The Cutting Edge: An Adventure in Fourth Generation Software

John Bescher

Limited budgets, time constraints, numerous applications to define, develope, and test. Such a scenario is faced by Data Processing Managers every day. I was not alone in my dilemma. The basic goal is to become as productive as possible and to meet the changing needs of users. I had some experience with VAX Fourth Generation Software and had found it to be very limited in scope. I was somewhat convinced that Fourth Generation Software was merely a lot of advertisements about an "RPG" type of language...sure you could put it in the hands of users, but the rest of the system would surely suffer from overloads and poor resulting response times. And certainally, the capabilities were severely limited. At best such tools could only be used in conjunction with "normal" programming methodology: Cobol, Basic, etc. However, because of the large quantity of applications development backlog and the fact that I could not hire hordes of programmers, and because most of the applications were custom, I elected to invest in Fourth Generation Software tools. I wish to share my experiences with you on my successes and failures with Fourth Generation Software.

Before discussing my adventures with Fourth Generation Software I would like to briefly describe the environment. The applications and the users at the Signal Companies are rather unique in some ways and common in other aspects. The environment is identical to a small '150 employee company. There are needs for a Payroll, General Ledger, Accounts Payable application for the activities of these 150 employees. On the other hand, the facility is the corporate head- quarters of a 7 billion dollar, over 73,000 employees, corporation. Many employees are Very Important People who are independently wealthy and where salaries exceed $500,000 per year without counting bonuses. The General Ledger System must contain up to 9 billion dollars per line item or journal entry. The typical user is not engineering, nor technical, nor manufacturing oriented. In fact, most users are first time data processing users.

The Signal Companies, Inc is a multi-industry company with strong, strategic positions in the aerospace, electronic communications, energy service, transportation and construction industries. Signal provides sophisticated technology and engineering services and high-quality products to these and related industries through- out the world. Signal is 60 years old and has its roots in the oil and gas industry. Signal was formally known as "Signal Oil and Gas Company"

Signal is an exceptionally strong, high-technology company with powerful earnings capacity. It ranks between 45th and 50th on the 1982 Fortune directory of the largest U.S. industrial corporations. Signal's sales approximate $7 billion, its assets total $5.4 billion, with over $2 billion access cash and lines of credit. The corporate headquarters is located at La Jolla, Calif, a suburb of San Diego. I installed Fourth Generation Software at this headquarters.

With the construction of this headquarters in 1980 the Chairman of the Board and Chief Executive Officer, Forrest N. Shumway, created an environment that would provide comforatable working conditions and maximum efficiencies for the approximately 150 employees at the facility. These employees consists of very high ranking executives, attorneys, public relations specialists, and staff. Banking activies, tax concerns, stock acquisitions, divestitures, legal corporate decisions, mergers, corporate finance, insurance, overall company policies, investments, and employees savings plans are all orchastrated from this headquarters.

The Spanish-California style architecture was selected in conformity with the San Diego/La Jolla area. Combining location, which is on the Torrey Pines ridge overlooking North San Diego county, architecture, interior design, and landscaping, the result is one of the most outstanding and impressive corporate headquaters in the western states.

In addition to many private offices, a large dramatic lobby, this 100,000 square foot building also has an employees' dining room with a fully equipped restaurant-type kitchen. The forty foot long board of directors room is impressive with its massive arched windows. The facilities include a wine cellar, photolab, executive dining room, medical wing with a full time physician, nurse, and x-ray equipment. Employees can also use the recreational facilities during non-working hours: a heated pool, spa, 2 tennis courts, racketball court, and exercise room with body building equipment.

This environment emits a feeling of quiet power and wealth. Being from a classic data processing environment, I was somewhat concerned about fitting in...will I spill coffee on the customized Persian rugs? How will these VIP's relate to "computer" technology? Will computer equipment and operations be delegated to the basement out of sight of the high ranking executives and their priceless antiques and paintings? On the other hand, I was intrigued on how computer technology could improve operations and efficiencies with- in the headquarters.

I was hired early in 1981 to create a data processing capability within the headquarters. Certainally there was some data processing operations already available: batch processing on a timesharing system. Computer capabilities were limited to card oriented batch input via a Remote Job Entry station with resulting printouts. Most of the applications were manually manipulated: A Payroll system, General Ledger, and Accounts Payable on the remote timesharing system with a mixture of customized Cobol applications. The Hewlett Packard System 3000 had arrived in a crate during my interview. It was sitting forelornly at the end of the first floor hall when I arrived for my first day at Signal. Smith Dennis and Gaylord, a software vendor in Santa Clara, was contracted to provide most of the financial software: Employee Savings Plan, Executive Stock Awards, General Ledger, Payroll, Accounts Receivable, Accounts Payable, and Fixed Assets.

I drafted a five year plan and distributed it to the staff outlining the directions for the new Hewlett Packard System 3000. It consisted of three phases: First the installation of the financial packages from Smith Dennis and Gaylord, next the procurement of Fourth Generation software tools to create the "other" applications that were either weakly defined in the prior discussions with the software house or not defined at all, and Third to create a link with subsidiaries via telecommunications. At this point the plan has been beat...the telecommunications system is underway using Remote Job Entry, IBM personal computers, a bank of modems, switches, modem eliminators, and a centralized tandom pair of IBM 3033's at Signal's Kellogg Computer Services in Houston, Texas.

Against this background, I now would like to share with you my experiences with Fourth Generation Software. The envirnoment is a very powerful corporate headquarters with a limited staff. Users are non data processing oriented. Probably typical. They had suspicions that Data Processing people were high technology "egg heads". Data Processing was very expensive, reports were never quite right, any development request would be met with blank stares or impossibly long lead times. Prior to the advent of the Hewlett Packard System 3000 the Cobol programs customized on the timesharing system were sometimes over 6 years old...and of course the programmers were long gone. Operations were based on folklore, heresay, guesswork, reruns, and card decks used over and over again with little understanding of what was actually happening. Timesharing rates were increasing with little control by the user. All data and input were the responsibilities of the Data Processing staff. Lastly, even the easiest appearing change or request for a new report was looked upon by the Data Processing staff with horror. Within the first few weeks of my job, I was asked by a senior vice president for a report showing tax data sorted a different way then the usual report. I thought the task was reasonable, but to my embarrassment found that the key needed to sort the data was not included in the data base, nor would ever be in the data base without a lot of conversions, program design and recompilations, and tests. I found the report would take several weeks to develope using the existing serial file structure on the timesharing system and Cobol. The VIP, of course, needed the report for a meeting the next day. Without going to extraordinary efforts (emergency consultants, an all night effort, etc) the report would have to wait. I vowed to remedy the problem with a Fourth Generation Software tool on the Hewlett Packard System 3000. Eventually all applications would be removed from the timesharing system...thusly, all effort except emergencies would be put on the System 3000.

In selecting a Fourth Generation tool I examined several packages. Utimately I selected Quasar's Powerhouse because I felt I needed ease of use and friendliness above all....even above computer efficiency, response time, etc. I was in a hurry to impliment some applications quickly with as few resources as possible. In retrospec, I believe any of the major packages could have been used successfully at Signal. I still think, and this is just my opinion, that the Powerhouse package is the most friendly, complete, and practical for my environment. Lastly I wanted a package that I could safely share with some of my users. I wanted some of the non-data processing employees to have report generating capabilities and perhaps, even creation of applications. My ultimate goal was to leave the data base definition, initialization, and maintenance to Data Processing and let more sophisticated users generate screens and reports for the data base. The vendor did not believe this would be a good idea. They felt users were capable of inocently destroying data, errorinously using Powerhouse, and in general, causing an uncontrolled series of data processing catastrophies. I began to feel the mistrust was equal on both sides: users thought programmers were arrogant, overpaid, and incapable of a sense of urgency on even the simplest request. "Computer" types thought users were scatterbrained, disorangized, unsophisticated neophytes incapable of understanding the complexities of data processing.

In a sense, my comments are not only directed towards the adventures of Fourth Generation Software, but also in reducing this attitude friction between users and Data Processing staff.

For those who have not been exposed to Powerhouse it consists of:

-A dictionary
-QUIZ: A report writer
-QUICK: A screen generator used to create menus and screens to create, maintain, and delete data base records.
-QTP: A program generate that can be used to make sweeping changes in a data base, editing an entire file, or producing logical updates, additions, or deletions in a data base.
-A series of utilities to compile, list, and integrate dictionaries, screens, and programs.

Powerhouse is capable of defining, creating, and maintaining data bases for MPE, KSAM , and/or Image files.

Both ample documentation and a call-in consulting service are available.

The Lure and the Promises of Fourth Generation Software!

I had expected Fourth Generation Software to reduce definition, creation and testing time for customized applications. I also expected a resulting application that would be homogenous to all users, easy to use, and a breeze to maintain. By homogenous I mean the same key functions, help message operations, format of screens, exit techniques, etc. between all applications. A user would not have to be retrained when going from one application to another. I would not have to remember, for example, that "exit" ended one screen, "End" exited another, the "F1" key another, and upper case "6" another.

In the almost two years of using a Fourth Generation package I found that the biggest gain was creation of "ad hoc" applications. Often, I would receive a vague description of a desired application from a senior VIP and turn over the completed application to his secretary in just a few hours! The speed of developing such "ad hoc" applications is very gratifying to both programmers and the user.

On the other hand, there is a big need to improve definition methodology. I do not know which methodology is best. Interviewing users is the first step in understanding an application. Unfortunately there is the innocent unsophisticated user who knows the application but who forgets to tell the developer all the vital details which if not included in the data base makes the resulting application worthless. The other type..on the other end of the range...is more sophisticated and wants to know why the masters are such, and why detail files, and why not put these 200 fields on one record, etc. They volunteer too much and thusly bog down development time. Endless discussions develope with these users about whether to put parameters in a control file or embed them in screens. This user seems to always have new ways of producing "better" application updates, reports, logic, etc. Inevitability one must either ignor such a user or give in to a particularly outlandish idea just to prove to the user it was a bad decision. Either path is a poor choice. And sometimes this type of user is correct...which feeds his appetite for more information on how the system works. Unfortunately this user sometimes neglects his real job for the intrigue of data processing...then in the midst of an application that he has masterminded with his own techniques he has to return to his primary job. At that point the project is orphaned.

Definition, then, is a still major task; and if not successfully completed, will be a fatal flaw in the project. The major benefit of Fourth Generation Software for definition phase is the ease of creation...and resultant review by user. I hope the universities help us in providing better ways of defining an application because the state of art right now still leaves a lot to desire. The gap between user and developer is very wide here. Not only are there misunderstandings, wrong assumptions, forgotten conversations, and changing requirements; there is always the very real possiblitily of the user not really knowing the details of the application. It seems humans are not very good at details. Computers need details.

A Commitment in a Corporate Environment

When I purchased the Fourth Generation Software package I made the commitment to myself that I would use the package for all new applications on the System 3000 unless an application was already available from a software vendor. I had been told by my friendly salesman that the package could do anything. This was a challenge to me...I had to find an application that Fourth Generation Software (at least this product) could not solve. This excluded communications software such as RJE.

In meeting this commitment to use Fourth Generation Software for all new applications I had to resolve several issues:

-Is Fourth Generation Software really practical for all applications? In other words, are there some applications that are too complicated or require logical operations not available with Fourth Generation Software?

-In spite of the fact that Fourth Generation Software requires less resources per application, I still need resources (people) who know both the Fourth Generation Software and the application requirements. Where will I get those resources?

-In charting this path, will I end up with an overload to the system? Is there a way I can measure the load on the system and thusly change my goal before it is too late?

The results? The commitment was kept. I found many applications that initially were too complicated for Fourth Generation Software. I found several old forms and reports that could not be identically duplicated with Fourth Generation Software. I experimented a lot. I called the software vendor often. When stuck on a particularly sticky problem I retained a consultant from Los Angeles for a week. In the end, I found solutions for all applications through Fourth Generation Software. I admit, sometimes I had to spend extra time with the users explaining why I could not produce an identical report, but we always found a compromise. And sometimes the compromise report was a better report than the original.

The solutions to resources and performance fears are many and direct: In order for Fourth Generation Software to work it must have a driving force: an individual who is determined to use it and will encourage others to use it. I became the champion for Fourth Generation Software. After attending vendor's classes and using it myself for several applications, I taught local users including my programmers and operators. I set up example source statements to use as standard screens for others. My biggest disappointment became the lack of local consultants. I could retain Cobol, Basic, Data Base experts...but in San Diego I could not locate any consultants for short term work who were knowledgable with my particular package which, by the way, is the most popular Fourth Generation Software package available for the HP 3000. Through the assistance of the vendor, I was able to locate several experts in Los Angles. I hope with the growth of data processing in San Diego the availability of such consultants increases.

As far as performance fears go, I could not find a fool-proof way of determining how badly I was loading down the HP 3000. Sure, I could measure percentage of time spent on I/O and CPU utilization. But, response time is a very elusive concept. I still have no way of knowing that I will, for example, increase average response time by 100 percent next month based on the past year's increases. In fact I do not even know my past year's increases. At this point I have several jobs (Basic programs written by consultants before I purchased the Fourth Generation Software package) that run for hours and hours. However, in general, I have not had complaints from any users on interactive response time delays! Will the very next application installed cause unacceptable response times for every user? I honestly do not know. All I can state, is that in spite of over 12 major applications written in Fourth Generation Software, none have ever been unacceptably slow or caused deteriating delays for other users.

To date the following applications have been created at Signal headquarters using Fourth Generation Software:

-Medical Insurance Claims
-Subsidiaries' Insurance Contributions vs Claims Cost
-Self Paid Insurance Cost History
-Public Relations Multi-Industry Mailing List
-Cash Management
-Accounts Payable Vendors
-Debt/Credit Data Base
-Dental Claims
-Income by Industry
-Bank Agreements
-Savings Plan Auxilliary
-General Ledger Auxilliary
-Payroll Auxilliary
-Stock Analysis

Planned applications include:

-Bank Accounts Directory
-Employee Expenses AP-AR Loop
-Company Car Maintenance and Cost
-Political Action Committee Eligibles
-Docket Scheduler
-Foreign Exchange Exposures Interface

The Cobol Trap. Conversions.

I had a choice. I could convert existing Cobol programs running on our timesharing system to Cobol for HP 3000 or I could use Fourth Generation Software tools to rewrite the applications. The programs had to run on the local HP 3000. The users were unknowledgeable and complacent...in the past they had merely filled out a form or called our Data Processing operations to run a job and, in return, had eventually received a resulting report. They could care less where the software resided. I was attracted to the apparent ease I could take these Cobol programs from IBM to HP. With very little effort I should be able to run these applications on the in-house HP 3000 by Cobol conversions...after all, one of the advantages of Cobol is its portability. I decided to try the "easy" conversion and then a Fourth Generation rewrite of just one of the applications; measure the results in terms of a product and time investment; and finally, decide which direction for the remaining 6 to 8 applications. Fortunately, I had joined the Hewlett Packard Users Group and learned about an opportunity at San Diego State University: For senior Data Processing students the university offered a course of field work. Students would get a chance to work in a live data center environment and earn credits. I could have students working on my experiments without any cost to the corporation! I assigned one group of students to the Cobol conversion. The next group to the rewrite in Fourth Generation Software. Two major applications were selected: a complex public relations mailing list of over 7,000 entries and a legal court docketing date scheduler. The result...the Cobol conversion too longer than anticipated and the final version on the HP 3000 operated just like the original timesharing version: batch with many restrictions: a poor product. Maintenance was going to be a problem. It took two students working 16 hours a week two months for both conversions.

On the other hand, the rewrite using Fourth Generation Software took the same time but resulted in a much more flexible mailing list application. During the rewrite of the court docket scheduler a major change occurred within Signal which made the basic applications applicability questionable. Essentially, it was better in the long run to rewrite using Fourth Generation Software than to attempt the conversion. With the conversions the users seemed to care less, with the rewrite they were pleased with the enhanced product.

My commitment to write all new applications with Fourth Generation Software was underscored! It works! I almost fell into the trap of the "easy" Cobol conversion...only to find it was not that easy and that the resulting software would (surprise) be no better than the batch 80 character input, non-inquiry, poor maintainable, timesharing version.

Plunging in where Angels Fear to Tread

Even the enthusiastic vendor salesmen was negative on letting non data processing individuals use the menu and screen generation Fourth Generation Software utilities. With these utilities a careless user could easy purge, destroy, change, etc data base information. The vendor guideline seemed to be "let the user have non-destructive report generation capability only". Only Data Processing personnel should have the capabiltity to create software that could change data. I presume this attitute was either to protect Data Processing personnel from being blamed for a destroyed file when the user did it unknowingly himself...or its the old job security syndrone. From the beginning I had envisioned violating this guideline if I could find some users who would have both the patience to learn about Fourth Generation Software and the common sense to keep himself from doing foolish things. For those individuals who were trusted with tools capable of changing data I made some disclaimers:

- I would be available as a technical adviser...but if questions became too frequent, I would take over the application.
- Only Data Processing personnel would be permitted to call the vendor for a resolution to a problem. I wanted all such calls to go through a senior analyst both to avoid unnecessary calls to the vendor and that any problems and resolutions would be shared...the next time that same problems occurred with a different user a senior analyst would already have the answer.
- Outside of guaranteeing the integrity of the hardware, backup operations, and Operating System soundness I would not be responsible for the data manipulated by the user created applications.

Users now having the capability and knowledge to create their own menus and screens include all data processing operational personnel, the Manager of Finance, the Assistant Tresurer of the corporation, and the Manager of Accounting. Thus far, no problems have been created via this arrangement. In fact, many of the former problems have vanished. If my resources are tied up in another application, some of these users can start their own scrèens and menus. This gives me some breathing room to finish one application and start the new one. Also, many users now appreciate the need to define applications before they are programmed. They have a better understanding of what it takes to develope an application.

I wanted to spread Data Processing appreciation to the entire staff. Because of the many manual operations within the facility I felt there were ample applications that could be computerized with Fourth Generation Software. The biggest problem I have, however, is defining those applications in sufficient detail to start the software creation phase. By holding seminars on Data Processing concepts I thought I could shorten the definition time requirement by educating users on terminology: fields, records, data bases, etc. Even such elementry concepts such as sort keys, numeric versus character, master records versus detail records, etc were foreign to these users. If users would understand these elementry concepts it would be easier for me to learn about their applications. We seemed to have two worlds: data processing terms foreign to users and application terms such as Coordination Benefits, Cut-out, etc that were foreign to me. The series of seminars; unfortunately, were unsuccessful. Most users did not have the time, were confused, promply forgot, could care less, or, in a few cases, had to learn everything about Data Processing for it to make sense to them. For those people, an incomplete understanding of all aspects of Data Processing seemed to make them apprehensive and uncomfortable.

In the future I will change my focal point for these in-house seminars. Rather than try to acquaint users with elementry data processing terms and concepts I will try to show them capabilities and possibilities of the system. Not how..but what. Not about signed numbers, master and detail data sets, why it is important to be able to identify each employee record by an identification number which is unique instead of the person's name...but rather what kinds of applications have already been created on the HP 3000 and the drugery and paper work eliminated by those applications.

I realize that changing the focal point of the in-house seminars will only increase the number of applications destined for the HP 3000 and will not help in the definition of those applications...but as I said before, I need a better methodology for defining applications...and seminars do not help significantly.

The Performance Poltergeist.

In the beginning I had opted for very friendly Fourth Generation Software over efficiency. There is a range of efficient software starting with assembly language and SPL through Cobol, Basic, Pascal and ending with Fourth Generation Software. Even with Fourth Generation Software some packages are more efficient than others. Generally, it seems the more friendly the less efficient the resultant software: efficient in terms of response time, load on the system, run time, etc. I had elected for friendliness in the extreme. I did not want to spend a lot of time learning about the new package...I want to create applications. I was nervously wondering if, in the long run, I would have to pay the price. So far, the threat of poor performance has just been a "poltergeist"...it has not materialized. One saving grace is that most of these applications in Fourth Generation languages are interactive, most have small data bases, and most are only weekly or monthly activies. I have not had a single complaint about response time or processing delays. Most of the terminals in the facility are still set to 2400 baud. In one case the user wanted to slow down the terminal because his reading speed was outmatched by the application.

To protect the system from future loads I have ordered disc caching for installation sometime in 1984. Because I have a basic model 40 (the least powerful processor) I can always upgrade the processor. Lastly, I can order more memory (I have 786K now) or another 404mb disc (I have 2 404K and 1 120 MB). With that room for growth I feel protected from any performance poltergeists.

A Stranger in a Strange Land: Interfacing Foreign Applications

One of the most productive and rewarding uses of Fourth Generation Software at Signal headquarters has been foreign software interfaces. There are several applications written by a software house in Basic using an Image data base. Maintenance of these applications is done by the software house. In many instances I have been requested by users for short reports not available from the existing software. There also have been requests for changes to the data base based on logical constructs that were not in the existing package. Without Fourth Generation Software I would have had to request the software house to provide those reports and programs...at considerable expense and time delays. With our Fourth Generation Software I could "share" the data base with the software house package and provide reports and generic changes not possible within the time constraints imposed by the software house. The purchase of the Fourth Generation package could have been justified just in this interface capability! I have used the report generator to test the software house's new releases by comparing our report against the software house's report. I have saved the corporation over $20,000 in "non-standard" reports....reports that were not defined when the initial application was given to the vendor for development. A new project is underway to provide an Accounts Payable link to Accounts Receivable using Fourth Generation Software.. such a link would have cost over $12,000 from the software house in its current undefined state. I am sure that cost would have been tripled after the definition is firm.

In many instances the use of Fourth Generation Software in conjunction with an existing application makes not only sense, but becomes mandatory for important, fast changing application environments.

User Involvement: How much is too much?

How much information does one give a non-data processing individual? How much capabiltiy? At what level do you keep data processing secrets? At Signal I felt that the HP 3000 system was common property...I did not own it, nor did anyone else. It is a facilty to be used by any appropriate employee. Data Processing was responsible for the integrity of the system, its operation, backup, current operating system, etc. Data Processing is responsible to ensure the correct tools, utilities, and methodogy is available and used. Security of the system is also a Data Processing responsibility. The content of the disc files and data bases, however, is the responsibility of users. I did give several non-data processing employees the capability of managing their own data...they could create their own menus, own screens, and own data bases in addition to their own reports. I have not had a single regret for giving these users such powers. On the other hand, only Data Processing personnel are assigned "manager" user-id's and password knowledge. Some of the adverse effects I have experienced from this arrangement are:

- I get a lot of telephone calls on "how to" questions. Most of the time I have the answers. For those times I did not, I found the resolutions and learned more about Fourth Generation Software. Such calls can be frequent, disruptive, and occasionally annoying. In the long run, however, more application work is being done then if I kept the work to Data Processing personnel only.

- Infrequently I become irritated that the software is not more user friendly. In spite of the quality of Fourth Generation Software there are many detailed technical implications...beyond what a normal user needs to know. For instance, "RTIO" must be specified in MPE files in order for deletions to work. Even Sometimes odd boundaries (versus even) causes system confusion for Image files. In the future, these "gotch ya" pitfalls will be eliminated from Hewlett Packard resident Fourth Generation Software making it totally tolerant for non-data processing people.

- Just like computer games and Person Computers, many users are intrigued by Fourth Generation Software. They want to know how, why, when, etc. When they learn they become "experts" and like real experts they do not take advise any longer...and run into unexpected (but predictable for the experienced) problems. Meetings with such users can be very lenghy. Instead of understanding the application being defined the meetings can disintegrate into a discussion and debate on computer methodogy...should this be a master, detail, auto, etc data set, how many passes will we need to generate the report, how many programs do we need? Sometimes the Data Processing staff can take offense at this intrusion into their domain. Often meetings become unproductive. Users begin to spend more time on the programming tasks than on their real job.

There probably is no real guide on how involved a user should be. In some cases, the user can be completely involved...let him write his own application. In other cases, especially with very complicated, far reaching, and large data base projects involving several departments, the design, development, and initial test should be solely the responsibility of Data Processing with the other departments assisting in the definition of the application.

Emergency Applications: Heros on White Horses!

There is nothing like being asked for an emergency software request and being able to deliver in an unbelievable short time. You become a hero on a white horse, a wizard. Somehow with some magic you did the impossible. With Fourth Generation Software such magic is possible. One afternoon I was asked to deliver to a very high ranking executive a summary of several interesting companies showing outstanding number of shares, price of stock, assets, etc. Such information which is publicly available would be used for Signal investments decisions. In two hours I was able to define and create a data base, construct menus, build entry and maintenance screens, and reports. The system was then turned over to his secretary for data input. Such a feat would be impossible in the amount of time for Basic or Cobol. Needless to say, the executive was impressed.

There are times when an application report seems to be erroneous or the information is available on disc, but no existing report shows the summary information just the way it is required at that time. At that point a Fourth Generation report can be quickly defined and executed giving the anxious user the information needed to find out why his report is out of balance, showing no entries, or faulty entries and summaries. Given the time spent on installing and testing the initial Basic programs, users are often astounded at the rapidity of a new report being created by Fourth Generation Software. My most rewarding and exciting efforts have been for emergency situations using the Fourth Generation Software tools.

Prototyping. Specificiations:

The Jellyfish Application

As I mentioned previously, the biggest problem I have is in the initial definition stages of a project. I tried a series of seminars so users could understand Data Processing concepts. That ended with what I felt were poor results. There was just too much to learn in too short a time. Most of the applications were started with several meetings..almost interviews..with users so I could understand the requirements. A prototype would be built and the users would review the structure. Changes would be made and the cycle would be repeated until, hopefully, the "correct" solution was available to the user. Problems I encountered are:

- Sometimes the prototype-review cycle would never end. Changes would be heaped on changes resulting in a very "patched" or "dirty" file structure. Eventually, the file structure would have to be redefined.

- Major changes often were reveiled by the user which rendered the original design obsolete. Sometimes these changes were simple omisions, but frequently they came from the increased awareness by the user of the capabilities of the system. "Wow! I didn't know you could calculate those summaries...That's exactly what

I need. Now, if you could only link with those summaries the subsidiaries' contributions.." Redesign, compromise or freeze the design! Because some users plead so well for a new found capability, often its redesign.

-In frustration, I have had to freeze the definition-prototype-view cycle for at least one major project because of the mounting changes that were occuring over several months. I regressed back to written formal specifications which were distributed to the staff for sign off. This has happened twice for this particular complex project. The user just could not make up his mind...or things changed. This project is now in hold pending formal signoff from the user. This emerging and changing design causes a "Jellyfish" project: half defined, half prototyped, ever changing, needed but obscure.

-Another potential problem arises when the user becomes too excited about the new application and begins to use it before its properly tested. Two frustrations are common with premature applications: 1) other departments get reports that are erroneous and therefore embarrassing (one user was so thrilled at the premature..and faulty...report that she hand corrected the incorrect totals on the report and sent them to other departments!) and 2) the user spends hours upon hours inputing all the data for the data base only to discover there are major changes to the data base. In that case, a conversion is needed instead of just re-entering a few test records.

-Always leave room in every record for expansion when prototyping...I define at least 20 characters per record for unknown, future fields. That way the data base does not have to be recreated for testing. One merely has to redefine the unused field for added status flags, company codes, totals, etc that were unfortunately overlooked in the initial design review.

-There is an axium in data processing that one must always throw the first design away. With Fourth Generation Software it is easier to throw the first design away and start again because the data base creation, menu, screen, and report generations are usually very easy to accomplish. Less is invested in system creation efforts, therefore the pain to discard the first version is bearable.

Junior Programmers let Loose

How much capability does a Data Processing Manager give to junior programmers? A little bit of knowledge can be dangerous. I have found most junior programmers to be very eager...full of energy. They are ready to conquer the world. They are confident they can do anything! And there lies the trap. Without realizing it, they get over their heads with complexity (as we all do occasionally) but rather than ask for help...or even recognize they are in trouble...they continue to storm ahead. I have had several student programmers work with Signal as a result of the San Diego State Universtiy student work program. In general, their accomplish- ments to Signal applications have been significant and I hope they have gotten a taste of the real data processing world by working at Signal. My observations are:

- We in data processing seem to be generally plagued with underestimating the effort to create applications. This becomes worse with junior programmers. In spite of Fourth Generation Software there remains many "gotch ya's" in data processing. Obscure problems that take time to resolve. For the students on loan to me from SDSU the underestimation was not only worse, but the students would return to school after a set time...whether the project was complete or not. It was up to the Data Processing staff to pick up the pieces. If such students are used, one must assume that the staff will take up where the students left off.

- The students (junior programmers) were assigned maintenance projects or were given models to use for new applications. This worked out very nicely, especially when software documentation from the vendor was comprehensive and descriptive.

- In the few cases where I let more aggressive students try to work with users in defining an application..and then work with me in laying out the data base, I found to be marginally successful. Users wanted to talk to more experienced data processing personnel. For some reason I felt a slight resentment by the users for the junior programmers. I could only attribute such resentment...and it was slight... to the fact that the junior programmers were just starting a career (Data Processing) that was very envious to secretaries, word processing clerks, and clerical staff members.

Conclusion: Promises Kept

By describing some of my adventures with Fourth Generation Software I hope you have detected my enthusiasm for such software in the type of environment at Signal: Standard vendor supplied financial packages, non-data processing oriented users, a small Data Processing department, with many customized applications requested by various legal, public relations, banking, tax, and financial staff members. In spite of threats of system overloads and applications complexity that could not be resolved by Fourth Generation Software I have found pleased users and more timely application development. I have cautiously allowed non-data processing personnel create their own menus and screens...with no regrets. I have let junior programmers maintain Fourth Generation Software applications with little supervision...with no regrets. And I have avoided, with one exception, the time consuming, dull, ritual of specification writing, review meetings, and reissues of specifications, sign-off, etc by wide use of prototyping...with no regrets. There are still a lot of pitfalls with Fourth Generation Software: technical aspects that need not be burdened by the user such as RTIO, Stack Size, Word Boundaries, and frequent need for double pass operations. However, we have come a long way from the laborious Cobol programs and reliance on programmers to create, maintain, and change user's own data.

In closing, I would encourage all of you who do not have Fourth Generation Software to explore the practicality of such a tool at your facility. You are doing yourself and your company an injustice if you do not at least investigate this proven way to make your staff more effective and efficient.