fairly straightforward: some development tools cost more to license per copy than others. The second concerns the ability to license a tool in development mode (i.e., using it to its fullest extent), and to also license it in run-time mode (i.e., using it to a lesser extent only to run an application that has already been built with the product). Development licenses tend to cost more than run-time licenses. And with some tools, applications can be delivered in a way that there is no licensing cost at all just to run an application already written.

The importance of license costs is a function of the number of licenses that a company will require. Generally speaking, it will be of less importance when CANDA use is viewed more narrowly as pertaining to systems used externally for agency review. Here, the number of run-time licenses could be measurable, but it is not likely to grow to very large numbers for most companies. It will be of more importance when a company begins to integrate CANDA’s internally,
providing them in run-time mode to large numbers of statistical, medical, data coordination, medical program coordination, and regulatory staff.

It is difficult to make definitive comments about the licensing costs when using SAS Software versus other products as your CANDA development tool. There are many variables involved that make the landscape uneven for comparison. But I can point out the following that may be helpful to you:

- The SAS Institute does not currently have a run-time license for running applications built with their product. The same license is required whether you are developing an application or just running it. Thus, you may wind up paying more per user to just run applications built with SAS Software than with other development tools.
- On the other hand, when you license SAS on a system only to run a SAS/AF application (and not to build one), it is not simply the case that you are over-paying for the software because you are not using all its development-mode capabilities. Rather, the application you are running is likely to be engaging other components of the SAS System (i.e., various reporting components) that are provided to you with the same license.
- In contrast, if another development tool did provide a less expensive run-time license, but if it did not provide an adequate data reporting tool such that you had to add a third-party product (e.g., Excel) to the system to cover the shortfall, then you would have an additional cost there.
- So, in short, the landscape is truly uneven here, and you have to carefully consider the full picture before drawing comparative conclusions.
- But difficult as the comparisons are, they should be considered, especially when broadening the CANDA to internal use, because of the potentially large number of licenses involved.

10. Effective Prototyping Tool: There is probably no better way to communicate with users as they express their concept of what the system must do than to show them their ideas in a mocked up system. Thus, system prototyping is critical during design. And, since ideas evolve over generations of expression, the prototyping must be done quickly enough to be able to iterate over all the needed generations until the final concept has evolved. If the prototyping tool is slow, there will not be enough time to fully evolve into the final system, and the resulting production system will be less than it could and should be. This process is sometimes referred to as RAD, or Rapid Application Development.

There are two approaches to system prototyping: (1) simulated prototyping, where screen images are built but no real work is going on behind the scenes; and (2) functional prototyping, where a working system is built in an abbreviated fashion. I found the SAS Software to be an effective tool for both of these types of prototyping, and especially for functional prototyping, which requires a lot of background processing (i.e., database and reporting related) which SAS is quite strong at.

11. Data Migration: If you use separate systems, with different data management and access tools, for the internal NDA development and for the external CANDA, then you will have to migrate the data from one data management system to another. This can be a very costly undertaking. Thus, there is a distinct advantage to choosing the same data management system for both internal and external CANDA use.

When the source of your data for internal analysis and review is SAS data sets, as is often the case with clinical trials data, then a SAS-based CANDA will save you the difficult and costly requirement of migrating that data to another database environment for your CANDA.

A Final Thought

A CANDA is a very interactive system, and when we hear the word “interactive” we tend to think of the foreground processing that a system does with a user. However, it has been my experience that even in very interactive systems such as CANDA’s, that the strong majority of the code that is written (upward of 2/3’s to 3/4’s of it) concerns the background processing. That is, it concerns the data access, the ad hoc querying, the removal of repeated data after combining data of different types during querying, the data subsets that are saved and retrieved, and the count and statistics reports that are produced by the system.

Thus, when you consider what tool to use for CANDA development, you must remember that you are looking for a tool that performs well at far more than building user interfaces. Since SAS Software has a long and rich history in background processing of all types, and has now added a nice graphic user interface tool, I found it quite effective as a CANDA development tool. I feel that
any CANDA developer will do well to consider the balanced perspective of both foreground and background processing when evaluating the merits of application development tools for CANDA development.

SUMMARY

As a CANDA development tool, I found the SAS System to be: (1) a fully functional GUI tool; (2) fairly but not completely compliant with GUI standards; (3) strong at data querying and reporting; (4) strong at associating data of different types, as is so often the case in clinical CANDA's; (5) strong at the custom programming required to create accurate count reports by dynamically removing repeated data created by combining data types during ad hoc queries; (6) able to build systems that can be migrated across all the major hardware platforms available today; (7) a savings in the cost to train programmers because SAS Software is already commonly used in the pharmaceutical industry; (8) not as strong as other tools in providing object orientation throughout the product and at providing a fully integrated development environment; (9) possibly more costly to license over many users when those users just need the product in runtime mode, but not necessarily so when the full picture is considered; (10) an effective rapid prototyping tool; and (11) able to avoid a costly data migration for the external CANDA when the analysis data used internally resides in SAS data sets.

REFERENCES


SAS and SAS/AF are registered trademarks of SAS Institute, Inc. in the USA and other countries. OS/2 is a registered trademark of International Business Machines Corporation. Windows, Windows NT, and Visual Basic are trademarks of Microsoft Corporation. * indicates USA registration.

ACKNOWLEDGMENTS

I wish to thank Mr. Frank Palmieri of the clinical CANDA development team at Merck & Company for his informed and critical review of my work. His ideas and suggestions have been quite valuable to my paper. However, any errors or other shortcomings in the paper remain mine.

For more information about this paper, contact the author:

Barry R. Cohen
Planning Data Systems
PO Box 666
Ardmore, PA 19003
610-649-8701

Mr. Cohen is an independent information systems consultant, specializing in interactive application development using object oriented programming and client/server solutions. He has been using SAS software since 1989 in a wide variety of industry applications, including a focus on the pharmaceutical industry.