USER PARTICIPATION IN SYSTEMS PROJECTS

Mohamed Alibhai
Department of Personnel Administration
Commonwealth of Massachusetts
June 1990

CONTENTS

I. INTRODUCTION: THE USER

This brief document is for you, the *user*. Its purpose is to describe what you can do to enable Systems (or MIS) to develop applications that are as close to your requirements as possible.

You are a *user* because you work with computerized data or with data in the process of being computerized (e.g., when you prepare source documents for online data entry).

The nature of your relation to the computer system determines the type of user you are. There is the user (e.g., President or CEO) who gives the all-important OK for a new system, but usually never sees it. At the other end is the data entry user who enters fresh data from source documents (e.g., bills, invoices, etc.). In the middle is the user who receives the computer's output in the form of screen displays and reports. The vast majority of users fall in this last category. The shop-floor manager at the Pratt&Whitney aircraft plant gets reports that tell him how far each engine has been assembled; the manager of billing gets all sorts of reports about customers and how much they have been billed; the bank manager gets a report of "exceptions" such as bounced checks; etc. Computer output has become the lifeblood of one's work. Think of what happens when a computer "crashes."

A user is not a programmer. It is not his job to develop the software and data base that will provide him with the information he needs for his work. Rather, the user relies on the programmer (1) to manage his data, (2) to process it, and (3) to present it to him in a usable form. Whether something is usable or not depends on the user as much as it does on the programmer.

Thus the user and the programmer cannot avoid working together. Each has a perception (i.e., a working model) of what the other is like. Since the user's participation in the work of the programmer is dictated by his perception of programming and his relation to it, it is important to have a correct perception of systems development as such and the role of the user in it.

II. USER PERCEPTIONS OF MIS

1. Input/Output User. A widespread view among users is that they are users only in the sense that they use the system built by programmers: one enters data into this system and one uses output reports and screens for meetings and other decisions. According to this perception, the user has little to do with the *development* of the system. This is a very entrenched view across the country. But it does not square with the daily life of the user.

For the fact is that the user is the source of the data and *how they are used* by him. The programmer only automates these rules for the user. Thus the user is part of the *total sphere of data flow covered by the computer* in your organization; and the combination of the computer system and the network of users who work with it is the *information system*.

2. Independence of Business Procedure from Computer Procedures. Users think that their "business" procedures are unconnected with the operation of the computer system. This perception is particularly important when a manual system is replaced by an automated one. If you buy an airplane but drive it on the ground like an automobile, why buy it? Why not just stay with automobiles? An airplane's singular characteristic is that it flies off the ground. You "fly" with the computer by automating the processing of data.

Just as the shift from the auto to the airplane requires a corresponding shift in the procedures for using them, so the shift from a manual to an automated system requires a new set of logistics. Things that could wait till the last minute in a manual system may now have to be done at the start of the process in the new system. So it is simply not true that the "business system" is completely separate from the "computer system."

3. MIS' Knowledge of Business System. Another widespread perception among users is that MIS is familiar with the details of the user's work. This leads them to think that it is unnecessary to describe fully their work process. The user will assume that the programmer is also fully aware of the *logistics* of the specific function (monthly reports should not be run before the month has ended, etc.).

4. Impact on Development. These "where-you-are-coming-from" perceptions can have considerable impact on development projects. The rules and the data they govern are derived from broader policies of the organization. These business policies are formulated by the organization's senior management. They are not formulated by MIS.

Yet MIS is charged with processing the organization's data in conformity with the rules that define how the data is to be used. Should the two items be added or subtracted? The programmer knows how to add and subtract. But only the user, for whom he is preparing the screen display showing these two items, knows (and should know!) whether to add or subtract. When users fail to be specific, the programmer is forced to guess the answer, and he may subtract rather than add. The user then gets upset ("Can't you add?"), and "after-the-event" changes must be made.

III. DIFFICULTIES FACED BY USERS

1. Language Barrier Analogy: Translation. Programmers are specialists in a technical trade, which might be likened to language. If I am an expert in Swahili and you want to translate an English document into Swahili, well, I must have the entire English text. If you don't have the text and say, "Mohamed, you know what I want to say, so just go ahead and do it in Swahili," the Swahili fellow is in danger of getting a garbled message.

Everyone knows that translations are not literal or word-for-word: the order of nouns, verbs, etc. differs among languages; some languages have no such thing as "is," "are," or any form of the verb "to be." Swahili will render the "message" in a different form.

Similarly, systems are "translations" of the user's "language" and his "message." The computer language might be COBOL, FORTRAN, dBase III, NATURAL, etc. Users have their data in the form of documents and other paper; these documents are filed, typically, in filing cabinets, bookcase-type shelving systems, even on one's desk! These same data, when they have been passed to the computer system, are stored in an entirely different way from the way they are stored by the individual user. The system for organizing a library is vastly different from the system (if there is any!) you use to shelve your folders and manuals.

2. Language Barrier Analogy: Accountant. Consider the accountant. All accountants, everywhere, learn the same body of knowledge. The rules of accounting are the same no matter what the nature of the organization.

As a student, the accountant does not learn the specific accounting systems of individual businesses. He learns the principles of accounting. When he joins his employer, he has to learn the financial processes of his company, and then create an accounting system that is tailored to the unique features of his employer. The flow of money in and out of the organization is henceforth governed by a new set of rules, that of accounting.

For example, if he is working for a bank, his details will be customer ID, deposit amounts (checking, saving), loans, etc. If he is working for the payroll office, his details might be employee ID, social security number, salary, benefits, other financial data. For the accountant to successfully tailor his accounting knowledge and fit it to a new environment, he has to have as much information about this environment as possible. But the basic principles of accounting remain the same: things must balance and reconcile; any money transaction must be "accounted" for by a corresponding physical event.

3. User's Unconscious Knowledge. There is, however, a special difficulty faced by the user in his efforts to describe fully his activities. Most of us are *operationally fluent* in our work. Our work has become so routine and automatic that we just move from step to step, unaware of the rules and conditions that we are applying to the materials with which we deal daily. There are many *if this...then this* actions we take without consciously pointing them out to ourselves at the moment we are doing them.

Consequently, unless the user makes a special effort to step back and identify the rules that govern his work, critical information will be unavailable to the programmer.

4. Analogy: Grammar. When we use our language, our sentences turn out to be *grammatically sound* even though most of us might not be able to describe the grammatical rules to which we are conforming. Our actual utterances conform to the rules of English syntax, yet few of us will be able to list these rules. We have operational fluency in English.

Children learn to speak grammatically long before they are subjected to the tortures of grammar at school. Among the so-called primitive peoples everyone speaks with complete grammatical consistency, but nobody knows any grammar!

What if, like an anthropologist, you are among the "natives?" You have a dictionary with you. But you soon discover that it is useless, because you don't know the native's grammar. One thing you might try is ask around. But that's hopeless, because the native doesn't know the parts of speech: verb, noun, adjective, etc. You have no choice but to observe what the native does with words and search for patterns. You are "extracting" or "constructing" the native's grammar.

The user is like the native! And the programmer needs to know the rules of "grammar" of the business process, not just the vocabulary! ("What do you do when a check bounces?" will give the rule for dealing with bounced checks.)

5. Localization and Turfism. But the most important limitation faced by the user is *inherent* to the very nature of an organization with a hierarchical (i.e., vertical) command and reporting structure.

The employee has clearly defined instructions: what data to process, how to process them, and when to process them. Another employee in a neighboring unit also processes the same data, but he uses them in a different way. Work is compartmentalized, so is data flow.

A consequence of this "pyramid" model of the organization is that *turfism* develops as a byproduct of this compartmentalization. People are held accountable only for life within their turf, and they are naturally less disposed toward worrying about the big picture, which they are asked to leave to the "senior people."

Yet it is undeniable that information crosses these turf borders and is the common currency of the organization. Information processed by one group affects the information processed by another group. The group that changes the value of a data item is not technically responsible, and does not feel responsible, for the fate of this data item beyond its boundaries.

IV. DIFFICULTIES FACED BY MIS

Consequently, no one in the organization has the full picture of the information. Job responsibilites prevent such entity-wide knowledge from being acquired.

MIS, however, *must* look at data and information processing as a whole. Its backbone is the integrated data base from which each turf-unit draws the data relevant to its work. But the data remains "public" in that it is available to another turf. Data processed by Exam Administration flows into Helen's turf (Local Government Services). If something is changed by one turf, it could affect the information of another turf. The first turf will probably be unaware of the impact of its action. Maybe it doesn't care, since this effect is felt to lie beyond its borders.

V. SYSTEMS ANALYSIS

That is why, from the earliest days of data processing, a special class of systems personnel has had to focus on *extracting* these *tacit* rules out of the users. These are the *analysts*. It is their main task to interview or interrogate the users and ask questions designed to extract the "grammar."

But more important, MIS is today the only functional unit within an organization charged with taking a global view of the organization's information, reducing duplication, and ensuring that changes made in one place are immediately made available to other parts of the organization. And it is the *analyst's responsibility to take this global view* and develop systems that operate under this global view.

VI. SYSTEMS DEVELOPMENT

An overview of what happens in systems development will enable the user to appreciate the scope of work involved as well as have a sense of how much time will be needed to do some project. [A separate document, *Systems Development: The Main Steps*, describes in greater detail the main activities that together make up the development process.]

There are at least eighteen distinct steps. They are listed below; for the narrative associated with each of these steps, consult the document *Systems Development: The Main Steps*.

Pre-Project Activities

1. Project Initiated
2. Project Team
3. Project Scope
4. Project Plan

Project-in-Process

5. User Requirements
6. Systems Analysis
7. Systems Design
8. Programming, Unit Testing
9. System Testing
10. User Tests
11. User Acceptance
12. Disaster Recovery
13. Documentation
14. Training
15. Turnover
16. Preliminary End of Project

Post-Project Activities

17. Monitor System Performance
18. Final End of Project

Since actual programming work varies widely in scope and sensitivity (i.e., potential for major damage from a small error), *common sense must temper the use of these steps.* In general, for new, large and sensitive projects it is in the user's own (self-) interest to want to follow the "full course."

Example: One does not implement the Library of Congress classification system to organize and catalog the few books in one's home. In fact, most people have no cataloging system at all; they know where the book is. Similarly, there is no need to spend a week doing the things indicated above for a task that is simple and a few hours long. Let's not use a sledge hammer to smash a tiny, light bug when blowing a puff of air will send it off!

There is, however, an important point that the user needs to note, appreciate, and (hopefully) remember when dealing with systems. Systems development should not be construed as a form of activity that can be started, interrupted, or terminated by fiat without potentially serious damage to the data. And bad data can land us inside the courtroom.

Users can form a reasonable estimate of the scope of a project by referring to this breakdown and assigning times to each of the steps. Any reasonable project, that is, any project that will require us to work through all the eighteen steps, will be several weeks long (a rough 1-day-for-1-step formula gives 18 days off the bat, that is, three weeks).

Users tend to think of systems development as just banging away at the terminal; that is, they tend to think of it as simply coding and programming. The breakdown above should clear this misconception. If the frontend steps have been completed thoroughly, the programming step is more efficient.

VII. USER ACCESSIBILITY

Imagine you are at your bank, cashing a check. You join the line at the *Enter Here* sign. You are patient. And soon you are "there," handing your check to the teller. Behind you are other customers, waiting for you to finish. The teller examines your check, inquires the terminal, etc., and wants you to provide her with some more information.

At this point, you say: "Gosh, I just remembered! I have to meet this person right now, I really can't avoid it. But I'll be back in a second. Just hold on for me, ok?" and dash out running for the all-important meeting. The teller has your check. She's supposed to wait for you.

You get the drift of this analogy. MIS is your banker, and your data base is your data bank. There are customers queued up for their respective "transactions" with the "teller."

Accessibility is a matter of inter-turf relations. There is no participation if there is no accessibility.

1. Conflict of Project Plans. The user's accessibility is linked to his work plan in his own group. This work plan has completion dates (drop-dead dates) set for him by his group. MIS projects are different from projects within other functional units of the organization. A systems project is an internal project of MIS. But it is also an internal project of the functional unit (say Exam Administration) for which MIS is doing the work.

Since the user does not construct the (software) system, he will (and does!) devote time to *other* projects that have been assigned to him by his functional group. These other projects too will have delivery dates. So he will obviously work on projects that need to be completed before the project that will be done by MIS.

Under such perceptions the user decides that he really doesn't have *to get serious* about the systems project until he has finished the others before it. Further, he believes (usually) that he doesn't really have to contact MIS about it until the others have been completed. Now if this systems project is such that it really must begin several months ahead of the user's other projects, we could be faced with some problems. Let's consider an example.

Mike has three deadlines: March, May, July. July is the deadline for his systems project (to be done by Mohamed). And Mohamed really needs to begin in January if he is to meet the July date. Mike, meanwhile, has to answer to his other projects, and devotes his time to them, and doesn't feel it necessary to tell MIS about the systems project until June (when presumably he has completed the other projects); and so Mohamed doesn't hear about his project until early June. Again, you get the drift of this example.

But there is more. Let's assume that Mike has been a good kid, and *did* contact Mohamed in December last year, and Mohamed *was* able to get started in January. But once Mohamed starts working on the project, Mike has no time for Mohamed because he has to work on those other projects that must be completed between January and July, i.e., those with due dates in March and May.

Meanwhile, Mohamed has a schedule worked out for him by *his* group, and he is scheduled to begin work in August on another request. But he can't get Mike to work with him because Mike's priorities are different. Mike is like that bank customer above who dashes out, leaving the teller (Mohamed) and the other customers stalled. Is there a solution to this conflict of project plans and priorities?

2. Resolving the Conflict. The most important action the functional units can take is to think as far ahead as possible, identify projects that will be done, and fix their completion dates.

For all those projects that involve programming work, MIS should know about them well in advance; and it should also know *the user's internal completion date*. MIS needs this drop-dead date to determine (by working backward) the *latest start date* for the project. If a project has March 31, 1992 as its drop-dead date, and MIS determines it to be a 10 month project, the project must start *no later* than June 1, 1991.

A second important action the functional units can take is to allocate time to the individual (who will be working with MIS) to become a team member of the MIS project. This individual must be able to respond promptly to the development team's needs.

These recommendations have become rather necessary in the dramatically altered context of the state's budget crisis.

VIII. SOME FINAL TIPS

1. Know Your Data. The most basic "homework" you should do is to become fully conversant with your own *data*, the *mutual dependence* of the data, the *rules* that govern their use, and the *time conditions* that control when and how they are used. You are dealing with *events* that make up the movement and transformation of data through your functional jurisdiction. The crucial thing about *events* is their sequential dependence on each other: e.g., the scanning step must be completed before any analysis of data can be (meaningfully!) done.

Remember that MIS is looking for data, the relationship among them, their respective processing rules, and the *scheduling* characteristics of your work. If MIS wants to know what you do with data item A, *and you know that data item A affects the value of data item B*, tell that to the analyst or the programmer if he is not aware of this fact.

2. Think Ahead Your Product. *Know the final form of your product* as fully as you are capable of thinking ahead. If you have been designated as the source of information about the requirements for the system, think ahead to the various components of the final product, and indicate the details of these various pieces. *Use paper and pencil*. In attempting to visualize the final form of your product, the easiest way to begin is with (1) screens and (2) reports. Ask yourself: "Now what do I want to see there?" The *Checklist* should assist you in what to consider when you want to give "flesh and blood" to your as yet unborn product. From this exercise you should be able to list the data items you want to see.

3. Prevent Resource Contention. A user (whether individual or group) who knows all his data and how and when they are used is helping to speed up the development process. He is helping to minimize re-programming time that is required when the user's intentions and the programmer's understanding are operating on different wavelengths.

Finally, by being prepared with his information, such a user is enabling other users to gain access to the limited resources of the programming group. Users are in contention with each other for systems work, so if one user is dragging the project other users are delayed.