System Implementation and Evaluation

I. Trotter Hardy

William & Mary Law School

Repository Citation
Hardy, I. Trotter, "System Implementation and Evaluation" (1973). Faculty Publications. 1232.
https://scholarship.law.wm.edu/facpubs/1232

Copyright © 1973 by the authors. This article is brought to you by the William & Mary Law School Scholarship Repository.
https://scholarship.law.wm.edu/facpubs
Once you have decided which method to use to implement a system, there is still a lot of work to be done. The author looks at each of the steps that must be taken to make the system successful.

There are essentially four ways to implement a new system. If no prior system exists, the new one must be installed whenever it seems ready. If there is an existing system, there are three choices: terminate the old system and begin the new; phase in the new system gradually, one segment at a time; or run both systems in parallel for a time, terminating the old system after the new one has proven itself. Each method has its advantages, although the extent of the advantages depends on the type of implementation and the current situation.

The first option is not really open to choice; if there is no existing old system, there is no alternative but to install the new one after suitable preparations. It can be installed all at once or in segments.

The second method, cutting off the old and installing the new, is risky. Where there is little complexity, or where bugs can be corrected easily, such a procedure may simplify installation enough to warrant the risks. But for an implementation of any size or complexity, it would not be advisable. In the latter case, if the system should malfunction—and it can be expected to—the organization is left with no backup, no reserve resources to carry it through until the new system is corrected.

The third method of phasing has definite advantages. But such a system must be divisible into fairly “natural” parts; it must illustrate an independence from part to part. This way confidence in the whole system can be built up from confidence in subsets. As Murdick and Ross point out, if the system is really suited to this kind of independent analysis, there is good reason to question whether it is in fact a system, where such a term implies an interdependence and an integration of all parts of an organized whole. However, if the new system is merely an automated version of an older procedure that does lend itself to segmentation, then a phasing-in may be appropriate.

The final choice for implementation is to run both systems for a time in parallel, waiting to see if the new one is going to be successful before final con-
version. This approach does away with a lot of the problems of the other three, but it does have its own significant drawbacks. It is likely to be very expensive and very complicated. For the duration of the paralleling period, two complete sets of files must be kept; two complete updating routines must be undergone; and reports and other output from both systems must be produced and compared. If the systems are complex, the two outputs probably will not match exactly, no matter how much they are scrutinized and revised.

Proper Method

What, then, is the proper method? The answer depends on the particular system and the organization in which it is being implemented. For simple systems the second method should prove effective. For upgrading an old, segmented system, the phase-in method is reasonably sure. For complex systems where the application is critical, the only choice is the expensive one of parallel operation.

Once the method of implementation is chosen, a great deal of work remains. This preparatory period is one of planning to ensure the implementation goes as smoothly as possible. First is task identification. All work associated with implementation must be detailed and carefully broken down into its constituent parts. Such work will include forms design, programming, physical plant layout, file-making, testing, conversion, and many other tasks.

Once these tasks are expressly identified, the second planning step can be undertaken, that of scheduling tasks in an optimal way, so that over-all time and cost are minimized. Any technique that can help with this scheduling should be considered: Gantt charts, PERT, or CPM. All these techniques have the advantage that they show graphically how the implementation is proceeding, and are designed to show up any activity that is running behind time. At this point, if the new system is going to include new hardware, planning should begin on the facility arrangement to accommodate the equipment. Planning for a computer site is not trivial, due to the complexities of temperature, lighting, humidity, flooring, power, backup power, fire protection, and other requirements.

Organizing

After planning is well underway, thought must be given to organizing in the most efficient manner for the implementation. Generally, an implementation project manager is desirable, and often (and profitably) he is the MIS development manager. His duties are similar to those described by Sanders for a computer implementation officer:

- Review policies, objectives and target dates . . . established by top management.
- Submit an implementation plan for top-level approval.
- Select the . . . personnel needed to carry out the conversion.

The systems people chosen will typically be members of the MIS Team, but this is not necessarily so. It may be useful to have unrelated MIS-trained people move in to help spot bugs and trouble areas. This is especially true during the evaluation phase. The last key elements in the team organization are the line people involved. These should be the people who are going to use the system, who requested it in the first place, and who are going to be responsible for its successful operation. To erroneously conclude that their participation at this point is marginal to the success of the system would be not only detrimental immediately, but probably could be fatal for future systems.

In the discussion of work task scheduling, mention was made of file-making. This is the type of significantly difficult procedure that requires the development of a formal procedure of its own. The questions surrounding this activity are numerous. If the files are to be created anew, where does the information come from? In what form will it be? Will it need extensive processing? Is it reliable? If the files exist already, how will they be converted without crippling their on-going use? At what point are the data in the new file sufficiently complete and accurate to warrant cessation of use of the old? Are the data in the old file complete and accurate to start with? These problems necessitate careful analysis by the project leader and the implementation team; some kind of explicit procedure for treating these problems must be authored at this point. And with the new system and its associated files there will be new manual procedures as well. These will also need to be planned along with the general requirements of any necessary paper forms.

In addition to manual procedures development there is the need to develop adequate test procedures. Good testing can often be allowed for in the actual design...
phase, for it is there that testing features may be built-in to the system to simplify matters later. Even if built-in features are not appropriate, some idea of the areas that need extensive testing can often be identified and documented for later checkout.

A good overall summary of procedural requirements, paraphrased from Benjamin Conway, follows:

- Procedures are adequate for all personnel to know what is expected of them.
- Procedures are fully documented.
- Procedures for formal notification of procedural changes are developed and tightly adhered to.
- All possible error or breakdown conditions will result in documented corrective action.
- Controls are established to ensure conformance with procedures.
- Performance measures are developed so that management can know without question when the system is officially operational and how well it is operating.
- Controls are established on personnel practices, so that any anticipated manpower savings can be realized by actual personnel transfers or modified hiring practices.\(^6\)

Training

The next area of planning concern is user training. Training can be broken down into three fundamental types—training for management, for professional support personnel, and for operational people. Each of these three groups has different requirements for both depth and scope of knowledge. For management, the need is typically met with two short seminars.\(^7\) Professional support people will require a great deal more than this. Actual classroom sessions are often required, as enough knowledge must be gained on their part to assist other users, to maintain and to modify the system. For operational personnel, the needs are several. They must be made thoroughly familiar with the workings, the mechanics of system operation. They must understand what the system is not supposed to do and they must know what changes in their job functions and responsibilities will occur as a result.

When implementation planning is complete, attention must be turned to implementation itself. Although unglamorous, producing the paper forms that must be a part of the system is an essential first step. Such forms are the major interface to the mechanized system for most users, and are required for logging data manually, for documenting errors, for distributing procedures, for recording calculations, for person-to-person communication about the system generally. A form must be readable, useful, have enough space to fit in all required information, and be logically organized. It must have all the attributes of any well-designed procedure. Good forms are usually simple, have clear labels and directions placed within the box to be filled wherever possible, do not assume user familiarity with the organization’s jargon and acronyms unless such assumptions are fully warranted, speak if possible in human oriented and not in computer oriented terms, and contain all necessary information relevant to filling out the form with minimal references to other documents, procedures, etc. It is not at all inappropriate to have layout and/or graphic arts personnel involved in the paperwork forms design.

Files

The next critical activity in implementation is the actual establishment of the files. Careful planning and detailed procedures as indicated previously will be put to the test at this point. At some point further into the file-building phase of implementation a crucial “go/no-go” decision must be made and the old file system must terminate. By this time confidence in the new system must be well-grounded and all files up-to-date and functioning.

Testing of the system is the essential last step before actual cutover. Testing can conveniently be broken down into three smaller steps, which provide for the testing of components, subsystems, and the system as a whole. All three steps should ensure that the appropriate system element:

- Works at all.
- Works under widely varying environments, such as different people, different functional areas, different times of the day, different volumes of data, etc.
- Works under “noisy” conditions, i.e., where data are erroneous or sloppy, where input is improperly keypunched, where users are not fully experienced, etc.
- Works accurately and as it was intended.
- Component testing is the easiest; among the components to be tested are equipment, forms, software routines, data collection procedures, work procedures, and reporting formats.

In the case of subsystem testing, larger segments will be put through the testing routines and correspondingly more complex problems will show up. More emphasis must be placed on varieties of input and attempts to check out all possible combinations of requests, updates, and manipulation of numbers. This ensures that all logic paths on the system have been tried and executed successfully. When the subsystem phase goes smoothly, a test of the entire system is in order. At this point a number of new aspects enter the testing procedure:

- Data prepared by the users must be utilized.
- All data flows must be tested—computerized and manual (paperwork).
- All subsystem links must be tested, including man-machine interactions.
- The user himself should participate.
- If thorough enough, the system test may serve as an acceptance test.\(^8\)
Finding Problems

Emphasis during the system test should be on heavy use of the system to uncover as many problems as possible in the shortest possible time-frame. This time is also a learning period of considerable value to all people involved. All operating personnel connected with the system should be encouraged to use it and work with it as much as they can, with the understanding that this is a testing period and it is not anticipated that everything will proceed smoothly. Care must be taken to avoid developing a complex set of testing procedures that will not be used during actual system operation; an extra workload of this type merely causes resentment and artificially complicates what is not likely to be a simple system in the first place. In addition, it increases the likelihood that actual operation will turn up bugs that are unrelated to the testing procedure.

Programming, operations and user documentation start when the system is relatively well fixed in its format and operation, usually between testing and turnover to the line personnel for their control. Some project managers insist on documentation being kept up right through the planning and design stages, to ensure that it is comprehensive and available by implementation time. This has a number of drawbacks, however. The chief one is the detrimental effect it has on morale. As a rule, programmers despise doing documentation even once, let alone continuously throughout development.

Systems are liable to such great changes over their lifetimes that most early documentation will wind up being completely re-done, possibly many times, before anything is ready for operational use. This later documentation prepared during implementation does not mean that a project file of significant correspondence, decisions, meetings and frustrations should not have been kept up from the first MIS conception. Likewise, a systems file should have been established toward the end of the design/beginning of the programming phase.

Once the system is installed and working, attention cannot simply be diverted elsewhere. Well-documented follow-up analysis can be of immeasurable use for future systems. Further, some idea of how closely the system stuck to budgeted levels should be obtained, especially to facilitate future planning. And, too, the system needs to be evaluated thoroughly on technical grounds—how well did it match up to planned capabilities? The best evaluation is the successful use of the system under all possible contingencies; seldom, however, can this evaluation be rigorous enough in practice. A proper evaluation, as does a proper test, must be formally planned and executed.

An acceptance test that covers all of the required features of the system is a suitable vehicle for evaluation. If a comprehensive test cannot be prepared then a number of subtests should be arranged and exercised in a variety of sequences and mixtures. Particularly important to evaluate are any operations that should execute concurrently, such as information retrieval and update, data base access from multiple locations, inquiry and printing.

If specific response time requirements were levied on the initial system design they should be checked out under full system loads. The question of who should do the evaluation is significant. It must be someone from or representing the user group, not the MIS design and implementation team. And finally, a formal report of the evaluation results should be prepared and forwarded to top management for approval and acceptance.

Although everything about the system may have checked out perfectly up to this time, control procedures must be implemented to handle unexpected problems as well as the day-to-day operation. Periodic checks into the accuracy of the data base should be made, along with planned checks on the accuracy of all "table" files: lists of rates, taxes, addresses, prices, costs, etc. that occasionally change and need to be emended in the files. Some aspects of the system will need modification from time to time and programs will always have some bugs that do not show up until long after implementation.

Top management has a particularly strong responsibility in the area of control: middle and lower level managers must not be allowed to circumvent procedures or sabotage the whole effort. Management generally (outside of ADP management) must consciously strive to understand enough of their systems to realistically evaluate their performance.

Conclusion

Successful system implementation, according to one definition, is the "continued acceptance of the outputs of the system by the user." Implementation thus puts the burden of continual proof on the system and ultimately the system designers. Though stringent, this philosophy should have its payoff in better system design efforts and a better relationship between user and systems designer.

References

7. Murdick and Ross, p. 514.