Paper 106-2023

Mining for SAS Gold

Tom Mannigel

Author:"FAST Solo Development"

ABSTRACT

SAS applications can be a gold mine for SAS developers, their managers, and their customers. The author Tom Mannigel knows this because he has spent over four decades working as a programmer building success SAS applications some earning millions of dollars. In this paper he describes how to find those high value SAS applications.

ABOUT THE AUTHOR

Since the paper is mostly about recommendation from the author, I think it appropriate to introduce the author first. For over four decades, Tom built software for enterprises ranging in size from a dentist's office to sixteen Fortune 100 companies. Most of his projects used SAS®. Since his career's launch in 1970, he has made 122 solo software applications at a 94% success rate. His first job was as a programmer in two aerospace research departments. After that, he worked at Exxon in several research groups for eleven years. He left Exxon in 1988 to become an independent SAS computer programmer in Houston, Tx, spending thirty years as a solo programmer on over a hundred projects. In 1988 he and another Houston based SAS consultant sponsored the first regional SAS conference. He was active in the South-Central Regional SAS group (SCSUG) for several of decades.

In 2014, after retiring and moving to Reno, Nv, he has applied his many years of experience and research, along with some serious retroflection, to complete his book, "FAST Solo Development" A Stepwise Method for Crafting Single Programmer Projects Rapidly.

His book describes a software methodology that single programmers' can use to excel.

INTRODUCTION

In this paper he provided a list of places to look(dig) for potential high value SAS applications.

Examples of Toms' SAS Gold mines

To establish my credibility, I would like to begin by discussing some of the million dollar and high value applications I built.

Million Dollar Example: High-Impact, Solo-Coder Application

My first software contracting job was with ARCO Oil and Gas Co.. The budgeting department called on me to solve a major problem: They had to develop a budget for the $500 million company at the end of every year. They manually combined 100 spreadsheets from each oil field into one spreadsheet for the entire company. This process meant the entire budgeting staff spent a miserable holiday season preparing for next year's budget so they could do their jobs.

My job was to automate that process, making everyone happy. This solo coder project not only solved a significant problem, but it eventually was also used to help run the entire $2 billion company, a prime example of how solo coders can have a significant impact. The ingredients for success were:

1. A customer group who had a desperate need.
2. A customer with the knowledge of how to solve the problem.
3. A project with very high-level support (the vice president).
4. A solo coder with the needed skill level could build a system at one-tenth the cost of a consulting firm.

This project was significant because it was built by a single programmer at cost of $250K as compared to $10 of millions by a consulting firm which normal would have been the case for a project of this magnitude.

Outrageous Return on Investment (ROI)Example

While working for a convenience store division of an oil company, I created an application with one of the highest ROIs of a solo-coder application imaginable. One of its IT managers was concerned about installing a new revenue posting system at its 300 convenience stores. The system provided minute-by-minute data about the sales at each of its stores. Even though the company that installed the system promised no data gaps, the IT manager worried the new system would miss sales. Then he asked me to create a program to track the revenue stream and detect "no sales" gaps.

Given the high traffic in a convenience store, there should be no gaps in sales. It was a simple program, taking me about ten hours ($500) to build. To our amazement, there were lots of gaps in sales, but not for the reason we thought. With further investigation, the company found the gaps were because employees were turning off the system and stealing the revenue. In six months, for the 300 stores, I found that $128,000 might have been lost because of the new system's vulnerability. Because it costs $500 to fix it, that is a 24,000% ROI based on the losses saved for the next six months revenue.

Saving Millions Example

For another large oil company, I built a system to simulate oil field production. Oil field production simulation is a very computation-intensive process requiring massive computer resources. In the early 1980s, Cray computers called "supercomputers" were used because they had a much greater capacity than the standard IBM mainframes of the day. The problem with Cray computers is that they are costly to run. Simulating a large oil field's production could exceed $100,000 in Cray computer time.

Using a numerical approximation approach, a colleague derived a shortcut to the simulation process. It was less accurate than the full-scale simulation but was good enough to use as a starting point for the full-scale model. Instead of costing over $100,000 to run a simulation, the cost was in the $100s. The process I built was used for decades and saved millions of dollars in Cray time.

This is another example of a high-value and high-impact solo coder project. This application was exciting because it was an analytic application three decades before analytics became common in the industry. It is also an example of digital masterminding (described later in this book). My associate had expert knowledge of the oil flow processes in an oil field. I turned his mathematical knowledge into code.

WHO DOES THE DIGGING?

Anyone from executives to staff can have an excellent SAS Gold idea. IT and operational departments may have a formal process to identify possible projects, but ideas can come from anywhere in the enterprise.

WHERE TO LOOK?

Select a project using one of the following processes:

1. Create a computerized uplifting application where the application uplifts staff to a higher level of operation (upstaff).
2. Do your homework: Ask your employees for ideas.
    a. Ask employees.
    b. Survey employees.
    c. Observe employees.
3. Create a digital mastermind system where a coder creates something based on an expert's knowledge.
4. Improve present information systems.
    a. Do away with mundane, repeated tasks.
    b. Perform legacy code analysis.
    c. Convert reports to graphs
    d. Reduce recurring crises.
    e. Eliminate manual steps between computer systems.
    f. Link users to data
5. Look outside the company.
    a. Your competition
    b. Consulting firms
    c. Contractors
6. Projects never to consider.

A DOZEN TRAITS OF ONE SUCCESSFUL MINER

1. CREATE A COMPUTERIZED UPLIFTING APPLICATION

To uplift someone, you work with a technical or business expert and convert that expert's knowledge into a computer program they can train staff to use. That computerized uplifting allows a staff member to utilize the expert's knowledge or empowers them and be uplifted to a higher position in the company. There are two types of uplifting applications:

A *functional uplifting application* is where an IT application computerizes an upper-level staff member's function so that a more junior staff member can carry out the task, thus uplifting the junior staff member to a more valuable and influential position.

A personal uplifting application is where an IT application gives a staff member more function and makes them more valuable.

Personal Experience: Functional Uplifting Applications

Functional Uplifting (Example 1)

A. Sales quoting systems. I automated two sales quoting systems. For the old quoting system, sales managers used spreadsheets to create quotes. The sales manager selected items for the sale and entered them into a standard spreadsheet to create the quote from several sources. I built a replacement application that automatically merged the sources and loaded the selected items into a spreadsheet. Finally, those selections were summed up and reported. The new quoting application was faster and more accurate than the manual system. It also meant that a staff member could do the quote, thus freeing the sales manager's time and uplifting the staff member to a more critical role in the company.

Personal Experience: Functional Uplifting Applications

Functional Uplifting (Example 2)

B. Clinic quality management. In another uplifting example, a pharmaceutical quality manager's job was to audit and evaluate the quality of the medical clinics that ran clinical trials for the company. First, she gathered operational data about each clinic. Then she entered that data into a spreadsheet. Finally, based on her expertise, she evaluated the quality of each clinic. I designed the new system, gathered the needed data, and presented it in interactive graphs and reports. The new system was more accurate than the old one and saved time. It allowed her to train her staff to audit and evaluate the clinics' performance. The application freed up the manager's time and uplifted the staff members' value and importance.

Personal Experience: Personal Uplifting applications

Personal Uplifting

Resource production management. On a project, I worked with a large consulting firm where they created a resource production management system that empowered a single staff member to schedule the entire company's production. Previously, an entire department would schedule the production, not a single person.

This example shows how an IT application can uplift a single staff member's importance and value.

2. ASK EMPLOYEES TO FIND PROJECTS

Every company has vast numbers of undone, high-value projects. They have more potential applications than any software development department wishes to develop. It is a matter of identifying those projects and choosing the highest value ones.

Locating worthwhile projects in the enterprise is a simple process. Have the company's employees who understand the company's problems tell you what's needed. The company's employees have a keen sense of what's going on. They know what's going right and what's going wrong. They understand company processes. They recognize sloppy processes and have an excellent sense of how to fix them. There are ways to find out what the employees are thinking:

Ask them. Just sit down and ask the employees' opinions by asking what needs improvement and how to improve it. Everyone likes to give you, their opinion.

Survey them. A more formal way is to create a survey, have employees complete it, and follow the money.

Observe them. Observe employees doing the most valuable functions and see if there are ways to create software applications to improve those functions.

3. CREATE A DIGITAL MASTERMIND SYSTEM.

Figure 1 Einstein

Digital masterminding creates high-value applications. The staff works with an expert to create an application, and the application converts the expert's knowledge into code. The code then improves the company's operations.

Every company has experts who know how the company runs. Most of the time, they are in senior positions with many years of experience. Over the years, I've worked with many experts to develop software that improves operations.

Digital masterminding examples:

A. Marketing director

The Mastermind: A corporate marketing director with strong sales analysis skills had many ideas for improving sales but no time.

The programmers Contribution: Working with the director and their assistant, I built several analysis tools using graphs and spreadsheets. These tools improved the sales of a large oil service company.

B. Banking VP

The Mastermind: A banking VP with strong loan marketing skills and a sound vision of what was needed to increase the number of loans made.

programmers Contribution: I put the VP's vision into graphs and reports that went to bank managers to help them make more loans. The effort created $100 million in additional loans.

C. Oil Company President

The Mastermind: President of an oil company who wanted to see the company's revenue from a unique perspective.

Programmers Contribution: Most oil companies' revenue systems consider income for oil fields as the sum of oil and gas. The president wanted the income separated. Where was the revenue for gas produced, and where was the revenue for oil created? I fashioned a program that separated oil and gas revenue. The president used this information to find several unprofitable fields and sold them using the new perspective, saving millions.

D. Complex Process Simulation:

The problem: In the 1970s, I was the sole programmer on a Laser Isotope Separation project. Laser Isotope Separation is a complex and challenging process that uses high-power lasers and metal evaporation equipment to enrich uranium. The highly complex process had to be economical for the project to succeed. My boss's challenge was to evaluate the complex processes involved from an economic standpoint.

The mastermind: My boss Charlie Lindenmieir was a Ph.D. in theoretical physics with twenty years of industrial experience and was very skilled at deriving mathematical models for complex processes.

The solution: He derived mathematical models using differential equations that simulated processes whose complexity was compared to landing a man on the moon. He then linked those models to the economics of the process.

The programmers' contribution: The mastermind had wild ideas. I was good at creating computer code for his wild ideas. I could link his complex mathematical derivations to his economic assumptions. We could then simulate a highly complex process and evaluate economic alternatives from the code I created.

The result:

- We performed analytics very early in the '70s.
- We performed process simulations very early also in the '70s.
- We combined economics with a physically complex process.

What to look for in Digital Masterminding

- An expert with idea.
- A high-skill specialist, such as a mathematician, statistician, or operations research expert, who creates the approach to solve the problem.
- A programmer that implements the procedure so users can use the created approach.

5. IMPROVE PRESENT INFORMATION SYSTEMS

All companies have systems that need improvement. The following are examples of how to do that.

Do away with mundane, repeated tasks. The pattern to look for here is to eliminate a mundane task where someone is doing a miserable, menial function that you automate, such as repeatedly manually building a spreadsheet, copying report entries into a spreadsheet, or doing the same manual calculations repeatedly.

***Perform legacy code analysis*.** Companies create hundreds of reports and spreadsheets using legacy systems that can be decades old. Review that legacy code and check if any of the hundreds of spreadsheets or reports produced are not used. If they are not, delete them. If they are being used, improve them by:

1. Using analytics to increase the knowledge value.
2. Changing a set of reports to an interactive dashboard.
3. Create a dashboard for the President of the company to review monthly.

Convert reports to graphs. Some groups have reports that no one reads. Converting these reports to graphs presents the information in a more easily understood form. Even if people use the reports, making graphs of the reports will make them easier to understand and improve insight.

Reduce recurring crises.

Some companies have a recurring crisis every month. Usually, the crisis is because data is unavailable, or a process fails consistently. A recurring crisis signifies that something needs to be fixed. Solo coders can improve them and eliminate recurring crises.

Eliminate manual steps between computer systems

Sometimes computer systems do not communicate with each other. Someone manually copies files from one system to another. This wasteful process also introduces the possibility of errors, so you should eliminate the process by automating the linked process.

How to find manual steps**:** A manual step in an IT process involves producing a file or report and then feeding the results to another process. The solution, in this case, is to combine the processes. Here are some examples on the next page:

PROBLEM	SOLUTION
Enter results from several sources into a spreadsheet.	To overcome this problem, link and combine the sources. Then make a single spreadsheet.
Construct a spreadsheet, then create a graph in it.	Creating the graph when you build the spreadsheet resolves this issue.
Linking spreadsheets to one another.	To correct this case, create a joint spreadsheet.

Figure 2 Eliminate Manual Steps

In each of the above cases, eliminating manual transfers saves time and reduces errors. Sometimes, removing manual steps frees up senior staff time and allows other staff members to do the senior staff's work. This function I called the computerized uplifting applications.

6. CREATE DATA LINKS FOR USERS.

Users sometimes need detailed data that is outside their regular reports. They must make a special request for IT support to get that data. Building a user interface that allows them to access the information they need without help will save IT support time and give more power to the end-user. The following are types of user data links.

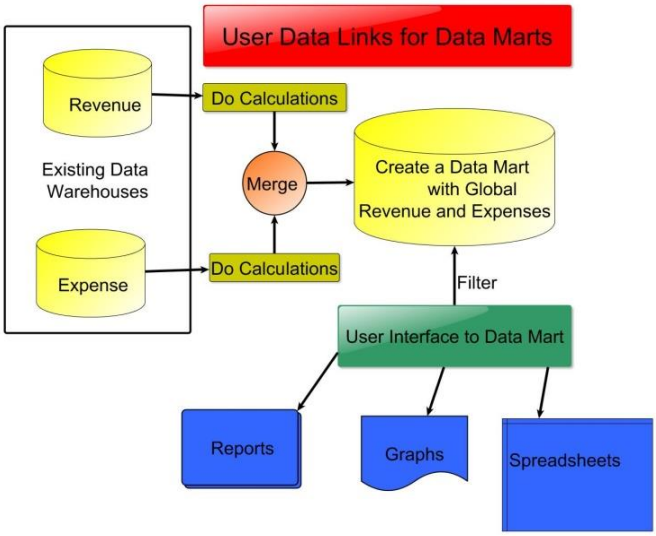

Figure 3 Data Links to Data Mart

This process links and combines different data systems into one datamart, then you create a user interface to the data mart that filters and selects the data you want (illustrated above).

If the data warehouses are significant, it is impractical to create a data mart. In that case, create a user interface that links directly to the warehouse, as shown below.

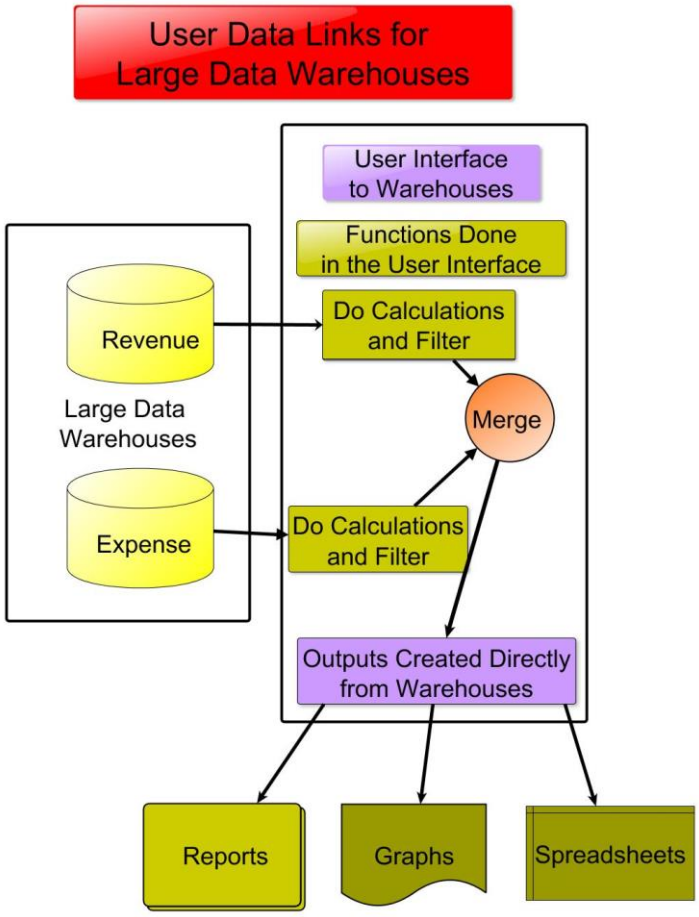

Figure 4 Data Links to Data Warehouses

7. LOOK OUTSIDE THE COMPANY

THE COMPETITION:

The competition is an excellent source of ideas projects that can be found when attending conferences and reading technical literature. An added benefit is seeing what the competition is doing.

Consulting firms:

Use a consulting firm to look at your processes and develop ideas you can implement independently. I worked on several projects that an outside consultant firm suggested.

Find a project then use Contract Programmers:

As a long-time software development contractor, I suggest that more companies work with software development contractors.

Advantages of Contract Programmers

A. They have a specialty that you do not have.
B. They can devote more time to programming than staff programmers because they are not part of the company and do not have to deal with company administration or other costly overhead practices.
C. They can advance the technology further to return value because of their experience.
D. A contract programmer should be noticeably more productive than an in-house programmer.
E. They can provide extra resources when resources are short.
F. Teach you new technologies.

Use contractors judiciously.

Some companies know how to manage contractors and do excellent jobs (See the example below). Although contractors are expensive, companies should get a high return on that extra expense if used correctly. Contractors are revenue-generating assets rather than a liability. Managing contractors well takes time and a commitment to those practices. When you use contractors, do not squander their time; ensure they are always busy and making worthwhile efforts.

Personal Experience: Getting the Most from Consultants

A multi-billion dollar privately held company that I contracted with used contractors and consulting companies extensively and got excellent results.

The clue to their successful handling of those people was that they had a group specializing in using them. That group knew how to manage and maximize outside contractors' and consulting firms' results because that's all they did.

I also struggled with other companies that had many headaches using outsider help. They had complications because they were unfamiliar with contracting and needed a process to handle them. The more you work with contractors, and outside consulting firms, the better you get. Start small and work up.

Do not use contractors for the wrong reasons like:

- Interfacing with unreasonable people.
- Having someone to terminate when times get tough.
- Boosting the body count after people get laid off.
- Inflating the size of the organization.

8. PROJECTS NEVER TO CONSIDER

Projects that Reduce staff: Never undertake projects that hope to save money by reducing staff. Who builds an application that wipes out their job or their colleague's job? Solo coders are very vulnerable, even on a typical software project. A project that reduces staff would be a terrible job, if not career-ending.

Instead, identify projects that increase the value of employees. Projects that reduce staff will at most provide a small percentage of improvement. Projects that increase productivity have no limits.

TRAITS OF ONE SUCCESSFUL MINER

Personal Experience: Why I was a success

Since most of the presentation is about my experiences, a reasonable question is why I was successful. Let me outline a dozen reasons I can think of:

1. I loved building software applications. Notice I said building and not seeing the results of my effort. Usually, I never got to see the finished product. You build it; you make sure it works correctly, then pass it on.
2. The individuals I worked with had a precise vision of what they wanted.
3. I had a clear goal and a plan for building software applications.
4. I never got into the business of fixing the organization I was working for. Most of them needed to be fixed, but that was not my job.
5. I never compromised my integrity. I abused no one, and no one abused me.
6. The projects I worked on were small > 1 programmer year.
7. I never pushed the technology too far. I tried new technologies but never to the extreme.
8. Being a good system analyst, project manager, and a really good programmer helped.
9. Good interactions, managing expectations with customers, and always ensuring they were involved was essential.
10. Minimized management involvement but getting them engaged when help was needed.
11. I could always get myself out of trouble by working just a little harder for a short time. I never got myself into too deep of a hole.
12. I never worked on Sunday. Working Sunday means two weeks without a break, which is too long.

Conclusion

There is gold in those SAS hills.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Tom Mannigel

Author of “FAST Solo Development” A Step-Wise Method for Crafting Single Programmer Projects Rapidly.

Phone: 832-722-6980

Email:tmannigel@aol.com

Trademarks