I. Introduction

Would a bus driver spend his vacation on a bus? Probably not. As programmers and systems analysts we spend a great deal of time and effort designing computer systems that automate the tasks performed by our clients. But, like a bus driver who would not consider taking a bus trip for a vacation, we have not been nearly so successful in applying automation to our own efforts, namely, the systems development process.

Evidence of this lack of success is not difficult to uncover. Despite many advances in software technology, the applications backlog at most MIS shops remains largely intact, stretched two or three years beyond available resources. Industry requirements for entry level programmers is expected to be one of the brightest growth areas for employment into the 1990's.

Yet, effective methods for addressing this situation have already been proposed. The use of database management systems and fourth generation languages reduces the time required to develop new applications. Improved system development methodologies have been defined which improve the software design process. Data dictionaries and logical design techniques are available to automate the data design and data administration functions. Programmer productivity tools such as screen painters, code generators, diagnostic aids, and source management libraries are available commercially. Project management software can be used to estimate resource requirements and track milestones. And now fifth generation expert systems are planned which will assist with the analysis, specification, and design phases of a project.

Therefore, it seems that lack of automation in systems development is not due to the lack of available remedies, rather the fact that the tools have not been widely or successfully used.

II. Why automation of systems development is difficult.

Applications development managers who choose to introduce automation into the systems development process do so with a certain amount of trepidation. There are several risks involved.

First, there is the highly variable nature of the process itself. There are nearly as many ways to develop systems as there are ways to skin a cat. Design methodologies vary from data-based methods to process-based approaches and various flavors in between. Programming efforts range from egalitarian "egoless programming" teams, to the "super-programmer with support approach. Process specification may involve a top-down structured method or results-oriented prototyping efforts. It is difficult to imagine an automated method that would support all of these approaches at once. Each new system presents different problems to the developer, some of which may be better served by one approach over another. If automation limits the development process to a single methodology it may lead to suboptimal development of certain classes of systems.

Second, there is the creative nature of the systems development process. I'm sure that many of you have known instances where the unplanned or serendipitous solution to a given problem proved to be the most elegant. Automated methods which are not flexible enough to allow this creativity to occur will lead to inefficient systems as well as unhappy systems developers.

Third, systems development methodologies are often provincial in nature to account for local standards and procedures, and the idiosyncrasies of systems development for a particular business environment. Automated methods which cannot be extended to accommodate local requirements may be politically unsalable.

Fourth, systems development teams are often heterogeneous in terms of the skills inventory and backgrounds of the participants. Automated methods which require significant amounts of training may cost more in terms of project overhead than they save.

Finally, automated systems development tools are often quite expensive, ranging from thousands of dollars for debugging tools to hundreds of thousands of dollars for expert systems. Any mistake made in product selection is likely to be an expensive one.
III. SAS® productivity tools at Midcon Corporation.

The Houston MIS department, Midcon Corporation, has been progressive in the application of advanced techniques to support corporate data processing requirements. They have successfully employed fourth generation languages, dictionary-based data administration, and relational data base technology in the development of accounting and operational systems. In an effort to make their systems development process more effective and efficient, they have experimented with the automation of phases of the process, with special emphasis on analysis and design tasks.

They decided to begin on a small scale, developing customized techniques on a project-by-project basis. In this manner they hope to learn more about the utility of various techniques, and their application to a specific corporate environment.

The methods and tools developed were left to the discretion of each project leader. Each project team developed customized reports to support their own requirements, with the constraint that the productivity efforts should not consume more resources than they saved from normal project overhead.

The base SAS system, augmented by the SAS/FSP® product has been used to develop the productivity software for all projects to date. There are several reasons why the SAS system is particularly suitable to this application.

The flexibility of the SAS system allowed them to build their project data bases and develop productivity tools in an ad hoc manner. Each project team started with a "minimal set" of project data and expanded on it as the project progressed and new requirements arose. This flexibility minimized constraints on creativity by the system developers.

Productivity tools lose most of their efficiency if they require generation of paper documents by the systems development staff. Therefore it is very important that productivity software operates online. The SAS/FSP system provided this capability very effectively. Most of the project data was updated using the default screens provided the SAS/FSP product.

The SAS system interfaces with many other software products. The interfaces between SAS and Midcon's data base management system allowed them to transport metadata from the data dictionary, directly into the project data bases, for use in other system development tasks.

The productivity tools developed with the SAS system diverted few resources from the main software development effort. The SAS system was already in place, so no new products needed to be purchased. The applications were developed with only a few man-days of effort, and no personnel training was required to use them.

IV. Case study 1: Time-constrained project.

The first project to use the SAS-based tools required the development of a data entry, update and control system for legal and operational information captured from transportation contracts. The project required development of two online data entry systems, one control and approval system, and an online inquiry system for the collected data. The level of effort involved development of about 60 screens, using a peak manpower loading of 2.5 analyst/programmers, and was constrained by a short three-month time frame.

After an initial list of data elements was developed, a SAS data set was built to collect additional information about each data item. Initial user interviews consisted of a SAS/FSP session where the analyst queried the user for a definition of each item, the source of the information in the business environment, and syntax rules such as field type, total field length, preferred display format, and encoding mechanisms. The use of online data entry screens for the interview process added structure to the metadata gathering process and reduced the effort necessary to collect the information.

The metadata from several user interviews was consolidated in SAS-generated reports to facilitate review by a larger user community, including Management.

As part of the logical design process, data elements were associated into logical records via standard third normal form analysis. A second SAS data set was developed to contain record descriptions that mapped the record to business entities. The original data element data set was expanded to collect information about element-to-record associations and logical key information. A second set of SAS
reports was written to display the logical record layouts as they were developed. These reports were used as a basis for logical design reviews with users and the data administration staff.

As part of the physical data base design process, logical records were assigned to physical files. This file information was stored in the records data set. SAS programs were written to generate the Data Definition Language statements necessary to define each physical file. These programs were used repeatedly as each proposed physical design was tested.

In the systems analysis phase of the project, design of the data-editing software proved to be very complex. Default values, existence rules, and even syntax validation for a data field varied from one source document to the next. To simplify the design, the SAS data set was used to collect information about default values, existence rules and syntax validation for each data field and each source document. Export software was written to transfer this information to a database file. Data-driven editing routines were written to edit each data value, based on the pre-stored edit rules, and the type of input document.

V. Case Study 2: Major development effort.

The SAS-based tools have also been applied in a larger software development effort. This project required the development of several online systems involving more than 100 screens, and 60 batch processes. This project had a peak manpower load of 10 programmer/analysts and a project duration exceeding one year.

This project developed software which synthesized and reprocessed information collected by other systems. In this case a SAS data set containing data element information was created by downloading information from the corporate data dictionary via the SAS interface to the Data Base Management System.

The logical design process generated record layout reports similar to those used in Case 1. However, because of the larger size of the project team, it was also necessary to generate change control reports which tracked changes from one version of the logical design to the next. In this way, each member of the programming team was aware of any changes that might affect the programming effort in his own area.

Another change control report compared the logical design to the existing corporate data dictionary. This made it easier for the data administration staff to evaluate design changes and their impact on the corporate data base.

The physical design process used the code generation software developed in Case 1. In order to improve communication within the development staff, each physical design prototype was date/time stamped to synchronize logical and physical design changes.

In a project of this duration, analysis of the impact that software design changes will have on the system developed to date become an important issue. To assist in the impact analysis effort, the SAS data sets were augmented with information about the relationships between files, records, and the screens where they were displayed or updated. Impact analysis reports could then be generated from SAS reports.

VI. Conclusion.

It is interesting to note that the most useful tools in Case 1, the ability to interactively collect metadata from user interviews, and the collection of edit rules for data validation were not used in Case 2. On the other hand the change control reports so vital to the large staff in Case 2 would have been a waste of time for the two-person staff in Case 1. The flexibility to customize the productivity tools to fit each situation was a very important feature of the SAS-based approach.

The SAS system proved to be a low cost, effective tool for the development of applications to automate the systems development process. The flexibility of the product allowed Midcon's project teams to experiment with various techniques for improving the efficiency of the systems development process. The knowledge gained will assist them in the evaluation of commercial products to support this function as well as in-house development of more sophisticated tools.

SAS and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA