Getting Started in Data Capture

Bruce Toback
Infotek Systems

I. Introduction

Over the past decade, on-line computer usage has been replacing traditional batch usage at an increasing rate. The reasons for this transformation are many, but most center around responsiveness: the ability to use the computer as an information source in real time, asking questions which require the most current answers.

In point of fact, however, a large portion of this transformation has been the replacement of batch systems with "faster" batch systems. Data are still collected and recorded manually, then sent to a terminal operator for entry into the "on-line" system. This method has advantages over traditional batch systems in that data are made available sooner than with conventional batch processing and inquiry is easier, but responsiveness can still suffer. An additional consideration is the accuracy of information presented to the computer. In both traditional batch systems and "on-line" batch systems, information is copied at least twice and perhaps several times on its way to the computer. Each translation increases the chance for error.

More recently, source data capture has been introduced. The idea of source data capture is not new, of course: every mechanical cash register is a source data capture facility, recording sales transaction information as each transaction occurs. Using source data capture as input to a computer, however, is a relatively recent innovation. Data capture has been in use quite extensively in the retail sales field in the form of point-of-sale terminals and is now being used in manufacturing as well.

This paper is intended as an introduction to data capture. It will serve as a guide during a data capture project definition, and provide a list of ideas and issues to focus on as you design your data capture system.

II. When data capture is used

In general, data capture is useful in a manufacturing environment when at least one of the following is true:

- The information needed is highly volatile (i.e., changes frequently)
- The information needed is too voluminous to be recorded and keypunched in a timely manner
- The information required is already available in machine readable form as a by-product of an existing manufacturing operation.

Note that in general, source data capture means only that information is recorded electronically as soon as it is generated. This may mean that electronically-assisted means exist to capture the requrired information or may simply mean that the person creating the information (e.g., the sales clerk or stock clerk) enters the information by hand into a suitable terminal.

Let us examine each of these criteria in detail. First, information may be considered volatile when its lifetime is shorter than the turnaround time of the system used to record it. For example, on a fast-moving shop floor,

a given assembly may move in processing from operation to operation several times in one day. If information is entered and recorded once a day, the report will be accurate only for those assemblies that are stuck. This kind of reporting is usually of limited value.

Information which is too voluminous to be keypunched normally can include such things as point-of-sale inventory transactions, e.g., grocery checkout; kit pull information when lot tracking is in use; and of course, job tracking information on a busy shop floor. In general, the volatility of information is related to its volume: information which consists of only a few data items can usually be entered into a data processing system in time to be reported before the end of its useful life.

Finally, information which may already be available in machine readable form includes such things as, again, grocery checkout, where UPC or EAN codes are pre-stamped on packages by the manufacturer; electronic time clocks which can record time and attendance information at the same time that an employee's time card is being punched; and electronic test equipment which may be computer controlled. In some cases, as in retail checkout, the machine-readable coding may be inherent in the operation being tracked by the computer. In other cases, such coding may be added at neglegible cost, such as when turnaround documents are already produced by computer.

III. Defining your Data Capture Needs

Once you have determined that your data entry needs can be met by a data capture system, your next task will be to design the system. The design can be broken down into three main concerns:

- o Volume capabilities
- o Volatility problems
- o Using preexisting data

Volume Capabilities

In order to design a data capture system, an upper limit on transaction volume must be determined. How this is done depends on the type of data that you intend to capture, and on what capabilities already exist for capturing it. When converting from an "on-line batch" system, the best indicator may be the number of turn-around documents currently being keyed into the system, multiplied by the number of transactions recorded on each document. This last is very important since in a properly designed data capture system, each "real" transaction (e.g., item sold, job move, etc.) should equate to one computer transaction.

If no automated data processing facility is currently in use to record the information you are trying to capture, then the volume should be estimated by the personnel most involved with the data.

Number of terminals Once you have determined the total transaction volume for each function you wish to perform, you will need to find the number of terminals required to support that volume. The number of terminals required depends both on the total transaction volume and the number of types of transactions.

Transaction types are important becase each function may take a different amount of time to perform, and more importantly, may take place at a different location from other functions. In general, enough terminals to support the volume for a particular function should be present in the immediate area of transaction generation. This is self-evident when machine-readable information is input through the terminals, but may not be so evident when operators are required to key in information. If an operator needs to "go over to" a terminal to make an input, the tendency will be for the operator to batch input and do several transactions at one time. This may result in both translation problems as operators write down transactions for later entry, or in timing problems as operators delay entering important information until a break period.

For hand data entry, approximately five seconds are required for a 10-digit numeric entry. This may be reduced substantially by using a machine-readable code such as bar code or OCR.

Machine-readable input Machine-readable input can have a significant effect on the volume capabilities and terminal requirements of your data capture system. First, of course, data entry speed can be increased dramatically because

operator keying is no longer required. Bar code or OCR scanning can reduce the operation of entering a 10-digit work order number to a single wand stroke, with almost no possibility of keying errors. However, a "one-operator, one-station" arrangement becomes almost mandatory, since sharing a "personal" device such as a bar code wand is generally reported as difficult. If information is coming from another computer, as in the case of automatic test result reporting, the system must be by definition one-operator, one station.

Volatility Problems

If the data you are trying to capture is highly volatile, you will need to consider several additional factors. Most of these are treated as application design problems, but will need to be considered during your system definition.

System availability Availability of the computer system is of interest in even a batch system, but is much more important in an on-line system. Availability is critical to the success of a data capture system dealing with volatile data. Imagine for a moment the chaos that might result from a malfunction of a centralized electronic door lock control! The data in this case are extremely volatile - someone wants in (or out) **now** - and it is gone forever if it is not captured at the instant it is generated. If your data are this volatile, you should make provision for spare or backup terminals, and for rapid switchover to a backup computer if practical.

In a situation involving volatile data, terminal reliability is as important as central system reliability. If an operator loses access to the system even though the system is still operational, data from that operator will be batched or lost, or the operator will simply not do any work until access is restored.

Many of the schemes for connecting data capture terminals to the HP3000 involve shared equipment, leading to the possibility of single-point system failure. For example, if all terminals are on a single multipoint line, failure of the INP or SSLC, of the factory data link adaptor (3074A), the cable itself, or certain terminal failures can render the entire data capture network inoperable. Suitable backup precautions must be taken, e.g., splitting the network into two or more lines, or connecting critical terminals point-to-point so that failure of an ATP, ATC, or ADCC can be worked around rapidly by moving the terminal to a port on a different ATP, ATC, or ADCC. The net effect of these considerations is that you should consider the possibility of using redundant hardware in situations in which data are extremely volatile.

Data Recovery In addition to high data collection equipment uptime, volatile data collection requires that you consider recovery methodologies in the event of system failures. These can take several forms. Assuming that retention of paper backup is undesirable, or that no paper backup exists to begin with, the most obvious recovery technique is user logging. This service, provided at an elementary level by MPE and at a more sophisticated level by IMAGE, causes MPE to attempt to journalize every file update to tape. The implication, however, is that you are willing to dedicate a serial medium, either tape, disc, or cartridge tape, to data logging whenever the data capture system is active. You should plan for this in determining your hardware requirements.

In addition to hardware requirements for journalization, you will need to examine your backup strategy for compatibility with logging. In particular, if you are using IMAGE, you **should** use DBSTORE instead of STORE or SYSDUMP to back up. This is because the Hewlett-Packard logging/recovery utility, DBRECOV, will examine the last **DBSTORE** date to determine whether your transaction log and data base "match" for recovery. If you normally do long-period incremental dumps (i.e., every two or three days), you may in addition be faced with a considerable recovery period after a system crash. While this is preferable to losing several days' worth of transactions (!), it also causes a **net** decrease in system availability as perceived by your data capture operators.

Finally, if you are not using IMAGE, you will need to design your own recovery utilities for your data base. If you implement your data base with KSAM or with a third-party data base system, this may be an important planning consideration.

Machine-readable documents If you choose to use paper backup instead of transaction logging, you should seriously consider the use of machine-readable turnaround documents. This will facilitate the recovery process in the event of a system failure, since a large amount of keypunching will not need to be redone. Rekeying already-entered data can be very destructive to the morale of an operator to whom computer input may be an unrewarding burden in any case (see User Training, below).

Using Machine-Readable Data

If you have chosen to use data capture to take advantage of existing machine-readable data, you will need to consider exactly how you will read the machine-readable information.

Bar coding Bar coding is becoming extremely popular in manufacturing as well as retail environments. It is easy and inexpensive to generate (and may in fact already exist, as in the case of the UPC/EAN codes in the retail field), easily read either under manual control or automatically by non-contact scanners, and is very reliable. Bar code readers are made by a large number of manufacturers including (in the United States) Interface Mechanisms (Intermec), Hewlett-Packard, Burr-Brown, Epic Data, and others. Some of these companies in addition manufcture complete data capture subsystems for inclusion on an existing computer system.

Bar codes may or may not be human-readable in addition to being machine-readable. Some bar code standards (UPC and EAN, for example) stipulate that the code shall have a human-readable translation above, under, or beside the code itself; you can, of course, design your system to provide the translation.

Optical Character Recognition Less popular than bar code is a system called Optical Character Recognition. Specially printed labels containing ordinary numerals (and sometimes letters as well) are read by a hand-held contact scanner. While the resulting label is smaller than a bar code label since the optical characters are typically more dense do not need translation, read reliability is rather poor with most systems, and several tries may be needed to read a label. OCR systems are usually manufactured as turnkey add-ons to existing computer systems. Some manufacturers of point-of-sale terminals offer OCR as an option.

Automated Test Equipment Automated test equipment systems (ATE) are becoming more popular as prices of versatile systems have dropped dramatically. By combining an ATE system with bar code or some other means of identifying assemblies, a completely automated testing and inspection records system may be set up. Interface to the HP3000 may be directly through RS-232C or RS-422 (terminal interfaces), or through GPIB or ordinary parallel interfaces using the HP3078 data collection terminal.

IV. Design and Implementation

System design aspects and their corresponding implementation aspects may be broken down into four areas:

- o Topology
- o User Interface
- o Application System
- o User Training

Topology

The network topology that you choose to implement depends on your needs as discussed earlier. The key questions are:

- o Where is the data available?
- o What form does it take?
- o Who has the data?
- o How will the data be communicated?

Where? On the principle that terminals should be located as close as possible to the source of the information that will be entered through them, you will need to locate as precisely as possible the locations in your plant which generate the information that your system is intended to capture. While this seems self-evident, all too often a location is denied a terminal because the usage would be low, even though the information generated there is as volatile as at higher volume locations. The result, of course, is "batching" of input and concomitant data timing problems.

Data capture terminals can usually be placed in crowded areas by using wall mounting. Hewlett-Packard's 3076A terminal is especially designed for permanent wall mounting; its keyboard and display may be used easily while sitting or standing. Wall mounting is also convenient when information will be entered by operators while walking or driving, as in access control or timekeeping applications.

What? The data to be entered can take many different forms. In many instances, data starts simply as knowledge: the receiving clerk knows that something has come in from a vendor, for example. In other cases, the data is part of an existing document, e.g., the packing slip contains a packing slip number that must be entered. If the data consists simply of knowledge, it is usually most expedient to simply key information into a convenient terminal. If data are contained on a preexisting document, it may be possible to code the document for machine reading. This is especially true if the preexisting document is generated within the company: the document can be generated on a printer capable of printing bar code to be read in later.

If the data to be captured exists in another computer (an ATE system, for example), no conversion or keying will be necessary.

Who? The data to be captured should be entered by the person (or object) most closely associated with it. In terms of the organization, the person or political entity that is **held responsible** for the accuracy, completeness, and timeliness of the tracked activity should have the responsibility for data entry. This means that terminals will be located close to the **people** handling the data: in the stockroom, on the shop floor, at the receiving dock, and so on. Where several people have knowledge of a particular physical event, the person who should be responsible for data entry should be the one who **must** have the knowledge for the business to run smoothly.

As an example, consider the shop floor controller moving assemblies on a shop floor. During normal operations, two people have the knowledge that an assembly has been moved to a new workstation. The shop floor controller, of course, is aware of it, but the because he or she caused the physical event. However, the assembly must be "formally" entered in the queue at the new workcenter. Thus, the workcenter's supervisor or lead operator, if any, should be responsible for data entry: the data capture system will then show the event only after action can be taken on it.

In some cases information will be subject to audit, and it may be desirable to limit update access. This may be accomplished with a machine-readable ID such as a magnetic striped badge or laminated bar code. In such cases, it may be expedient to use a method of physical identification similar to that which is used for automated data entry.

How? Once an event has taken place, is discerned to have done so by somebody, and a record of the event has been made and entered, the record must be transmitted to the computer. This is done either directly by the terminal, or by a data capture subsystem to which the terminal is connected. Hewlett-Packard, in general, favors the former approach; most other data capture vendors favor the latter. In practice, both may be used for different parts of the same data capture system.

In most installations, the data capture terminals will be located in the same building, or at least in close proximity to the computer acting as host. The discussions which follow assume that this is the case. If your situation is different, you will need to explore data communications options, but a full discussion of data communications is beyond the scope of this paper. The principles which will be discussed, however, are applicable to both remote and local data capture terminal networks.

For Hewlett-Packard data capture terminals, two options are available: multipoint connection, and point-to-point connection. Which of these options you choose should depend on:

- o Transaction volume
- o Volatility (i.e., minimum required reliability)
- o Proximity to the computer site
- o Cost
- o Flexibility

The first of these, and the one usually recommended by Hewlett-Packard, is multipoint. Hewlett-Packard offers two different hardware implementations of this software protocol.

Standard multipoint requires that all data capture terminals be connected in a "string," i.e., "daisy-chained" together. Cabling this arrangement over a large factory may be somewhat awkward, and the arrangement tends to be inflexible. Adding a terminal generally requires extensive additional cabling and in most cases will require that the data capture system be shut down during the installation. In addition, the method is limited in the distance that it can cover, and its immunity to electrical interference is somewhat limited.

The immunity of ordinary multipoint to single-point failure is limited by the length of the string. If a terminal fails, it will usually prevent operation of all terminals beyond it in the string.

The Factory Data Link, or multidrop, is an arrangement wherein many terminals share a common "supply" line, rather similar to houses along a water main. Physically, the data link consists of a fairly thick cable which may be initially installed as a loop around the inside of a building. The data link is limited to a maximum of five miles in length or less depending on the locations of terminals relative to the computer site. Data capture terminals are "tapped" into this line in much the same way that sprinklers are "tapped" into a garden hose. Each "tapped" terminal may have more terminals daisy-chained onto it as ordinary multipoint terminals. The method is very flexible, since installing a new terminal involves only tapping into the line at the point closest to the site of the new terminal, or daisy-chaining off an existing terminal if a second terminal is to be added to an existing data capture site. Adding a

terminal disables only part of the system, and that only for a few minutes. Removing a terminal from the factory data link will usually not cause any downtime. Noise immunity characteristics of the factory data link are excellent; the data link will often function in environments with such high levels of radio frequency interference that no other connection method will permit error-free operation.

Immunity to terminal failure is very poor, however. Failure of a single data capture terminal can potentially "hang" all other terminals sharing the Factory Data Link.

Both multipoint connection methods, the daisy-chain and the Factory Data Link, are quite susceptible to single-point failure since many terminals share, at a minimum, a common computer interface and communication line. Failure of either of these components will cause the failure of all terminals in a network. For this reason, it is usually best to split a network into two or more "subnetworks" if a multipoint protocol is to be used.

If transaction volume is very high and the amounts of data transferred per transaction is large, performance can suffer. Because only one terminal can be using the common line at any given time, one very busy terminal can use most of the communication capability available. This must also be considered when designing the data capture network topology. But because of this facility sharing, the communication cost per terminal is quite low.

Point-to-point connection of data capture terminals has few of the disadvantages of multipoint connection. Immunity to terminal failure is excellent, since all common elements in the communication facility are thoroughly isolated from the terminal itself. Failure within the communication interface may easily be remedied by switching plugs at the computer site, an option not easily available with multipoint connection.

The chief disadvantages of point-to-point connection are the need to run separate cables for each terminal to be connected, resulting in decreased flexibility and significantly increased cost, and relatively low immunity to electrical noise.

Designing the User Interface

One of the most significant roadblocks in implementing a data capture system, and one of the least quantifiable, is simply getting people to use the system. One of the key considerations in this is the "user interface" - that part of the system which the user sees.

An almost universal characteristic of a data capture system is that the operators doing the most input usually have the least to gain from having the input done. This is in sharp contrast to batch systems or "on-line batch" systems in which the operators primary responsibility is to enter data. For the operator of the data capture system, then, maintaining the system is a burden rather than a part of the job.

By properly designing the user interface, this burden can be made as small as possible, and some element of reward can be introduced to make using the system more palatable. In general, the interface designer should be concerned with simplicity and feedback.

Simplicity Most data capture transactions will involve several steps, possibly implemented with a question-and-answer dialogue. If possible, this dialogue should be designed out of the user interface through the use of machine-readable documents. For example, consider the dialogue which might be required to move an assembly from one workstation to another. The important elements of the transaction might be the work order number on which the assembly is being built, the workcenter being moved from, the workcenter being moved to, and the quantity being moved. The system should generate all ancillary information such as the date and time of the move. In a dialogue, the transaction might be:

```
Computer:  W/O#?
User:        4158245 <enter>
Computer:  FROM OPN?
User:        45  <enter>
Computer:  TO OPN?
User:        50  <enter>
Computer:  QUANTITY?
User:        100 <enter>
```

This transaction requires 18 keystrokes and requires waiting for computer response after each of four entries. (These count a single initial keystroke for beginning the transaction.)

The transaction can be shortened somewhat by using defaults:

```
Computer:  W/O#?
User:        4158245 <enter>
Computer:  FROM OPN 45?
User:        <enter>
Computer:  TO OPN 50?
User:        <enter>
Computer:  QUANTITY 100?
User:        <enter>
```

This example requires only 12 keystrokes, since the application tries to anticipate the user's responses by checking existing information about the work order.

The transaction can be reduced still further by using a computer-generated work order form with bar codes pre-printed:

```
User:      (Wands composite FROM)
Computer:  W/O# 4158245: 45 TO -?
User:      (Wands composite TO)
Computer:  QUANTITY 100?
User:        <enter>
```

This example requires only one keystroke and two bar code reading operations. The "composite" is a bar code digit string containing the work order number, the operation number, and a transaction code meaning "move assembly." The operation code is ignored during input from the TO? query, e.g.:

041582450045

Thus, by careful choice of user interface, 18 keystrokes and four waiting periods have been reduced to three keystrokes or user actions, and only two waiting periods. These waiting periods can be made quite short with proper application program design.

Feedback Because the primary responsibility of the data capture operator is typically something besides data entry, it is important that he or she receive some kind of feedback or reward in return for entering the data. This will usually consist of an acknowledging beep or message, but should include some additional information when possible. This may be difficult to achieve, but might come in the form of a count of units of production, e.g., number of units tested today or total from this lot. This serves to give the operator a sense that something inside the terminal is listening. Giving the operator a printed summary of the previous day's input can serve the same purpose. In general, the important point will be to insure that the operator realizes that some concern exists that the data entry part of the job gets done as well.

The amount of feedback should be proportional to the amount of work required for keying in. If the operator is simply required to run a bar code wand over a tag on the assembly, or to drop a card into a slot, little feedback will be required. If some dialogue is required, as in the above examples, more feedback should be provided.

Designing the Application System

Designing application programs for a data capture system requires attention to some aspects of programming that are not normally given much weight in ordinary batch or on-line processing. In particular, the programmer for a data capture system must pay attention to:

- High transaction volume
- High reliability and fail-soft operation
- High volume reporting

High transaction volume

When the number of transactions to be processed is very high, special attention must be given to such aspects of application design as file locking, transaction selection methodology, and general response time optimization. Because the key to all a successful data capture operation is a good user interface, and because a good user interface will derail the operator's primary activity for as short a time as possible, every effort must be made to shorten response time. An execllent paper on this subject was presented at the North American meeting of the HPIUG in 1983[1]. This paper covered various aspects of application design relative to response time and overall performance.

Using IMAGE

Of considerable importance to maintaining adequate transaction volume is the use of record-level locking when using IMAGE. Because many operators are likely to be accessing a fairly small group of high-use records, attention should be given to minimizing locking conflicts; locking at the lowest possible level is therefore a necessity. In addition, PUTs to detail data sets should involve primarily data sets with few paths. Sorted chains, unless used for maintaining chronological sequences, should be avoided. In addition, if several records on a chain must be manipulated, as in the case of the work-in-process tracking example above, the records should be manipulated in core to minimize the number of physical I/O's required to finish the transaction. Programmers should be willing to sacrifice simplicity for performance in this response-time critical situation.

Menu Selection

Many traditional menu selection schemes rely on process handling, creating and launching a new process for each transaction, or group of similar transactions. The overhead of this operation is unacceptable in for high transaction volumes and should be avoided. Other methods, such as using the menu (function selection keys on the terminal) to selectively call procedures in a single large program file, will improve response time and decrease I/O and memory resource usage. Since program code is shared among all users of a particular program, considerable simplicity of design can be achieved along with a performance increase by using this method. Common data base access

routines should be placed in a single segment and shared among all transaction executors.

One helpful system design method is to avoid hard-coding menu parameters in your application program. If the program is designed to configure the function-to-key correspondence from a file, it becomes possible to provide each terminal in the data capture system with a subset of functions from a single large library. This feature becomes increasingly useful as your organization gains experience using the data capture system, since functions can easily be moved from terminal site to terminal site as the initial expectations inherent in the original design are modified.

Reliability

More than any other computer application, data capture systems require highly reliable software. While all software should be bug-free, data capture software with its limited communication back to its users, and its users' extremely limited computer training, is particularly dependent on non-stop, predictable, and accurate operation. The system should be designed to the greatest extent possible to be immune from problems from system failures, data base problems, program bugs, and especially user input.

Immunity from system failures can be provided in a number of ways. For a data capture system using IMAGE, the best way seems to be user logging. While requiring extra programming and extra design attention to be used effectively, transaction logging can provide up-to-the-minute backup for a busy data capture system.

Data base problems represent a broad class of difficulties such as broken chains in IMAGE data bases, DATA SET FULL errors and their equivalents in non-IMAGE systems, concurrency problems, and other difficulties in accessing or updating data on an otherwise healthy system. In general, the application should perform whatever checking is necessary to **commit the transaction to succeed.** This means checking data set capacities, insuring that records have not changed across locks, and perform whatever logical (data dependent) checks are necessary before actually updating the data base. Problems which are found should be reported to a central controlling process which can shut down the data capture system and inform the computer operator before any damage is done. In addition, the data capture system must **immediately** inform its operators that the system is no longer available. Otherwise, the operator may continue to "input" data, unaware that the system is no longer listening!

In order to prevent program bugs from causing serious damage, programs should be self-checking whenever possible. For example, if inventory counts are kept in several places (possibly split in several different ways), any time two or more counts are available, they should be compared. If an error is found, the problem should be reported to the operator, and depending on the potential seriousness of the problem, the data capture system should be shut down automatically. This strategy will prevent a "cancer" from occurring because of later transactions relying on the incorrect data.

In some cases, it is possible to repair damage to the data base in real time. For example, if it is possible to check a summary field against its details, this should be done whenever it is not inconsistent with performance. (This does not mean that the data check must execute rapidly; it may simply be performed as part of an infrequently-used function.) In the event of a discrepency, the damaged summary field should be repaired and the error logged. In this way, it is possible to insure maximum availability of the data capture system, while simultaneously maximizing its reliability.

High-volume reporting

In using a data capture system, a large amount of data can be recorded for later use. While it may seem obvious at the outset that these data will be too voluminous for repeated reporting, this fact should be kept in mind throughout the design of the data capture system. Reports should be provided with significant "filtering" capabilities so as to produce meaningful information from the huge amount of data collected.

User Training Program

Data capture systems present a special set of user training problems in that the system's users will typically have a lower level of computer education than a data entry operator. In order to mitigate this problem, the application should be designed so as to minimize the amount of training required. If a particular transaction requires more than one step to complete, the data capture system should guide the user through the transaction in such a way that only one path is possible. Error messages **must** be phrased in the user's "native" language; computer terminology should be avoided. **An uninterpretable error message will not be interpreted!** Failure to recognize this fact will mean lost transactions as users ignore rather than report cryptic error messages.

With this in mind, an effective user training program will be one which stresses training by example. For single-step transactions, the data entry becomes just one more step in a routine that the user is as already accustomed to as part of his or her primary duties. A multi-step transaction will require that the user understand something of the information that he or she is asked to present.

V. Evaluating your Data Capture System

After your system has been in operation for several months, you should begin to evaluate its performance and select changes to be made. Evaluating the system's performance relative to organizational objectives is beyond the scope of this paper, but four major operational areas should be evaluated:

- o Reliability
- o Operator acceptance
- o Information availability
- o User acceptance

Reliability

One repeatedly-stressed factor in a successful data capture system is its reliability. Uptime over the evaluation period should be measured, and any problems analyzed. Have there been communication failures? Have many problems been found by self-checking routines? Have the automatic repair and shutdown facilities been effective?

Operator acceptance

If the system has been reliable and available, it is important to measure operator resistance to the system. This is normally high at first, and then gradually decreases as data entry becomes a normal part of the operators' routine. (If the system has not been accessable, resistance will usually remain high regardless of other factors.) If resistance to use of the system is high after the system has been in place, the causes should again be analyzed. Have terminals been put in the right places? Are there enough terminals to prevent operators from standing in line? Is response time low enough? Is the user interface too complex, thus imposing a burden on the operator? Is insufficient feedback being provided?

Information availability

Since the primary purpose of a data capture system is to provide information which was too costly to gather with ordinary batch or semi-batch processing techniques, the first evaluation is whether the system is capturing the data it was designed to capture. If the operators have accepted the system, information should be available. If not, the design factors used relative to timely and useful presentation of information may need to be reexamined. Can reports be produced on an exception basis if desired by users? Can information on reports be tracked directly to physical activity monitored by the data capture system? Can information be obtained on-line so that users can "watch" the system work? Are users confident that they know the source of all of the data presented to them?

User acceptance

Finally, if information is available, properly presented, users should accept the system. Acceptance means **using** the results of the data capture system in their daily operations. If in fact decisions are made on the basis of information captured by your data capture system, it is by definition successful. After all, that is the end purpose of information systems!

Bruce Toback wears many hats at Infotek Systems, which keeps him very busy, indeed. What free time he does have is being currently spent in acquiring a pilot's license.