Paper 48-26

The Phoenix: Creating Applications which Outlive Current Users

Jeff LeSueur, BMG Direct, with Destiny Corp.

ABSTRACT

The Phoenix is a mythical bird which is periodically reborn, to live beyond generations of mere mortals. As developers and business managers, all of us harbor the desire that our applications will also be phoenix-like. While gratifying for the developer, the Business Manager truly benefits from applications that have a useful life beyond the last delivery milestone.

Accomplishing this goal today means far more than writing easily maintained and bug-free code. Today's challenges involve choice of platform, database design, user interface *medium* - the web or 3rd party tools, and application development products. Development cycles need dramatic fore-shortening if a product is not to be outdated before completion.

These challenges can be overcome by adopting a broader view of a project, careful selection of major components, increased planning for integration and turnover risk, and experience in a suite of development languages and operating systems. The broad array of SAS® Institute software products, from Base to SAS/AF® and SAS/WebEIS®, provide distinct advantages here.

A case study will be used in demonstrating how a top down design approach with an integrated suite of products can result in a more satisfying development solution, expected to outlive current users and perhaps the developers (who are aging) themselves.

INTRODUCTION

Significant demands are placed on developers today, particularly lead designers and project managers. Experience is mandatory in not just one language and one operating system but in fact a suite of development languages and their capabilities. For example Base SAS, SAS/SCL, SAS/AF®, HTML, JavaScript, JAVA, XML, UNIX, Windows/NT, and Novell will be required for many larger scale, enterprise wide applications.

How these components are applied to solutions today, and how the input, processing and output will be seamlessly combined poses another risk in development: integration. Standards and partnerships between major application providers sometimes do not quite accomplish this goal, a point that is not generally discovered until the pieces are 'press fit' in the early integration phases, or which haunts a completed project through successive "new release version" testing of separate components.

Integration of actual components into a final deliverable can surface the risk of unmet expectations on the client side. Despite all best intentions of both parties, even the most completely specified and well documented project can fail to meet the client expectations.

Developers also have expectations, expectations of delivering stunning functionality earlier than expected or at least within a few weeks of plan. None of us hold fondly the memory of late night(s) facing seemingly intractable problems, or trying to maintain credibility with the client in the face of long delayed deliverables.

Not too long after a project is delivered and the client has begun to enjoy the benefits of new functionality, the maintenance issues begin. Business environments are dynamic, projects have to be adapted to keep up. Making changes to complex projects can be easy if the original developers are still available, but when they're not, maintenance and adaptation can be expensive, time consuming or simply not possible with available resources.

All of these risks can be mitigated by a combination of common sense steps, a few essential software development features, careful project management, and two human characteristics common to successful software developers: perseverance and ingenuity.

Mitigating the risks will result in a more complete application, probably at lower cost, which will develop higher end user satisfaction, over a longer period of time.

Introduction of Case Example

For the discussion that follows, a specific example is referenced. This is an actual case study, one which in fact prompted the idea for this paper. Details of functionality for this example have been intentionally excluded as not relevant. This is not an atypical example, the authors have together seen many similar examples in their respective experience.

In this case, a legacy database application was in need of replacement. The database was large (45 gigabytes), the platform was expensive to maintain (weekly updates on MVS), and new features such as web publishing were not available for this application. In fact, the application was no longer supported by the vendor.

This application acted primarily as a repository of data on marketing campaign performance. End users retrieved the data they needed (VSAM) and performed analysis and reporting through excel spreadsheets. Since the repository incorporated both detail records and many levels of summarization, and was very responsive, the overall functionality for the end users was satisfactory. It was however costly to maintain, requiring weekly manual maintenance for new hierarchical information and data cleansing, an effort involving approximately 1.5 people.

The end users wished simply to replicate the existing database application on a less costly and more flexible platform. They had not requested expansion of functionality: they were not developers and did not have the requisite background to perceive potential opportunities afforded by today's technology in their day to day activities.

Controlling Project Scope

At the outset of a project, the one, two or three primary goals should be established. The earlier in the project these goals are identified and agreed upon, the more likely it will be to actually achieve them. Expanding a project beyond – sometimes way beyond – the original goals is a frequent cause of failure and certainly behind escalating costs of delivery.

On the positive side, identifying and agreeing upon the fundamental goals enables development to proceed unencumbered from design to execution and delivery. When the goals are fluid, projects move continuously from design to execution and back to design. Most frustrating for all is the time then spent re-designing components which have already been designed.

Agreeing on the goals early is a method to control scope creep. The goals can be repeatedly surfaced throughout the project to enforce upon the client (and sometimes developers) project priorities. Clearly, developers and project managers must not be tempted to introduce new features and functionality in the middle of the project, simply because the timing seems to be right. Knowing the priorities at the outset of design will also ensure this early stage of the project is completed.

That being said, the case study in hand went exactly the opposite direction in the middle of the design stage, and the case is made here that this was appropriate. While maintaining priorities is essential, developers – at the design stage – must not be blind to clear opportunities to help a client, where the client may not be able to see these opportunities for themselves. At design stage the project should be established in its larger setting to legitimize the scope of the project, as well as its place in the larger scheme of the business and related business processes.

Case Study: Design

Following the decision to replace the database, design requirements were established after a thorough review of the database, source data and update process. During one of the review sessions with end users, some questions were posed on how the information in the database was actually used, primarily to assess alternate storage and retrieval processes. Rather than simply mimic the existing database structure and copy the old data to a new platform, the idea was to understand how the information was used, and thereby improve how the information was stored and delivered.

From these initial questions, it became clear that there was a much broader application for the information and the project than simply a response platform for a few queries. The database was the foundation for an entire business planning process, involving results projections, financial forecasts, business plans, as well as marketing campaign planning. These much more business essential functions were carried out in the language with which the end users were familiar: manually intensive PC spreadsheet applications (Microsoft Excel).

The ensuing conversations with more end users further clarified opportunities for improving these business processes. Most of the improvements entailed replicating current (manual) spreadsheet functionality into databases and automating refresh scripts.

By clearly defining the expanded functionality, it became equally clear that the initial project could easily be expanded to include much of the present manual processes into a very sophisticated and interactive database solution.

TOP DOWN SYSTEM DESIGN

With a functional design specification in hand, and with agreement from all parties on the priorities and sequence of activities, the implementation of the design could be introduced. In order to accomplish delivery of a truly long lived solution, it is more important now to approach this problem from the top down, identifying major functions and their appropriate implementation platform, language and interaction medium.

The primary components of the system design are the Network Environment, central processing Platforms, Data Storage and Database Design, Processing Language where appropriate, and the User Interface Medium.

All of these components can involve products from multiple vendors. This introduces the risk of integration. Starting with a proven suite of applications, provided by a single vendor such as the SAS Institute, can greatly aid in eliminating this risk altogether. This in turn allows the developer and system designer to focus on meeting end user requirements in a seamless solution that goes farther than ‘skin deep’.

In the case study, the base project platform was to be a UNIX/SAS environment, providing ample storage and expected response capability for the database aspects of this particular project. The user interface aspects would be accomplished through a combination of web based reports, HOLAP data tables, and integration of content from several external databases.

The challenges would be the conversion of the original 45 GB of source data, integration of external content, the scheduling of table updates, and facilitating access to a number of data tables from several sources.   Some code would be required in the processing of the data, in producing consolidations for example.

SAS Warehouse Administrator could provide a number of valuable features in such a distributed environment. In establishing a metadata based environment, knowing the actual location of information/data is no longer necessary.   Accessing multiple database formats such as Oracle can be accomplished with SAS Access tools. Finally, the scheduler function can be used in scheduling the various updates required.

The one area in the case study which created a more significant challenge was that of incorporating manually generated content and developing a method for end users to interact with and update that content. This functionality would move the project beyond the area of relatively simple database functionality, with automated updates driven by scheduled development applications. It would require a method of enabling users to interactively access data, define and modify calculated values to create business forecasts, and save the results. It would require a multi-dimensional table or set of tables, multi-dimensional viewing tools, a process for controlling consolidation of individually prepared forecasts, in short a much more sophisticated user interface.

While several approaches were available for creating this functionality within an application, it was eventually decided that **this** functionality **was** beyond the scope of the project. The original priorities were recalled and re-established. To achieve the originally stated goals, more advanced functionality was put off to a later date, ie, **after** the initial goals were achieved.

In summary then, the platform for this project was established, and the tools for development identified. Database development, multi-dimensional table forms, user interaction, and web interface tools would all be required in delivering this project. At this stage in the project, having a suite of applications available from one vendor was very reassuring.

DATABASE DESIGN

Having defined the major functions, and expanded the functionality to encompass the broader end user applications, the database design could be refined. From the original mainframe based data tables, a careful evaluation determined that the data was not normalized. A single record included some summary data and some detail data. In addition, the format was very ‘horizontal’: lack of adequate dimensions in the original database had resulted in a dimension being incorporated ‘across’ instead of ‘down’: sales of several types of product and margin on these sales was broken out as separate variables, arranged as follows:

SalesA SalesB SalesC CostA CostB CostC TotalSales TotalCost

By normalizing the data and adding two more dimensions, several benefits were gained: a much simpler design would ease maintenance and provide for easier expandability for new products, and a summarized form of the data would take up much less disc space, dramatically improving system response. For the users it was clear that analysis of Sales by Product was much less important and less frequent that analysis of Total Sales and Total Margin. Hence the more frequently accessed data would be available that much more quickly, improving productivity.

Proof of Concept

To establish confidence that the initial design was in fact achievable using the proposed tools and processes, a complete solution was developed based on sample data. Each of the stages of database development, consolidation, reformatting and end user deployment were built, tested and delivered to the end users for their review.

Because sample data was used, and because the suite of products chosen eliminated integration risk, this ‘straw man’ process was completed in a very short period of time. That being said, participants in development exercised perseverance in bringing the demonstration solution to a successful close. That is to say, there were a few late nights required.

Sample data provided a distinct advantage: concrete data against which to develop the major components of the system with a minimum of invested development time, *a proof of concept*. Samples which are wisely chosen will surface inconsistencies, incorrect assumptions, and validate major design premises. Samples also provide tangible evidence of development progress, progress which can be demonstrated to end users by providing information which they use and recognize, albeit in a new form, on a new platform. The use of sample data in any project development phase cannot be over-emphasized.

The Big Pieces

In the course of development there can be a tendency to ‘get everything 100% correct’ at each stage. This is a natural, but myopic approach. The project can be easily delayed by an inordinate focus on issues of less importance. Formatting of output for example, or the look and feel of a user interface, can take considerable time to establish. In the meantime the data consolidation which feeds the content to that screen may not be complete! It is better to identify and successfully implement ‘The Big Pieces’ – such as the major data manipulation steps - and save the more minor issues – such as screen formatting - for resolution at a later date.

Similarly, project deliverables should be structured on an incremental basis, preferably one major deliverable per month. Monthly deliverables are going to be more tractable, more manageable than larger increments, hence they will have a higher likelihood of taking place.

Correspondingly, a sequence of successful milestones on this basis establishes credibility with the team and the client more quickly. Credibility earned earlier on the project can be ‘banked’ for that certain future when at least one deliverable or expectation is not met.

Perseverance and Ingenuity

It was said earlier that at an early stage in the project, in developing the ‘straw man’ or proof of concept, that some late nights were invested. While some developers may claim that their successes have not necessarily involved similar investments of time, ‘problems’ are an inevitable attribute of most projects.

Solving problems beyond the 9 to 5 day is not the issue, the issue is the recognition that the solution to more significant problems is through perseverance and ingenuity. These attributes require motivation, a commitment to success. Those engaged in projects, particularly project managers should ensure that the commitment to success is shared by all participants.   When it is, and when problems inevitably do arise, perseverance and ingenuity are more likely to surface, followed thereafter by the necessary solutions.

Maintenance

Simplicity and **Documentation** are the hallmarks and truisms for any project’s long term maintainability. Turnover associated with the explosion in web based services development has been very high recently. The risk posed in turnover impacts the long term applicability of completed systems. Simplicity and documentation are mandatory to ensure applications can live beyond the original development team members.

Modular design and carefully staged deliverables contribute to the simplicity of an overall solution. Choosing appropriate development tools can greatly facilitate the documentation process. For example, the model documentation provided by SAS Enterprise Miner is a major step forward, leaving the maintenance problems associated with user programs and undocumented code well behind.

Similarly the documentation which arises from the development process in SAS Warehouse Administrator is far better for maintenance than the plethora of UNIX scripts normally associated with large, multi-table database projects.

In the case study, several benefits arose from the emphasis on simplicity. For example, the original system focused all of the data in one very large table. The table had reached capacity, it was not possible to add more fields, neither was the data structure itself sufficiently flexible to make such additions manageable.

By normalizing the data structures, a new table was identified as being the logical structure for ‘Product Attribute’ information. Ironically, It was this information which users had the most interest in expanding. Through a simpler, logical data design step, the door was open to improving the availability of information on Product Attributes. It was further determined that many of the previously unavailable data fields were already being provided on the original data sources, they simply were not being saved in the current system because of the capacity issue. Hence a new ‘Product Attribute Repository’ was created, providing still further expanded functionality for the users and a simpler, more easily maintained data structure.

Final Results

The original project envisioned the transfer of one large (45 GB) data table to a new platform. The current system design involves a set of twelve integrated tables, some in multi-dimensional format, some in ‘standard’ form, with at least four unique interactive user interfaces, encompassing five different business requirements. Almost 20 analysts will completely re-define their day to day business activities, from maintaining a significant array of spreadsheets to developing analysis of business variances and recommendations for improvements in profitability. Business Planning exercises which previously required several months of time from all participants will be reduced to several weeks. Development of new employees will involve learning a documented database application instead of an array of spreadsheets and their tenuous spider web links. What was previously only documented in the minds of the analysts will be fully documented in SAS program steps.

Unexpected Benefits

While providing a design review for current users, a different management group expressed interest in a purported new application, which was described as a having a completely different set of functional requirements. By keeping an open mind and with irony being uniquely in a position to understand both user requirements, it was soon clear to the development team that satisfying the requirements of the first application in fact became the base for satisfying the requirements of the second application. Completing the second application would therefore be far less costly and development would proceed that much more quickly. It was clearly ironic that neither group of users was completely aware of how the source data for one project could provide the raw material to satisfy both requirements.

CONTACT INFORMATION

Jeff LeSueur
The SAS Institute
SAS Campus Drive
Cary, North Carolina 27513
919 531 0108

Dana Rafiee
Destiny Corporation
100 Great Meadow Rd, Suite 601
Wethersfield, Connecticut 06109-2379
860 721 1684
www.DestinyCorp.com