

Should we build our own MIS?

Excerpted from “MIS for Microenterprise: A Practical Approach to Managing Information Successfully”, by Charles Waterfield, published by the Aspen Institute, 2002

After reviewing commercial software options, many institutions have chosen to develop their own system, either in-house or by hiring a programmer. They express several reasons for their decision: lack of solid, identifiable alternatives for purchase, lack of a fully integrated system, a preference for a system fully compatible with their operations, and concern about ability to improve and modify the system to meet future needs.

These are legitimate concerns, particularly if an institution is considering a small custom module to complement its other software, e.g., a module to track training course attendance. However, if an institution is considering development of a fairly large system or an integrated system to avoid having to manage multiple, incompatible MIS modules, they should proceed with extreme caution. The experience of the vast majority of institutions that have chosen this route is that the results are far from their original expectations. Development of a small, basic application in Microsoft Excel or Access can sometimes be done even without the aid of experienced programmers. However, developing a more complex application, such as a loan portfolio module or a fully integrated system is far more challenging. In a disturbing number of cases, the software simply fails to work and the effort is abandoned. We would not be acting responsibly if we did not strongly caution you about pursuing this option, as tempting as it may initially seem.

What can go wrong? Everything! In virtually all cases, the software development takes far, far longer than planned, generally two to three times longer. Costs also vastly exceed original estimates as the development period drags on.

Whereas full functionality is often a primary motivator in choosing this route, much of the planned functionality never makes it into the custom system. Instead, energy is directed into finalizing and debugging core, or basic, functionality. As the budget dwindles, the additional features get postponed. Systems get implemented once the core features are in place. Attention then turns to debugging, a process which can in itself be quite time consuming as well as frustrating for operational staff, which are relying on an “unreliable” system to give them accurate information. Reports are often the last aspect of the system to be developed. In the meantime, staff members learn to manually generate the reports they need.

In most cases, software vendors have invested years of effort in developing and refining their products, at great cost. That cost is then distributed over a (hopefully) large user base. The result should be a better system at a lower cost than what an institution can develop on its own. However, many times managers look at existing systems and think they can do better. They talk to programmers that claim it wouldn't be that hard to develop a system. Initial time and cost estimates come in looking very promising. But, things become more complex as the development process gets underway, for many of the same reasons cited throughout this Manual, including: poor definition of information

needs, lack of clear and consistent operational procedures and policies, an inability to recognize how many exceptions there are to the norm, and not understanding how difficult it is to develop software that adequately handles those many exceptions.

One contributing factor is that programmers often are inexperienced in the specifics of the microenterprise institution's business. They may be developing a loan portfolio system without really understanding how interest is calculated and how delinquency is measured. Or, they assume that since they have written an inventory control system for an auto parts store, they can easily manage a class scheduling system. Often they want to leap immediately into system development. But, if the system is not well conceived beforehand, the development can take much longer with major elements of the system needing to be reworked. And, in the worst cases (unfortunately, not all that uncommon) the system may never work properly. The importance of following a systematic process of needs assessment and system design cannot be overemphasized.

Some institutions prefer custom systems in order to have control of the process and the final product. A custom system can be developed in-house by staff of the institution, which insures access to the source code and provision of technical support, although the ongoing costs of that support may be high. Or, development can be contracted out to an independent firm, in which case ownership of the source code and the cost and reliability of technical support need to be carefully negotiated. With control of the source code, the institution has much more control over future alterations and additions, or critical bug fixes.

Despite all of the dire warnings presented thus far in this section, we need to include one final strong cautionary note before closing. In our research, we heard plans from a significant number of institutions that were developing or had developed custom software at significant expense, in the hopes of being able to then market that software to other institutions and recoup some of their investment. In only one case that we know of did this actually work as intended. In all other cases, the institutions decided that the software would require significant additional investment before it would be worthy of distribution to other institutions. In addition, many also commented that they realized their strength and expertise were in microenterprise assistance — not in software development and support — and that any attempt to add software marketing to their institution would distract from their core business. Several institutions were considering fall-back plans of selling their custom software to an existing software company, so that that company could expand upon their work and make it available to others.