COMMUNICATION SKILLS REQUIRED BY COMPUTER PROFESSIONALS

Norman Enger
Applied Management Systems

ABSTRACT

Computer professionals usually have technical backgrounds that concentrate on information systems and computer science courses. These courses primarily emphasize the use of various technologies to implement automated systems. This emphasis often neglects the teaching of interpersonal communication skills and understanding of human factors as they relate to the design and implementation of computer-based information systems. The computer professional usually needs these neglected communication skills to successfully implement integrated information systems.

INTRODUCTION

The success or failure of a major computer system development effort often depends on the interpersonal communication skills of the project's systems analysts and software engineers. If computer professionals are aware of the human factors that inhibit effective interpersonal communication and impede the implementation of a system, they can often act to eliminate potential human barriers to the successful development of an information system.

Unfortunately, few data processing managers, systems analysts, or software engineers have received formal training in either interpersonal oral and written communications skills or human factors considerations related to information systems. Academic computer science and information system curriculums often stress technical system analysis and software engineering skills and minimize courses that cover human factors or behavioral sciences.

For example, the successful design and implementation of a major new system often requires marketing skills and an understanding of human and organizational psychologies. Often, data processing management and the technical staff must "sell" a system to both policy management and user personnel. They must overcome the human fears and resistance to change that are common to most large organizations. Management and the systems analysts become "marketeers" "selling" the benefits of the new system to prospective users. They often must make oral presentations promoting the development and implementation of a new system.

Many data processing managers have been promoted from the data processing ranks. These managers usually have strong technical skills, but limited formal managerial and business training or background. Difficulties can be created when a data processing person is promoted into a managerial position because a talented systems analyst or software engineer does not necessarily make an effective manager.

A recent book by J. Daniel Couger and Robert Zawacki, Motivating and Managing Computer Personnel, indicates that computer professionals have a lower social need and higher growth need than other professionals. The need for social interaction with co-workers or user personnel is low. Many computer professionals are "loners", who obtain great satisfaction from technical achievements and have limited interest in socializing or communicating with co-workers. This profile of computer professionals has implications for the skills required to develop information systems.

Organizations may have to develop special seminars and training classes to strengthen the oral and written skills of computer professionals. Data processing management may have to actively promote more effective communication between computer professionals on development teams and between the teams and users.

REQUIREMENTS ANALYSIS

During the Requirements Analysis phase of system development (See Figure 1: Structured Systems Life Cycle Phases), the functions to be performed by the new system are defined. The systems analyst works closely with the user organization to identify the user requirements that must be satisfied by the new system. The analyst identifies the information that is needed, the sources of the information, the destinations of the information, the use of the information, the required frequency of output reporting, and the response time requirements. The analyst will use this definition of requirements of the "problem" to be solved to develop the "solution" in the Logical Design phase. The user requirements become the framework for the future development of the system.

Interviewing users is usually the systems analyst's primary method of data gathering to identify user requirements. The analyst schedules interviews with the user organization to discuss the functions to be peformed by the new system. Unfortunately, the importance of interviewing skills is often not recognized by data processing organizations. The systems

analyst typically receives extensive training in technical subjects but little training in interviewing techniques or technical writing. Interpersonal relations and communications skills which are vital to effective interviewing are neglected.

Two types of interviews commonly used by analysts are structured interviews and unstructured interviews. A structured interview uses a set of predefined questions to guide the interview. It is normally used when several people with similar or related positions are to be interviewed, either individually or in small groups in an attempt to arrive at a set of conclusions. An unstructured interview is less formal and more flexible in structure. The format and direction of the interview session will vary greatly with the personality of each interviewer and interviewee. However, the analyst will still have established objectives for the interview prior to the meeting with the user.

Too often, if three different analysts are sent to interview the same user group and define user requirements, they will define three different sets of user requirements. The three sets of requirements will overlap but they would not be identical. A fourth analyst might interview the user group and find additional requirements not contained in the first three definitions. This variation in user requirements definition by the analysts usually reflects flaws in the interviewing methodology. If the "problem" to be solved has not been properly defined, then the "solution" will probably be unsatisfactory to the user organization.

Interviews must be carefully planned and the interview results should be checked for accuracy and completeness. Some general rules for interviewing are:

- Understand the business functions of the user organization being interviewed to better relate to the needs of the organization.
- Study the job functions of the person to be interviewed to understand his or her work responsibilities.
- Set objectives for the interview to guide the interview and measure the success of the interview.
- Have an appearance and bearing appropriate to a computer professional.
- Try to be a listener--do not dominate the conversation.
- Prepare questions to stimulate the thought process of the person being interviewed.
- Seek to make the person being interviewed relax, to develop a friendly context for the interview.
- Discuss user responses to remove ambiguity or contradiction and where appropriate, promote a more detailed discussion of an answer
- Set a tentative time limit for the duration of the interview.
- Document the interview results during and immediately after the interview while recollection of the interview is still vivid.
- Schedule a second meeting with the same person if additional information is needed.
- Use the interview as a vehicle to build a working relationship with the user organization.

Questionnaires can also be used for gathering facts where the analyst must contact large numbers of individuals and obtain answers to a variety

of questions. These questionnaires must be carefully worded to elicit meaningful responses that will contribute to the study. The questions must be tested for both clarity and objectivity.

The systems analyst must be able to prepare interview questions and questionnaires, conduct interviews, establish rapport with interviewees, and accurately document the information obtained from the interviews. Gathering data by means of interviews is a primary method of collecting data to define user requirements. The analyst can use the interview to win user cooperation and support for the information system to be developed by involving the user in defining specifications for the new system.

LOGICAL DESIGN

During the Logical Design phase, the "solution" to the "problem" identified in the requirements analysis phase is developed. This phase requires that the user requirements be transformed into a logical model of the new system environment.

A major communication challenge is the translation of user requirements for a major information system into logical design specifications. This remains one of the weakest steps in the entire system development life cycle. There is substantial risk that this translation will contain flaws that will, if they are not corrected, cause user dissatisfaction with the operational system. The analyst may misinterpret a user statement or fail to verify a user requirement. Any communication problems between the analyst and user will increase the probability of a poor translation.

The systems analyst must be able to communicate effectively with th user during this logical design phase. If the user is deeply involved during the design phase, he can often make suggestions which are useful the analyst. These suggestions may lead to methods for performing functions which the analyst may not be aware can be changed. Through user interaction, the analyst can quickly identify the functions in the user organization that have a high resistance to change. He can then, with full awareness of the attitudes of the user organization, proceed with t development of the new system.

When considering how one should go about designing a system or data processing program with active user involvement and whether there is som orderly process by which the design of a system can be organized, the to down design method emerges as the currently popular approach.

The structured top-down approach stresses extensive interaction dur logical design between the systems analyst and the user organization. Although referred to by such other names as hierarchical design or systematic design, most of the variations of top-down design have the same objective: to identify the major functions to be accomplished and then proceed to the identification of the lesser functions that derive from tl major ones. This functional decomposition results in hierarchy charts o the system that should be discussed, in detail, with the user organization. As the top-down design process proceeds from the identification o

the major functions and their interfaces to the breaking down of these functions into successively smaller ones, each sublevel becomes a self-contained component whose operation is subordinate to the next higher level.

Top-down design facilitates the clarity of communication between data processing and user organizations by producing graphic presentations of the new system that can be understood by and discussed with users. It results in the construction of a modular, hierarchical system and it is the isolation-by-function feature that can be used to great advantage to decompose, with the user, the logical structure of the system. Due to this functional modularity, the scope of needed changes can readily be ascertained, and the changes made and tested with a minimum of effort.

Perhaps the most important element contributing to the success of top-down design is its formal approach to the specification of each module's inputs, functions, and outputs. A precise definition of user requirements is essential to the development of a correct system or program. Too frequently, the contents of specification packages are vague, inaccurate, and incomplete, resulting in the developed system possessing these same characteristics. It is for this reason that several serious attempts to formalize the steps of top-down design have been made.

If a structured approach is being followed by the analyst, he can use data flow diagrams, a data dictionary, data structure diagrams, and transform descriptions to define the logical description of the system. Deci-

sion tables, decision trees, Warnier Bracket Diagrams, Nassi Schneiderman Charts, or Chapin Charts could be used to illustrate the logical processes and data transformations in the new system. Structured English or pseudo-code might be used by the systems analyst to describe policies and procedures in a precise form of English using the logical structures of structured coding.

One of the more widely known examples of a method to illustrate a system or software hierarchy is IBM's HIPO. HIPO is an acronym for Hierarchy plus Input, Process, and Output. It is one of several valid methods of graphically describing a software system or program as an arrangement of functions to be performed. HIPO provides both documentation and process visibility through its diagrams.

Some of the major benefits claimed by some users of HIPO in the design phase are:

- A description of system requirements represented in HIPO form is more easily understood by users than standard written specifications
- The capability of specifying specifications through HIPO allows a user organization to graphically describe a system and its functions.
- Users become acquainted with hierarchical design and the concept of "function" during specification development. This familiarity helps them to follow design walk-throughs and to understand their system more readily.

The hierarchy portion of HIPO involves a structure which is much like an inverted tree. This tree-like structure is comprised of functions,

actions, and objects. Each function is represented as a box and can be described within that box by an action verb and by an object (the data which is affected). Each branch of the tree will have its subordinate branches; therefore, any element of the tree can easily be traced back to its root. The hierarchy of functions is created by a technique known as functional decomposition, wherein a function is exploded into increasingly lower levels of detail until all subfunctions have been defined. The top box of the hierarchy describes the entire system or program as a single function, and each lower level function is a subset of the function above. The top level functions contain the control logic which determines when and in what order lower level functions are to be invoked, and the lower level functions are where the coding statements are predominantly found.

When the thoughts and ideas of the user and designer are not properly recorded, the results are inefficient use of time, implementation of ambiguous functions, and the creation of a system which does not adhere to its intended purpose. HIPO provides a means of more precisely defining user requirements and improving communication between the users and the data processing staff.

The systems analyst should be willing and able to explain or demonstrate elements of the new system to the user organization. To improve communication, the analyst will often prepare, for the user, examples of system output displays and reports. The user can review and discuss these

tangible system outputs with the systems analyst rather than just narrative descriptions. The software engineers might also develop a prototyp of the final system for the user organization. The prototype might have proposed visual and hard copy outputs derived from a test data base. On some projects, the prototype could be gradually refined by the systems analysts and software engineers into the final system.

STRUCTURED PHYSICAL DESIGN AND TOP-DOWN IMPLEMENTATION

During the Structured Physical Design phase, the logical design is used to produce detailed subsystem, data base, and program specificatio The logical design is matched to the physical software and hardware env ronment in which the system must operate. The communications challenge to translate the logical design specifications into physical design req ments. At this point, also, there is substantial risk that the logical physical design translation can be flawed.

Following the structured approach, hierarchy (HIPO) or structure charts would be used to delineate the software modules of the system. These charts show each module as a part of a hierarchy and identify inv cation, intermodular communication, and the location of major loops and decisions. An unresolved communications problem in the Physical Design phase is the role of the systems analyst relative to the software engin This issue, as to which of the two professional groups performs the phy ical software design work of this phase, has become a controversial iss

in the industry. Whichever group performs this translation function, there is a continuing need to communicate with the user organization and clearly define the physical software design requirements to the implementers of the system.

During the Top-Down Implementation phase, the software modules defined in the physical design phase are coded and integrated into the framework of the ultimate system. Structured techniques have proven to be very effective in this phase of the system life cycle. The analysts and programmers, following the structured life cycle, would use such techniques as top down implementation, structured programming, chief programmer teams, and structured walk throughs to code, program, and integrate the software.

Communication requirements during top-down implementation relates to team organization and structured walk-throughs. Working with a software engineering team requires that the Chief Programmer, Team Leader, or Project Leader and the staff have effective oral and written communication.

The technique of top-down expansion of functional specifications must be combined with the computer professional's perceptive analysis of the task to be programmed. Testing each level of a program before engaging in further programming will assure that the resultant program achieves the intent of the original specifications. Therefore, design and interface errors will not belatedly be discovered.

Top-down program development, starts with the functional requirements of the program and develops the design downward with the lowest level modules being designed and coded last. The traditional development phases--design, code, unit test, and integration--are no longer sequential processes for the entire project but are concurrent and overlapped throughout the development cycle. A program unit is coded only after the module which invokes it has been completed. After an input routine is coded, it is then tested and integrated into the existing code, which, at the same time, design and coding may be in process on another routine. This approach enables the unit testing of an element of the system as soon as it is completed and can be of great value in isolating errors early in the development process and in reducing the time spent in correction.

Structured walk-throughs can be used to improve the exchange of information among team members at various phases of a project. These are formal reviews, held at specific points throughout the development process. They are formal, with an agenda, and all participants are walked through the problem, program, or project. In the Top-Down Implementation phase, the purpose of a walk-through could be to insure correctness of a program, or module. A walk-through will be set up to determine the completion of a milestone event wherever possible. When complex subjects are to be addressed, all participants should be provided working papers in advance. All attendees should be aware of what is expected of them and what is to be accomplished. The purpose of a structured walk-through is to determine if the design or coding is correct.

If necessary, a separate meeting should be scheduled to discuss options and alternatives directed toward improving a system or code. The person conducting the walk-through has the responsibility to keep the walk-through direct and effective. When alternative approaches that could improve the design or coding are identified, it is the responsibility of the Chief Programmer or the Team Leader to insure that a separate meeting is scheduled to investigate these alternative approaches. The backup for a project may be established by having backup personnel attend these walk-throughs.

Attendance at structured walk-throughs will vary depending upon the specific subject being discussed. Users may have to attend meetings discussing the system's functional capabilities. Management attendance may be required when information on subjects outside the jurisdiction of the team is required, or when the status of the project is to be reviewed. On projects that have been set up for structured testing, the test supervisor should attend appropriate walk-throughs.

The programmers on a software development team may have to communicate and interact with a Librarian who manages the written and machine documentation of a software development effort. The Librarian maintains a support library for a team effort. The programmer will usually enter modifications to programs and will initiate compilations after the first one. Unit testing will always be directed, and normally accomplished by the programmer.

The Librarian catalogs all formal changes to the contents of the library. No version of a program or document can be entered or delete from the library without the Librarian. Usually an initial entry of p gram and changes is made by the Librarian. Programs for system testin and integration are provided from the library. Prior to the performan of a test, the Librarian selects the program requested from the librar and transfers the responsibility for it to the Test Director. When a is completed, the Librarian will, as directed by the Test Director, id tify and/or store, in the appropriate form, the versions of the progra tested.

A catalog or index of the library contents is maintained by the Librarian. Each document or program entered or deleted will be record to include the date, action, and if appropriate, the authority of the action. The Librarian will usually initiate the first compilation of program. Subsequently the Librarian will, when requested, initiate fu ther compilations and test programs. These may be done by the Librari only if sufficient instruction and test data are available.

In summary, during the Physical Design and Top-Down Implementatio phases, there is a need for the systems analysts and software engineer communicate between themselves, with user organizations, and as member development teams. Top-Down Implementation encourages team development efforts and structured walk-throughs which in turn require effective

inter-personal communication. Chief Programmer Teams have Librarians who are responsible for the external (written) and internal (machine) documentation of a software project.

ACCEPTANCE TEST GENERATION PHASE, QUALITY ASSURANCE PHASE, AND SYSTEM OPERATION

The structured systems life cycle provides for an Acceptance Test Generation Phase and a Quality Assurance Phase. During the Acceptance Test Generation Phase, test specifications and test cases are developed. During the Quality Assurance Phase, acceptance tests are performed using the test cases and users are trained to operate the new system. Structured testing techniques would be used during the quality assurance testing of the system and, therefore, each level of the system will be tested to assure that the new system satisfies the acceptance test specifications and that design and interface errors are not discovered after the system becomes operational.

When a new system is installed, there is often a major change in the environment of the organization. The employees may have different job functions or be separated from co-workers. Work formerly performed by the employee may now be automatically produced by the computer system. This may be a report that was formally manually prepared but that now is computer produced. This in turn may cause a feeling of loss of importance in the worker. The worker may discover that his contribution to the end

product of the system is no longer very visible and they therefore may have less interest in the end product. The unhappy employee may express his or her hostility by deliberately reducing work efficiency or making system errors.

Job security fears may also arise. The workers may fear that the new system will displace them. If the workers have not been involved in the development of the new system, they may regard it as an undesirable external force. Fears may be overcome if the analyst involves end users in the development of the system, asks them to contribute ideas during system development, and provides them with adequate pre-operation training.

Employees may be reluctant to learn new computer related jobs and overtly upset the system. This reluctance may stem from feelings of personal inadequacy. They may fear complexities in their new job role. The analyst should try to overcome early signs of resistance before it hardens. He should try to develop good working relationships with the users and stress the benefits of the new system. At some point, trust between the user and the analyst should be established.

Even if the user and data processing staff interact well during the development and design of the system, they may fail to consider the clerical staff that will operate the system. A system which is designed without consideration for the clerical staff that will use it, is likely to result in a system that will be difficult for them to use. The effect new

procedures will have on the clerical staff must be considered. Work must remain challenging to the clerks and they must feel they are constructively contributing to the system or otherwise boredom will set in, there will be many careless errors, or they many sabotage the system. Often, the effect of a new system on office attitudes is not considered.

The testing of computer programs and systems prior to implementation should be a formal process in which the input is well defined and the expected results are known. Effective testing also has the characteristic of enabling the quick isolation of errors when they are encountered and should be an orderly and well planned process. The Test Plan should indicate who will perform the test, the environment of the test, the input descriptions, the output requirements, and the test process. These items should be addressed in sufficient detail to provide understanding.

Testing discipline should be used to insure that there is an orderly, step by step process of testing. This involves the identification of a supervisor, the establishment of a specific order of test, and the method of keeping track of progress. Using top-down development, programs and/or modules will be created in the sequence in which they are to operate. A level of detail of the Test Plan should be set so that an orderly sequential test can be defined and tracked through the testing process. The Chief Programmer, Team Leader, or Project Leader will construct this chart. A program/module to be tested may range from a single module of one page of coding to a major part of a full system of several thousand lines of

code. The System Operation phase begins after the system has been fully tested, user personnel have been trained, and conversion has been successfully completed.

CONCLUSION

The computer professional needs various communication skills to properly implement integrated information systems. These skills are often overlooked in conventional educational programs for computer professionals.

During the Requirements Analysis effort, interviewing techniques must be improved to better define user needs. Rapport with the user organization must also be established to develop a cooperative working relationship. During the Logical Design phase, a communications problem exists in trying to effectively translate the user requirements into logical design functions. Top-down design promotes communication by using graphic displays of the hierarchy of a system. During Physical Design, problems may exist between the systems analysts and software engineers who often fail to agree upon the boundaries of their work domains. The development of software during Top-Down Implementation requires effective inter-personal communication for team operation and structured walk-throughs.

Finally, during Acceptance Test Generation, Quality Assurance, and System Operation, the analyst must calm user fears of the new system and overcome any resistance to change. Usually, the computer professional requires numerous inter-personal communication skills during the development of major integrated information systems.

Norman Enger, Ph.D., C.D.P.

Norman Enger is the President of Applied Management Systems, Inc., an international consulting company serving a variety of industrial, institutional, government, and scientific clients. The company's activities have included the design and installation of large telecommunications networks, the design and implementation of numerous computer applications, the design of data base systems, the education of data processing personnel and the development of complex simulation models. Prior to founding Applied Management Systems, Inc., in 1970, he was a Director of Systems for the Control Data Corporation.

He is a frequent speaker before the ACM, DPMA, and other professional groups on the design and implementation of computer based information systems. His latest book, Computer Security: A Management Audit Approach was published by the American Management Association in 1980. He is the author of four other books and a large number of professional articles.

Since 1967, he has been an Adjunct Professor of Information and Management Sciences, School of Government and Business Administration, The American University. In 1979, he was presented with the Distinguished Adjunct Faculty Award by the American University's Center For Technology and Administration.

A Certified EDP Auditor (CDPA) and a Certified Data Processor (CDP), he is a member of the Association for Computing Machinery (ACM) and the Institute for Electrical and Electronics Engineers (IEEE).