IMPLEMENTATION OF AN AUTOMATED CODE ENFORCEMENT SYSTEM VIA THE INTEGRATION OF THIRD PARTY AND IN-HOUSE DEVELOPED SOFTWARE IN A MIXED 3rd and 4th GL ENVIRONMENT

Kathleen P. Metz Edwards
City of Plano
Plano, Texas

INTRODUCTION

Before I launch into a discussion on the merging or co-developing of third party and in-house developed software, I will describe the process by which my organization, the City of Plano, Texas, decided to embark on this means of systems development. Following the discussion on why, I will discuss the implementation phase, as well as the benefits to the City and to the contracting third party vendor. Lastly, I will conclude the paper with discussion of the question, "Is it for everyone"?

DEFINITION OF PROJECT ALTERNATIVES

While many users may look at this means of project development as a substitute for good design work, they are mistaken. As with all systems development projects, the design phase is absolutely critical to the ultimate success of the project.

The process of automating the City's permits and inspections began with a definition of the current, albeit manual system. In 1985 the City's Code Enforcement Department manually issued nearly 12,000 building permits and tracked 98,609 supporting inspections to accommodate a 300 percent increase in growth. Permits were issued via eight separate documents and inspections were tracked using 13 different posting cards. The resulting paperwork was time consuming and inefficient. Documents were lost and inquiries were impossible.

An internal review of procedures was jointly performed by the Code Enforcement and Data Processing Departments. The review resulted in the replacement of the permit and posting forms by single forms designed for each function. These forms were implemented nine months prior to the installation of the software. This nine month leeway allowed the users to become accustomed to the new forms, without the added burden of automation. It also defrayed some of the fear of change that was to be expected in a project of this magnitude. In addition to training Code personnel, meetings were also held for developers and contractors as the new

forms would drastically alter what they had traditionally become accustomed to. Lastly, the City published documented working procedures in support of the streamlined processing. Construction guides for both residential and commercial projects were also published and released to the public in conjunction with the new forms and procedures.

Data Processing and Code Enforcement were then ready to begin the definition of requirements for an Automated Code Enforcement System (ACES). The requirements were compiled and a Request for Proposal was sent out to prospective vendors. At the same time, Data Processing, with the support of the users, prepared a Detail Design strategy, defining the systems' basic program functionality (attachment 1).

The systems' requirements were divided into five broad functional areas: major file maintenance and report programs, other file maintenance and inquiry programs, other file maintenance report programs, operational report programs and management report programs. Each broad functional area was then further subdivided into programs. Admittedly, we designated these subdivisions as "programs" for purposes of evaluation. In practice, if the City had decided on an in-house solution, a single "program" might have become several actual programs.

Each of the "programs" was assigned a "relative degree of difficulty", i.e., "H" (hard) - 10 days, "M" (medium) - 5 days, and "S" - 2 days (simple). The programs were then factored with the assigned values. Please note that the document was not a Detail Design. It merely identified the functions requiring design work and some "guesstimates" that would be used for comparative purposes. Using this methodology, we estimated we would need 245 days to complete a detail design document.

As a public institution, the City also had the option of reviewing and implementing public domain software. This option was considered in the evaluation process, recognizing that the software would require changes to meet our specific requirements.

The time estimates for the implementation of an in-house developed system and the three modified public domain systems were evaluated via the programming estimates form (attachment 2). A separate programming estimates form was completed for the in-house development option, as well as for the three public domain options. Analysis revealed the City would need an additional 640 days to code and test in-house developed programs. Time to code and test public

domain software systems included 447 days, 540 days and 580 days.

Vendor responses to the RFP were scrutinized in a similar manner to determine if the City would incur additional costs. For example, did the vendor include all travel and training expenses in his proposal? Were all reports coded or was the City expected to code them using a report generator or QUERY? Would the vendor provide source code? Was the system written in COBOL, IMAGE and V/PLUS? Would the vendor offer flexibility in maintenance contracts especially during the implementation phase? Lastly, if a vendor omitted any costs, they were added by the City as part of the evaluation process.

The City also applied an additional $34,545 to each alternative, i.e., in-house, public domain or vendor for parallel testing and system documentation.

DECIDING ON A COURSE OF ACTION

A cost analysis was performed on the options described above. The recap figures for the alternatives are attached (attachment 3). The cost of development ranged from $293,205 to $61,545. The City selected Interactive Computer Applications Development (ICAD) of Sarasota, Florida not solely on the basis of cost. ICAD was willing to work with the Data Processing staff to customize the software to meet the City's specifications and to provide flexibility in meeting the City's long term goals.

In addition to the cost analysis, I would like to briefly mention the goals of the organization in evaluating the criteria. Plano is a rapidly growing City. The City has many automation needs that have yet to be defined. While we knew in time the City would implement a parcel/geo-file system as a basis for permits, we simply did not have the time to implement a system of that magnitude - we needed automated permits and inspections yesterday. Therefore we wanted to make sure that the system we implemented would accommodate an interface to a geo-based system. Since we were not clairvoyant, we obviously needed the source code to ensure our ability to interface. Additionally, the City fully intended to assume the long term maintenance of this software.

IMPLEMENTATION

The system we selected met approximately 75 percent of our requirements as demonstrated. Namely, the system issued permits and tracked historical inspection data. The system

did not calculate permit fees or provide for on-line inspection requests, two critical requirements. Additionally, the software was built upon a land use parcel system. Since the City was not ready to implement parcel management at the time, it did not wish to pay for software that might never be used. Through discussions, verified by contract, the vendor agreed to modify the software to calculate fees and to process on-line inspection requests. The City agreed to develop two sub-systems, specifically, the Street Index for street verification and the contractor sub-system for registering and licensing contractors/sub-contractors. The Street Index system would be used in lieu of the parcel system. While the vendor's software supported contractor registration, it did not meet Texas legislative nor, Plano's City Council requirements. Lastly, the City agreed to interface the software to its existing systems.

A WORD ABOUT COMMUNICATIONS

I would like to digress here and speak briefly about communications. Before the project was completely implemented, it would touch on personnel in Florida - the selected vendor, Texas - the City of Plano and California - the 4GL vendor. Our contract with the vendor called for only two on-site visits; the costs of additional visits, if needed, would be borne by the City. During the vendor's first visit, we discussed the necessary modifications to meet our requirements. The vendor returned to install his portion of the software on the second, and last visit. All other communications were via the telephone or letter.

The need for good communications cannot be overstated. Coordinating the implementation of the software required many hours in preparation of written correspondence. I firmly believed that projects fail because they lack clear, concise, written communications. Programming has proven to be the smaller part of any project but frequently we have been under pressure to begin coding before questions have been answered and problems resolved.

All successful automated projects have required consistency in the definition of the user's requirements. As I have previously stated, this method of systems development was not a panacea. In any new project, the users needs must be outlined and agreed upon prior to beginning software development. Admittedly, this consistency was easier said then done.

The successful implementation of integrated in-house and vendor developed software was also dependent on the delineation of duties. Each entity understood fully what it

was to accomplish to make the project a success. This delineation was also contractually stated, however, if it was not understood, all the legalese in the world would not have gotten the job done.

Lastly, the successful integration of in-house and vendor developed software was facilitated in part by recognizing the need for a single contact point on either side. While several programmer/analysts worked on the project, all questions to the vendor were directed through a single individual. Along the same line, we directed all our questions to a single individual in the vendor's office. This arrangement worked well throughout the implementation process.

The system when it was implemented on November 1, 1986, consisted of 20,245 lines of on-line COBOL, 100 V/PLUS screens, and 39,267 lines of batch COBOL written by the vendor, and 52 Speedware modules written by the City to meet its portion of the agreement. The total elapsed time from the definition of manual permit processing to the implementation of on-line permit processing was 13 months.

OTHER IMPLEMENTATION REQUIREMENTS

Naturally, to be successful in this environment you must have access to an experienced HP programming staff. During the 13 month implementation period, there was at least one, and at times four members of the Data Processing staff working on the project. The staff members worked in the definition phase, as well as in the final implementation phase. Without the expertise of the various staff members, the project could not have been completed within the 13 month time frame.

Simply stated, if you do not have this type of expertise available, you would not be a good candidate for co-development.

INTERFACING VENDOR & IN-HOUSE SOFTWARE

The vendor sent the source code tape to the City several weeks prior to his arrival for installation and training. This lead time allowed the City the time to compile the programs in its own environment. We experienced only one compatibility problem due to a disparity in operating system releases, specifically, the vendor was on a later release of COBOL. The vendor made some minor changes and we were ready to continue.

CONTRACTOR/SUBCONTRACTOR SUB-SYSTEM

At the same time we were defining ACES, the City made the decision to replace its C/PM micro computers with IBM PC clones. Code Enforcement had a simple contractor/sub-contractor system in place on its C/PM micro that needed immediate replacement. Because of the short time frame, we developed a Contractor/Sub-contractor System using Speedware. We originally intended this sub-system to be a temporary one, however, as our evaluation progressed, we realized that the system we developed was preferential to others on the market.

Over time, the contractor sub-system has evolved into a full Contractor/Sub-contractor management information system. In addition to using the system for permit validation, it has also been used for a variety of other functions. Specifically, using Reflections, we have downloaded data to MS Word and sent out contractor renewal notices, newsletters, ordinance changes and a variety of other correspondence. This feature has allowed the user the flexibility to design and run reports independently.

Integration of this system with the vendor's software was accomplished without problem. We simply sent the vendor a copy of the contractor's schema. We also defined the edit criteria that we expected for acceptance of permit applications and for issuance of permits.

STREET INDEX SUB-SYSTEM

The Street Index Sub-system was established to verify streets in the absence of a geo-file. The Street Index sub-system was simply designed to be what its name implies - a listing of street names. We designed the Index to store the streets in geo-file format, i.e, storing street name, street type and direction. This format will allow the City to make the transition to a geo-file/parcel system as resources become available. The "Map Name" that appears on the reverse side of the City's map was also stored in the Street Index. All data elements were designed for maintenance by the Engineering Department.

Since the City did not have the Streets resident on the 3000, we uploaded the names from a Lotus file, formatting the records via Speedware. We also developed the Street Index in Speedware as we had a limited time frame and we knew that the City intended to migrate to a parcel management system in the future. While the Index is a simple "add", "delete", "modify" and "inquiry" type of application, it has served the City quite well. Due to the

overlap in school district boundaries, the City had experienced a problem in issuing permits outside it boundaries. The Street Index prohibited this error and provided uniformity in spelling street names.

We experienced no major technical problems in the Street Index integration, however, we encountered a minor problem in the handling of numeric fields in Speedware vs. COBOL. This problem has since been resolved; we circumvented the problem at the time by changing a particular numeric field to an alphanumeric field as the particular application did not absolutely require the use of a numeric field. We were then able to manipulate the data in the receiving COBOL program. As with the Contractor Sub-system, we forwarded a copy of the Street Index schema to the vendor. He made the necessary calls to the Index, and applied the desired edits for street validation.

TAX AND UTILITY BILLING INTERFACES

In order to issue a permit for an existing structure, Code Enforcement needed to validate the structure's existence within the City. While the City's Utility Billing System contained the address information on any structure receiving water in the City, Code Enforcement had no means to access the data. Code Enforcement also needed to determine the parcel's legal description so that the permits could be forwarded to the Central Appraisal District for tax purposes. While the Tax Master held this information, it was not readily available to any other department. Code personnel manually searched appropriate sub-division plat maps and recorded the lot, block and subdivision. This process was time consuming and resulted in permit processing delays.

These interfaces posed quite a challenge as Tax and Utility Billing were driven by disparate keys; the Tax key was a composite of the lot, block and subdivision - the very items that were unknown. Utility Billing's primary key was an account number that had no significance to any other City department. While its alternate key was street address, the streets in the system were not verified upon entry and therefore varied to exact nomenclature. The primary key to ACES was an application/permit number with an alternative street address key in the fixed geo-file format.

We could not seriously consider a conversion of either Utility Billing or Tax in the time frame that we had defined for ACES implementation. We solved the interface problem through the application of Speedware's "Speedex" to both systems. Since neither system had the security for outside

inquiry, we built separate "lookup" screens and files for each system, selecting required data elements and applying "Speedex" to owner name and address. Admittedly, this approach required a 50 percent increase in data storage however, it markedly reduced the effort necessary to process permits. Additionally, permit turn around time decreased and data accuracy improved dramatically. The "lookup" files are rebuilt through weekly batch runs. The Utility Billing "lookup" extracted and rebuilt nearly 40,000 active utility accounts, running in less than two hours; the Tax "lookup" extracted and rebuilt nearly 60,000 real property accounts, running in less than three hours. Both these jobs run on the weekends and have exclusive access to the HP3000; if run during normal business hours, run time will increase substantially.

We encountered some minor, but time consuming problems in establishing batch run streams for the "Speedex" "rebuilds". Speedware was designed for on-line processing and its manual provided little in the way of documentation for batch processing. Running the "rebuilds" on-line required the presence of an operator, a costly and in this case, needless expense. These problems were repeated with the installation of Speedware's version 5.0.

"Speedex" allowed the user to obtain required information without knowing the record key. It also allowed the user to process data using a "wild card" character, i.e., if the user knew only a portion of the element to be searched, the "Speedex" software returned records that had matching entries. Specifically, the Tax "lookup" allowed multiple key access via owner, property location or legal description; the Utility Billing "lookup" allowed multiple key access via customer name, property location, driver's license number or social security number. While the "lookups" were designed specifically for Code Enforcement, they are now popularly used by many other City departments that have the need to process owner/ occupant information. These departments include Streets and Traffic, Solid Waste, Planning, Capital Projects, Engineering, Health, Police, etc.

The "lookups" have been further enhanced to allow the users to selectively download names and addresses via Reflections to Micro Soft Word where they have been used for a variety of notification purposes. This feature has saved the City many hours in addressing and preparing correspondence.

As time permits, we will add "Speedex" directly to the Tax master and eliminate the need for the duplicate data. This effort will require some reprogramming to ensure security

regarding tax payments. There are no immediate plans to add "Speedex" to the Utility Billing master as the master is a KSAM file; Adding "Speedex" to the UB master would require a migration to IMAGE.

Although "Speedex" has proven to be of great benefit to the City, a word of caution is advised. "Speedex" was not a substitute for quality design. We still needed to strive for logical and optimum key paths. If an item was truly unknown, "Speedex" could help you find it. For example, we directly added "Speedex" to both the Contractor's company name and to the Contractor's owner name several months after we had implemented the Contractor/Sub-contractor System. We discovered that Contractors frequently sent representatives to obtain permits who did not know the Contractor's registration number. This was not surprising as the City did not actually require the Contractors to carry the registration card for permit issuance. We also discovered that representatives did not know the actual registered company name (an alternate key). "Speedex" allowed us to circumvent these problems and located the proper records quickly. We justified the "Speedex" overhead in this case to provide better service to the Citizen. Since the Contractor data base was a relatively small one, the added "Speedex" data sets did not consume a great deal of space.

INTEGRATION OF THE SOFTWARE MODULES

As discussed above, we encountered, no major problems in tying the Street Index and the Contractor sub-system to the vendor's software. The integration of these three modules was controlled through standard calls to IMAGE.

We did encounter a problem in pulling the software together under a single menu. The vendor's software contained a "Main Menu" that was to coordinate all automated Code Enforcement processes. We had the menu set up to call the Speedware modules when requested. For example, if the user wished to register a new contractor in the Contractor/Sub-contractor (Speedware) system he would enter the proper option in the V/PLUS controlled menu and depress ENTER. The on-line COBOL program would then execute the proper call to Speedware and return with the desired Speedware sub-menu.

We struggled initially to make the call successful. We encountered a problem with bounds violations; the problem was solved when we determined the correct account capabilities and applied them as necessary. We encountered a second problem with Speedware loosing a temporary file designation. The problem was solved via resetting the file

designation in the specification file just prior to exiting Speedware to return to COBOL.

While we solved the problems in the COBOL/Speedware interface, we discovered the call required 15 - 25 seconds to complete. Over time, the users found the delay unacceptable. The timing problem was resolved when we added a Speedware "Main Menu" as the driver, and allowed Speedware to call COBOL. Although we never performed an in-depth analysis of why the Speedware to COBOL required less time, we conjectured that each time COBOL called Speedware, Speedware had to reinitialize its overhead. Apparently, if the system was designed with Speedware as the controller, the overhead would be executed a single time, with the initial call to Speedware.

Infocentre did not encourage the COBOL to Speedware interface. In fact, the Speedware manual contained no instructions on how to effect the call. The manual did however, contain instructions for a Speedware to COBOL interface.

BENEFITS TO THE ORGANIZATION

The primary benefit to the City was increased speed of implementation. As described above, we had calculated 245 days to complete a detailed design document and calculated an additional 645 days to code, test and document the software. We needed 3.4 man years to complete the project and we simply did not have the time.

Secondly, the approximate cost of in-house development was calculated at $159,330. The projected cost of a co-developed project was approximately $61,545. This figure included the actual salary costs for the Data Processing personnel during the implementation period. The City saved an estimated $97,785.

A third benefit to this type of arrangement, was reduced long term cost to the City for maintenance. While in-house maintenance required in-house staff, it did not require the monthly cash outlay to a third party vendor. While organizations have paid monthly maintenance fees to outside vendors for software usage, these fees do not negate the need to have staff on hand to support the software releases. In theory, support for software releases should require less time than in-house maintenance however, many programmer/analysts earn their living doing just that.

A fourth and perhaps most important benefit to the City was the increased flexibility this type of arrangement allowed.

As contractually stated, the City received the vendor's source code. The City did not have the right to sell the code, nor could it be given away under the purview of public domain. However, after a six month warranty period, the City had the right to make any modifications or enhancements that it deemed necessary.

ACES has by no means remained static. As of March 31, 1988, 17 months after implementation, ACES programming statistics have increased significantly. The City has added 1000 lines in on-line programming, yielding a total of 21,245 lines of on-line code. 38,332 lines of batch COBOL programming have been added, yielding a total of 77,599 lines of batch code. 29 Speedware modules have also been added, yielding a total of 81 Speedware modules.

The City has retained COBOL as the batch reporting language as the added programs handle data in much the same manner as the original 20 supplied by the vendor. A major sub-system supporting Zoning Enforcement was developed during this time period and accounts for most of the added Speedware modules.

A fifth benefit to the City in this type of arrangement was the sense of security that source code provided. The City received four bid proposals in response to the RFP. The high bid provided for a vendor to design and implement a customized Code Enforcement system. The next two bids were submitted by vendors that no longer exist under the same legal identity as they did when the bids were submitted. If the City had accepted either bid, a new contract would have been necessary. A renegotiated contract might not have been favorable to the City. The low and selected bid provided the City an excellent system as well as the source code for future growth and development.

Lastly, the City has derived great benefit from the Tax and Utility Billing "lookups". Few departments in the City do not use one or the other of these modules. As a result of their popularity, the City will be able to justify the cost of reformatting the Tax and Utility Billing keys to provide automatic access through a common key, namely street address. This conversion will eliminate rekeying of needed data. The conversion will also serve a second goal of preparing both Utility Billing and Tax for interface to the postal tape, allowing the City to implement zip plus four.

The reformatting of Tax addresses will be a step towards the City's goal of a parcel management system. Additionally, Code Enforcement has requested support for the storage of permanent parcel data in the forthcoming fiscal year. A

permanent parcel data set, combined with a reformatted Tax Master, will provide the City with a geo-base cornerstone.

BENEFITS TO THE VENDOR

While the benefits of co-development to the City entity were somewhat obvious, the vendor also derived benefit from this type of arrangement. The major benefit to the vendor was reduced development costs. The City purchased software that had been running in test mode at another site. The vendor programmed and unit tested the contracted enhancements. However the City, as the first live site, worked with the vendor to correct any problems and implemented the software into production.

A second benefit to the vendor was another happy customer. While Plano's situation was somewhat unique, I am sure we were not the first HP site that did not wish to enter into a long term maintenance agreement with an outside vendor. We have been very satisfied with our arrangement and when asked, we will provide an excellent vendor's reference.

Lastly, the vendor's product was enhanced considerably during the implementation process. On-line inspection requests, and automatic calculation of fees were missing from the system that was originally demonstrated. While the vendor programmed and tested these changes, the City had defined a need that would be of benefit to any governing body issuing building permits. The vendor was now free to market these enhancements.

IS IT FOR EVERYONE?

The long and short terms goals of the organization need to be evaluated in answering this question. If the contracting entity did not have a trained HP staff and had no intention of hiring such a staff, then this method of development would not be appropriate. However, if the entity wished flexibility to grow and to add to its systems in a logical and productive environment, then this method would be of obvious benefit.

In conjunction with the organization's goals, would it be willing to provide the programming and analysis time that was obviously required in this type of development? Approximately 13 man months of Data Processing support were required during the 13 month period. At one point during the project's development, four members of the Data Processing staff were dedicated to the implementation. This was a large commitment of resources, requiring total management support.

Secondly, integration of third party and in-house developed software assumed that a third party had the software available to meet a reasonable percentage of the entity's needs. If the software was not developed, would the vendor add the code to bring it to a satisfactory level? This was a tough question as it involved judgment and compromise so that a reasonable solution could be met. Resolving this issue could also cost money and a decision would have to be made as to who would pay the costs.

The third question that had to be answered was, could the entity find a vendor willing to engage in a joint development process? Part of the answer to this question goes back to my discussion on communication. Correspondingly, the vendor might have viewed this type of development as too expensive. In a large software development environment, the vendor must sell many systems to remain profitable. Many vendors consider customization too costly.

Lastly, could the contracting bodies resolve the legal issues arising in a co-development environment? Naturally, the vendor wished to protect his investment in systems development. He obviously wanted to retain the sales rights to his product. Also, would he be willing to relinquish the maintenance fees frequently associated with purchased software? On the other hand, could the contracting agency provide the assurance to the vendor that it would protect his rights as well?

IN SUMMARY ...

In writing this paper on the integration of third party and in-house developed software, I have merely described how this method of development worked in my organization. The success of this method is dependent on many factors, only a few of which have been noted. Each organization is unique, complete with its own set of idiosyncrasies. Suffice to say, that in the proper environment, co-development can be a cost effective and productive process.

ATTACHMENT 1
DETAIL DESIGN STRATEGY

MAJOR FILE MAINTENANCE AND REPORT PROGRAMS	RELATIVE DEGREE OF DIFFICULTY
MAIN MENU	H
APPLICATION DATA ENTRY VERIFICATION & UPDATE	H
PLANS EXAMINING ENTRY VERIFICATION & UPDATE	H
APPLICATION INQUIRY	H
PERMIT DATA ENTRY VERIFICATION & UPDATE	H
PERMIT INQ	H
CONTRACTOR PERMIT INQ	H
CONTRACTOR INQ	M
INSPECTION REQUEST	M
INSPECTION REQUEST PRINT & RPT	M
INSPECTION POSTING	M
DAILY REPORT OF INSPECTIONS MADE	M
PERMIT FINALIZATION	H
OTHER FILE MAINTENANCE AND INQUIRY PROGRAMS	
FEE SCHEDULE FILE MAINT & INQ	H
PERMIT TYPE FILE MAINT & INQ	M
CLASS OF WRK FILE MAINT & INQUIRY	M
TYPE USE FILE MAINT & INQ	M
INSPECTION TYPE FILE MAINT & INQ	M
PERMIT STATUS FILE MAINT & INQ	M
PLANO STREET INDEX FILE MAINT & INQ	H
INSPECTOR NUMBER/NAME FILE MAINT & INQ	M
OFFICE PERSONNEL NUMBER/NAME FILE MAINT & INQ	M
INSPECTION STATUS FILE MAINT & INQ	M
OTHER FILE MAINTENANCE REPORT PROGRAMS	
FEE SCHEDULE RPT	S
PERMIT TYPE RPT	S
CLASS OF WORK RPT	S
TYPE USE RPT	S
INSPECTION TYPE RPT	S
PERMIT STATUS RPT	S
PLANO STREET INDEX RPT	S
INSPECTION NUMBER/NAME RPT	S
OFFICE PERSONNEL NUMBER/NAME RPT	S
INSPECTION STATUS RPT	S

OPERATIONAL REPORT PROGRAM

DAILY PERMIT DETAIL RPT	M
WEEKLY PERMIT DETAIL	M
MONTHLY RPT OF INACTIVE & FINALED PERMITS	M
ANNUAL REPORT OF ARCHIVED PERMITS RPT	H

MANAGEMENT REPORT PROGRAMS

MONTHLY PERMIT ACTIVITY RPT	M
MONTHLY CENSUS RPT	M
DAILY RPT OF INSPECTIONS MADE	M
MONTHLY SUMMARY OF INSPECTIONS MADE BY INSPECTOR	M
MONTHLY SUMMARY OF INSPECTIONS MADE BY DISTRICT	M
MONTHLY RPT OF COMMERCIAL NEW CONSTRUCTION & APARTMENTS	M
MONTHLY RPT OF APPLICATIONS SUBMITTED W/O AN ADDRESS	M

TOTAL # OF "H" (HARD) = 11 X 10	110	
TOTAL # OF "M" (MEDIUM) = 23 X 5	115	
TOTAL # OF "S" (SIMPLE) = 10 X 2	20	

	245	TOTAL NUMBER OF DAYS FOR DETAIL DESIGN

ATTACHMENT 2

SYSTEM: CITY OF PLANO PROGRAMMING ESTIMATES

FNCT Y/N	EASY TO DO Y/N	TIME REQ. DAYS	OLD PGMS #	NEW PGMS #	LANG USED C/S	** MAJOR FILE MAINTENANCE & REPORT PROGRAMS **
N	N	15	-	1	C	MAIN MENU
N	N	30	-	1	C	APPLICATION DATA ENTRY VERIF/UPD
N	N	30	-	1	C	PLANS EXAMINING ENTRY VERIF/UPD
N	N	15	-	1	C	APPLICATION INQ
N	N	30	-	1	C	PERMIT DATA ENTRY VERIF/UPD
N	N	15	-	1	C	PERMIT INQ
N	N	15	-	1	C	CONTRACTOR PERMIT INQ
N	N	10	-	1	C	CONTRACTOR INQ
N	N	10	-	1	C	INSPECTION REQUEST
N	N	10	-	1	C	INSPECTION REQ PRINT/RPT
N	N	10	-	1	C	INSPECTION POSTING
N	N	10	-	1	C	DAILY RPT OF INSPECTIONS MADE
N	N	15	-	1	C	PERMIT FINALIZATION
==========			**OTHER FILE MAINT AND INQUIRY PGMS.**			
N	N	30	-	1	C	FEE SCHEDULE FILE MNT & INQ
N	Y	5	-	1	S	PERMIT TYPE FILE MNT & INQ
N	Y	5	-	1	S	CLASS OF WORK FILE MNT & INQ
N	Y	5	-	1	S	TYPE USE FILE MNT & INQ
N	Y	5	-	1	S	INSPECTION TYPE FILE MNT/INQ
N	Y	5	-	1	S	PERMIT STATUS FILE MNT/INQ

FNCT Y/N	EASY TO DO Y/N	TIME REQ. DAYS	OLD PGMS #	NEW PGMS #	LANG USED C/S	** OTHER FILE MAINTENANCE REPORT PROGRAMS **
N	Y	15	-	1	S	PLANO STREET INDEX FILE MNT/INQ
N	Y	5	-	1	S	INSPECTOR NAME/NO FILE MNT/INQ
N	Y	5	-	1	S	OFC. PERSNEL NAME/NO. FILE MNT/INQ
N	Y	5	-	1	S	INSPECTION STATUS FILE MNT/INQ
N	N	15	-	1	C	FEE SCHEDULE REPORT
N	Y	5	-	1	S	PERMIT TYPE REPORT
N	Y	5	-	1	S	CLASS OF WORK REPORT
N	Y	5	-	1	S	TYPE USE REPORT
N	Y	5	-	1	S	INSPECTION TYPE REPORT
N	Y	5	-	1	S	PERMIT STATUS REPORT
N	Y	30	-	5	S	PLANO STREET INDEX
N	Y	5	-	1	S	INSPECTION NAME/NO. REPORT
N	Y	5	-	1	S	OFFICE PERSONNEL NAME/NO. REPORT
N	Y	5	-	1	S	INSPECTION STATUS REPORT
==========		**OPERATIONAL REPORT PGM.**				
N	N	15	-	1	C	DAILY PERMIT DETAIL REPORT
N	N	15	-	1	C	WEEKLY PERMIT DETAIL PERMIT
N	N	15	-	1	C	MTHLY RPT OF INACTIVE/FINAL PRMTS
N	N	15	-	1	C	ANNUAL RPT OF ARCHIVED PRMTS RPT

FNCT Y/N	EASY TO DO Y/N	TIME REQ. DAYS	OLD PGMS #	NEW PGMS #	LANG USED C/S	** OTHER FILE MAINTENANCE REPORT PROGRAMS **
==========		**MANAGEMENT REPORT PGMS.**				
N	Y	5	-	1	S	MTHLY PERMIT ACTIVITY REPT
N	Y	5	-	1	S	MTHLY CENSUS REPT
N	Y	5	-	1	S	DAILY REPT OF INSPECTIONS MADE
N	Y	5	-	1	S	MTHLY SUMRY OF INSPS MADE BY INSP
N	Y	5	-	1	S	MTHLY SUMRY OF INSPS MADE BY DIST
N	Y	5	-	1	S	MTHLY RPT OF CMRCL NEW CONST & APTS
N	Y	5	-	1	S	MTHLY RPT OF APPS SUBMITTED W/O ADDRESSES

NUMBER OF DAYS NEEDED (AT 6 EFFECTIVE HRS PER DAY): 640

PROGRAM COUNT:

NEW COBOL PROGRAMS TO WRITE: 19

NEW SPEEDWARE PROGRAMS TO WRITE: 29
(THESE ARE LOGICAL, NOT PHYSICAL PROGRAMS)

ATTACHMENT 3

RECAP OF ALTERNATIVES

OPTION	TIME MAN YEARS	TOTAL COST
VENDOR A	N/A *	$293,205
IN-HOUSE DEVELOPMENT	3.4	$159,330
PUBLIC DOMAIN A	3.1	$150,870
PUBLIC DOMAIN B	3.0	$145,230
PUBLIC DOMAIN C	2.6	$139,617
VENDOR B	N/A	$116,870
VENDOR C	N/A	$ 68,045
SELECTED VENDOR	N/A	$ 61,545

* implementation times were not calculated for vendor options as times would vary by contract.