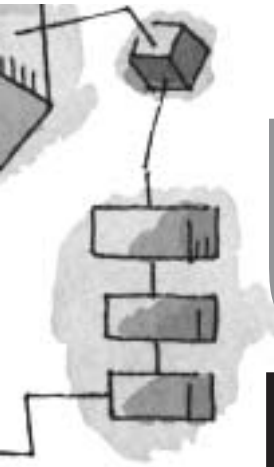


UniverSIS: Flexible System, Easy To Change

Designing flexible information systems saves time and money, now and in the future

By **Robert Miller, Bruce Johnson,** and **Walter W. Woolfolk**

Typical information systems are inflexible, not easily accommodating changes in business requirements. This article identifies characteristics of a flexible information system, using UniverSIS — a student information system developed at University of Cincinnati (UC) — to illustrate principles of flexible design. (See the sidebar "The UniverSIS Project" for background about UC and the project.) We demonstrate the benefits of this approach by discussing specific features developed to support the admissions application process. These benefits include increased business control over system behavior, reuse of system solutions, and reduced involvement of technical staff as business requirements change. We examine the impact of flexible design on business and technical staff, and suggest specific steps that developers can take to achieve system flexibility.

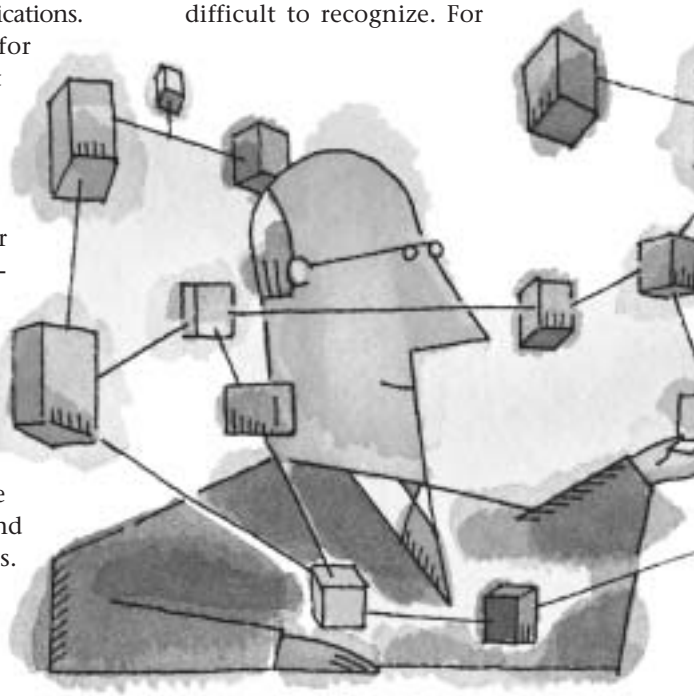
What Is Flexibility?

Flexibility in a system means the ability to accommodate a change in business requirements with a minimum of modification to system components. Since some system components cost more to modify than others, builders of flexible systems focus on achieving a design that allows subsequent modifications to be limited as much as possible to the least-cost-effect components. The ideal system would be one in which changes in system behavior result entirely from business staff modifying data values rather than from information technology (IT) staff modifying file definitions or program code. In those cases requiring coding changes, they are restricted to local modifications that do not result in a chain reaction of compensating modifications.

Developers can also look for opportunities to leverage effort by designing reusable functions. Rather than addressing each business requirement as a separate problem, they look for patterns of behavior and for opportunities to establish general solutions to types of problems.

A stable data structure forms the most critical element of a flexible system. Design of UniverSIS began with a careful analysis of the business data requirements and design of the supporting files. Recognizing that business requirements are constantly changing, the developers designed the system to accommodate potential future requirements, not just current needs. The goal was a system that would support the most complex real-world situation, even if such a situation did not reflect current reality. In pursuit of this goal, developers kept a few key questions in mind while modeling the data:

- *What implicit assumptions are being made?* Such assumptions are often difficult to recognize. For



example, UC's legacy systems contained an implicit assumption that the number of colleges within the university was fixed. When UC added a new college a few years ago, thousands of hours of IT staff effort were required to accommodate the change. In UniverSIS, addition of a new college involves only the addition of data values by business staff.

- *What meaning is attached to record identifiers?* Such meaning is often so ingrained in the culture of the institution that it is difficult to eliminate. In its legacy system, UC embedded information about the college, department, and subject matter in the course number. Periodic renumbering of courses required significant system changes. In UniverSIS, the course number is simply an identifier. Separate data elements record the information previously embedded in the course number.
- *What will change?* While there are practical limits to accommodating all possible change, assuming anything can change is a good starting point. The rule of thumb is that change should be accommodated through changes in data whenever possible.

The UC Application Process

UniverSIS includes features for admissions application processing. UC's application process resembles that of most universities, involving three steps:

1. applicant submits an application,
2. applicant provides the university with required credentials, and
3. university reviews the applicant's credentials and informs applicant of its decision.

In the past, the application process at UC followed this scenario. Applications were received and assigned to an admissions counselor who monitored them until all credentials had been received. At that point, the counselor evaluated each applicant's credentials, made a decision, and recorded the decision in the system. Although the results were recorded in electronic form, the review process itself was entirely manual.

Based on their academic credentials, applicants for a particular academic program fell into one of three groups:

- clearly admissible
- clearly not admissible
- admissible but in competition with others

Applicants who were clearly admis-

The UniverSIS Project

University of Cincinnati Profile

- Founded 1819
- Second-largest state university in Ohio
- 35,000 students
- 5,000 Faculty and Instructors
- 5 campuses
- 17 colleges and schools, including Law and Medicine
- 250 undergraduate programs/majors
- 150 graduate programs/majors

The UniverSIS Project

In the mid-1990s, the University of Cincinnati (UC) faced the need to retire its legacy student information system. The system was in fact a set of individual systems that supported specific business processes and were used exclusively by individual business units. The systems interfaced via batch feeds of data. A significant amount of data was stored redundantly. Most of the systems were written in PL/I using VSAM files for data storage; one major component used an IMS database.

UC decided to replace these systems with a single, integrated student information system. When existing vendor product suites were found to meet only 60 to 70 percent of the university's needs, UC decided to build the system. Most of the system was custom-developed, but vendor products were used for specific purposes. The design of the system emphasized principles of flexibility.

System Scope

- Demographics: personal information
- Curriculum: colleges, courses, academic programs, classes, classroom scheduling — Series 25 (CollegeNet)
- Admissions: recruitment and application processing
- Financial Aid: Financier (Wolffpack)
- Registration/Student Records (enrollment processes and academic records)
- Student Accounts (fee assessments and billing)
- Degree Audit (DARS [Miami University – Ohio])

UniverSIS was implemented on schedule, within budget, and with all mission-critical functionality in place. Implementation began 18 months after the start of the project and continued in stages. Since development and implementation overlapped, construction of temporary interfaces proved necessary. The existing legacy systems were retired on the fourth anniversary of the project's start. Included in the implementation effort was conversion of the academic records of the 250,000+ students for whom electronic records existed. The UniverSIS programming was done by fewer than six people.

UniverSIS is currently in its third year of operation. As expected, business requirements change frequently, and the demand for enhancements is continuous. As hoped, the flexible design of the system has allowed many enhancements to be accommodated with little or no IT intervention.

sible could be offered admission immediately. Those who were clearly not admissible could be denied admission immediately. Review of those in the third category was postponed until the end of the application process.

The Problem

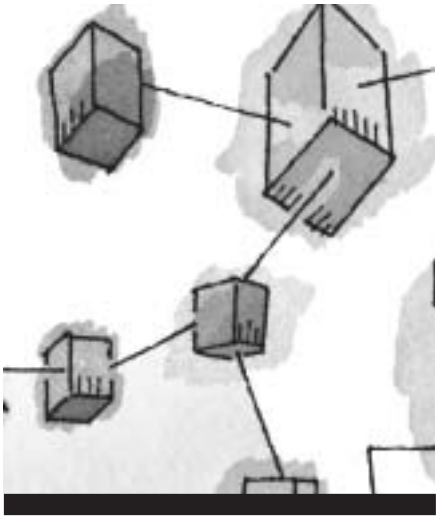
UC competes for those high school students that all universities try to attract — good students who fall in the "clearly admissible" category. Success in recruiting such students can depend on reducing the time required to notify them of their acceptance. A trend among universities is "instant admission." As reported in the *Christian Science Monitor*, "...a trend toward rendering speedy decisions in person is turning on its head the long wait for a 'fat envelope' in the mail."[1]

Processing an application involves business rules and actions based on these rules:

1. Determine, for each academic program, which credentials an applicant must provide.
2. Determine, for an individual applicant, whether all required credentials have been received.
3. Determine the conditions under which an applicant is clearly admissible to an academic program.
4. Determine whether an individual applicant is admissible.

Points 1 and 3 refer to business rules. Credential requirements and admission requirements are typically stable, but may change from year to year. Experienced admissions counselors remember the requirements. New staff members typically need to refer regularly to a procedure manual.

Points 2 and 4 refer to actions taken on an individual application. These actions occur only when the admissions counselor becomes aware of the need for intervention. Any delay in that awareness translates to a delay in reviewing the application. In addition, during peak application review periods, even when a counselor becomes aware of a completed application, time and the volume of applications already in the queue may delay review of that application.



It was critical that the solutions to both the awareness and the evaluation problems provide both automation and business control over change.

In an environment where instant admission is the trend, the business problem was to expedite the processing of applications, particularly those of students who were clearly admissible. The development problem was to solve the business problem with a design that would easily accommodate future changes.

Improving the Process, Flexibly

In designing the system to streamline the process of reviewing applications, the developers identified two independent problems: awareness and evaluation. The awareness problem involves identifying required credentials per academic program, and recognizing and recording the fact that an application is complete. The evaluation problem consists of identifying admission requirements per academic program, determining that an applicant satisfies those requirements, and recording the decision.

It was critical that the solutions to both the awareness and the evaluation problems provide both automation and business control over change. UC offers more than two hundred undergraduate programs, each with potentially different credential requirements and different admission standards. Each applicant's situation is potentially unique and could require exceptions to the general rules. A solution that required technical staff to be involved with each change in credential requirements or admission requirements would have been doomed to failure.

Flexible Awareness

To solve the awareness problem, the developers designed mechanisms to let administrative staff define required credentials and record receipt of an individual's credentials.

Define Credentials. Credentials take a variety of forms: transcripts, test scores, letters of recommendation, auditions, and portfolios, for example. Business staff define and maintain credentials. A credential definition may specify ratings where appropriate. For example, the College of Music may define a credential of "audition" with ratings from "poor" to "excellent."

Define Credential Requirements. Business staff define the standard credential requirements of each academic program. These requirements include the following options:

- *Levels of specificity.* A credential may be required of all applicants or limited to specific colleges, programs, or majors. For example, "application fee" might be a general requirement, but "audition" might be a requirement specific to applicants to music or dance programs.
- *Count*. Multiple credentials of the same type may be required, such as letters of recommendation.
- *Verification.* Upon receipt of a credential, the content and validity might need to be verified separately (letter of recommendation, transcript).
- *Alternatives.* A credential requirement can be met in an alternative way, as when one test result may be substituted for another.

Record Receipt of Credentials. Business staff record receipt of an individual's credential as a document. Multiple versions of the same document might be

recorded for an individual. In addition to the credential information itself, business staff might record expiration date/term and free-form notes.

Achieving the objectives. The solution UC developed addresses all of these considerations. When an individual applies for admission to an academic program, the system retrieves the set of credential requirements pertinent to that program. These standard requirements are copied to the individual's application profile. The admissions counselor can customize the applicant's requirements to accommodate any special circumstances, such as home schooling, foreign schooling, or need for English proficiency certification. The list of required credentials functions as a checklist.

As an applicant provides credentials, receipt of each credential is recorded as a document. The system automatically matches the individual's credential requirements — the checklist — against the documents received. When each item has been checked off, the system changes the status of the application to "Ready for Decision."

The credential-related mechanisms described above provide a high degree of flexibility. No IT intervention is required to define new credentials or to maintain credential requirements. Unique needs of individual applicants — exceptions to the rules — can be accommodated within the system, again without IT intervention.

It is important to note that the concept of documenting credentials did not come directly from the admissions staff. They expressed their requirements very specifically: application fee, letter of recommendation, test score, high school transcript, and so on. The UniverSIS developers, thinking in terms of flexibility — accommodating change — identified a general category into which these specific items could be grouped and identified a pattern of data elements needed to support members of that category. That pattern has proven to be consistent with the pattern of documenting credentials in general. The developers have been able to reuse the credential/document mechanism for other business purposes. For example, it is used to record that an incoming student has attended freshman orientation and when a student's address information was last verified.

In addition, the mechanism has been enhanced over time. The expiration date/term and free-form notes were later additions, prompted by the experience of the business staff. These modifications did require intervention by the IT staff, but the modifications were localized.

Flexible Evaluation

The solution of the evaluation problem allows administrative staff to define admission requirements and provides a mechanism that lets the system evaluate credentials automatically. The first step was to identify the factors that the colleges consider in making their decisions. At the undergraduate level, the number of factors was small:

- High school GPA
- Percentage rank in high school class
- Ohio High School Proficiency Test rating
- Best ACT or SAT math score
- Best composite ACT score or SAT total score
- High school core courses completed

The developers designed a mechanism that lets administrative staff define requirements in the form of logical statements about the factors. Requirements are attached to the appropriate academic program and can be as simple or as complex as necessary. A requirement has date-sensitive versions. This permits maintaining a history of requirements. It also allows administrative staff to prepare and test new versions of requirements without affecting current processes. IT staff get involved only when a new factor is defined.

A fictitious example using the 22BBA Accounting program (see Figure 1) in the College of Business illustrates the design. Any applicant for this program between Terms 01W and 05U who meets the requirement called ACCT will automatically receive an admission decision of "OF" (Offered). In other words, if the applicant meets the criteria found in requirement ACCT, the system will automatically admit the applicant to the 22BBA ACCT program.

Figure 2 shows that the requirement called ACCT specifies two conditions. If either condition is satisfied, the requirement has been met.

The first condition is shown in Figure 3. The second condition would be the same except that SAT scores are specified instead of ACT scores.

A mechanism called Evaluator compares the applicant's credentials to the criteria specified in the requirements. This mechanism consists of two components: retrieval and analysis. The first component retrieves an individual's data for the specified factor. The component consists of a set of modules, each corresponding to a decision factor. For exam-

Figure 1

Example of Decision Rules

```
==========================================================

                Maintain Admission Decision Rules

                *Academic Program:  22BBA _____  Bachelor of Business
                *Academic Area...:  ACCT _____  Accounting

    *Decision Code: OF Offered
    *Begin Term...:  01W _____ Winter  Quarter  2000–01
    *End Term.....:  05U _____ Summer Quarter  2004–05

    *Requirement: ACCT_____ 09 12 2000 CBA ACCT Criteria
==========================================================
```

Figure 2

Conditions for "ACCT" Requirement

Maintain Evaluator Requirement

Requirement: ACCT_____
Version Date: 09 12 2000

Description: CBA ACCT Criteria_____

*Conditions (meeting one condition satisfies requirement)
1 CBAACCT1_____ CBA Criteria (with ACT)
2 CBAACCT2_____ CBA Criteria (with SAT)

Figure 3

ACT Condition for "ACCT" Requirement

Maintain Evaluator Condition

Condition ID: CBAACCT1_____

Description: CBA Criteria (with ACT)_____

	*Selection Element	*OP	*Value
	HS-GPA	GE	3.000
AND	HS-CLASS-RANK-PCT	LE	25
AND	BEST-ONGPT-RATING	EQ	E
or		EQ	P
AND	ACT-BEST-MATH	GE	22.00
AND	ACT-BEST-COMPOSITE	GE	22.00
AND	HS-CORE-DEFICIENCY	EQ	0.000

ple, one module retrieves an individual's high school GPA, another retrieves best ACT math score, and so on. Once written, these modules are available as building blocks for an unlimited number of conditions. A factor can be data retrieved directly from the individual's record, or it can be data derived programmatically from a combination of records, such as ACT-BEST-MATH.

The second component analyzes the applicant's data, comparing it against the requirement, factor by factor, until it can be determined whether the applicant has satisfied all criteria. The conditions described earlier are stored in the form of a logical binary tree. The mechanism that performs the analysis parses the binary tree, plugging in the individual's data values appropriately, and returns either a "true" or "false" value. When all criteria have been satisfied — the result is true — the system records the decision automatically on the individual's application.

The business objective was to design a mechanism that would allow the system to admit qualified applicants automatically. The technical objective was to meet the business objective with a design that was flexible. Both objectives were achieved. Business staff in the admissions office maintain the requirements without IT intervention.

The effort to develop the Evaluator tool was significant and complex. However, the payoff justifies the work. The pattern of evaluating an individual's data profile against a set of criteria has proven to be one that repeats itself in many business processes. For example, the Student Accounts office maintains all rules for assessment of tuition and fees, and the Registrar's office maintains rules for registration restrictions. As rules change, business staff make the necessary system adjustments. UniverSIS provides a tool that allows business staff to test requirements online. Staff can view the result of each step of the analysis to pinpoint any difficulties with the logic.

A reusable tool such as the Evaluator, used for multiple purposes in multiple business units, presents potential risks. Without some control mechanism, staff in one business unit might inadvertently make changes that affect another unit's processes. To address this issue, UniverSIS segments Evaluator requirements by type, ensuring that only authorized staff members maintain them. Every business process that uses the Evaluator is tied to a requirement type.

The Improved UC Application Process

Solution of the awareness problem has improved application processing significantly. Admissions counselors review applications "Ready for Decision" and take action only on those applications for which the system was unable to make an automated decision. The system functions as each counselor's self-maintaining "to do" list. Counselors no longer review applications for completeness. The system does that. This feature is especially valuable for new counselors who are unfamiliar with the requirements of each program.

UC chose to implement automated evaluation of applications as a batch process. Each night, the system identifies applications that have reached Ready for Decision status and evaluates them. For any applicant who meets the requirements, an offer of admission is recorded and an offer letter is automatically generated.

With the automated evaluation mechanism in place, the UC admissions office can mail an offer letter to a clearly admissible applicant within 48 hours of the

time the applicant's documents have been recorded, without any intervention by an admissions counselor. Even when decisions cannot be made automatically, the typical turnaround time for an application has been reduced significantly. This was evident from a campus survey in which the UC characteristic that was rated highest by the students was "timeliness of your acceptance."

The automated decision process was initially implemented for a limited number of academic programs. After one year, analysis of the results indicates that approximately 12 percent of the offers of admission were made through the automated decision process. Clearly qualified applicants were offered admission, and also confirmed their acceptance, earlier than in years past. The criteria for "clearly admissible" were set conservatively, to avoid any potential oversubscribing of the programs. The list of academic programs for which automated offers can be made has been expanded, and the "clearly admissible" bar is likely to be lowered as well. The goal is to have 70 percent of offers made automatically by the system. The expectation is that the admissions staff will make any system adjustments required to achieve that goal without IT staff intervention.

Staff Impact of Flexible Design

A flexible system design changes the roles of the developers and operators of that system. In the development of UniverSIS, the key business roles were played by the individuals who could explain current processes and also analyze and clarify the underlying business rules and requirements. Similarly, the developers had to think beyond the requirements of the current processes, understanding the underlying rules and requirements well enough to design solutions that would easily accommodate changes in those rules and requirements.

In analyzing business requirements, the developers regularly posed two questions: "Why do you do that?" and "Do the rules ever change?" The first question was often met with puzzlement, particularly when no one was sure of the answer. If the answer was "because we've always done it that way" or "because the system won't let me...," more analysis was needed. The answer often traced back to the design of the legacy system, in which changes almost always required modifications to program code. Business staff had found ways to work around the system's limitations to accommodate changes in business requirements, often by recording odd values in fields designed for a completely different purpose — "beware the asterisk." Over the course of the project, the business staff began to recognize that workarounds occurred because often it was too difficult to change the system.

The experience of long-time business staff was particularly important in answering the second question. They knew that the answer to whether the rules ever change was "Yes!" They knew which business requirements had been stable over time and which changed on a regular basis.

The good news is that business staff have come to see the value of flexibility, particularly in putting direct control over change in their hands. The bad news, or so it sometimes seems, is that the business staff have active responsibility for maintaining business rules as data, which involves a lot of work. Before a system feature can be implemented, the rules have to be in place. In addition, because formerly separate modules are now integrated, staff in one business unit have to communicate more closely with their colleagues in other units. The impact of change can be far reaching and may be felt immediately in unexpected places. Ultimately, however, this is good, requiring the various business units of the university to follow a consistent set of business rules and to explore the potential impacts of change.

In addition, the nature of system testing has changed. Changes to business rules ideally do not involve changes in program code. Testing has become, therefore, not only a significant responsibility of business staff but an activity for which they are almost solely responsible.

The change for the IT staff has been equally significant. Just as the front-line business staff traditionally concern themselves with individual processes, programming staff traditionally concern themselves with coding programs to support those processes. Because the flexible UniverSIS design allows business staff to exercise control over much of the system's behavior, programmers do not spend their time on the traditional task of recoding programs when business rules change. Instead, they design and code solutions that frequently work indirectly. Programs often don't contain the actual business rules. Instead, they contain the logic to look up the rules.

Furthermore, since the developers have isolated many reusable pieces of complex logic, programs are often assemblies of references to other specialized programs. This approach takes some getting used to. It also requires good communication and documentation. The independent-minded programmer who would rather do things his or her own way presents a significant barrier to the success of such an approach. It is critical that management take an active role in establishing and enforcing adherence to standards, including the concept of reuse. While design and construction of flexible solutions initially may take longer than task-specific solutions, over time flexible solutions lead to a significant reduction in routine system maintenance and an increase in the amount of time available to IT staff for new projects.

Designing Flexible Systems

An information system that incorporates effective use of flexible design embodies a set of technical principles. It also reflects an understanding of the attitudes of both the IT developers and business staff who will be involved in the project.

Technical Guidelines

To design a system that is flexible, developers should

- *Conceptualize the system as a whole dynamic entity, not as a set of connected pieces*. The goal must be an integrated system with a shared information structure, not one in which independent modules are merely interfaced, passing information back and forth between one another.

- *Define and enforce consistent design standards for both technical requirements and the user interface.* A data dictionary is essential to defining standard fields and relationships among entities. Use of program models and templates facilitates coding. A code generator, in conjunction with the active data dictionary, significantly reduces the amount of manual coding required. Technical managers must ensure that standards are documented and enforced.
- *Separate the user interface, business rules, and data (n-tiered architecture).* This permits use of a new interface medium, such as the World Wide Web, without change to business logic or data storage. It's not just the business that may change — the world is also changing.
- *Identify logical data entities and entity types, and maintain them as individual objects.* The object maintenance program performs all maintenance of entity records. All business rules are enforced through that program, ensuring that those rules are applied consistently. Variations of entity types must considered in the design. For example, a Person may be an Applicant, a Student, an Employee, and so on.
- *Eliminate meaning from record keys.* Keys are simply pointers to unique records in the database. For user comfort, naming conventions may be needed, but meaning must not be embedded in the name, and enforcement of business rules must not use a naming convention.
- *Store business rules as data.* Changes in business rules are the most likely trigger of system modifications. Whenever possible, business rules should be maintained by business staff, reducing the need for file and program maintenance by technical staff.
- *Code reusable logic in callable routines.* Documenting such routines is an essential aspect of technical management.
- *Develop general-purpose, reusable business processes whenever possible.* Separate business processes often follow a common pattern. Develop general-purpose, rather than purpose-specific, processing tools.

It is essential that developers recognize inflexible design and correct it. At times the developers of UniverSIS missed requirement details or made incorrect assumptions about what could change. On such occasions, they tore apart what they had done and redesigned it. Though painful, the effort of correcting the design to make it more flexible was worthwhile in the long run. The tools used to develop the system made the effort much less painful than it might have been. (The sidebar "UniverSIS Tools and Techniques" summarizes the technical tools and environment.)

UniverSIS Tools and Techniques

UniverSIS was built using the Software AG development tool set. It runs in a mainframe environment.

- ADABAS is the DBMS.
- NATURAL is the fourth-generation language.
- PREDICT is the active data dictionary.
- CONSTRUCT is the code generator.

At the outset, project stakeholders agreed to use standard CONSTRUCT models and a consistent look-and-feel for presentation of information on screens. This facilitated both technical development and training of business staff. The system behaves in the same way regardless of which component is being used.

UniverSIS uses standard CONSTRUCT object models for maintenance of data. All data is stored in ADABAS, and maintenance of all records is managed through object subprograms, which enforce the stored business rules. The developers used PREDICT extensively within UniverSIS, both to provide documentation and to enforce standards. The system was developed with mainframe screens as the user interface, but that interface contains no business logic. This approach permits easy substitution of a different interface, such as the World Wide Web.

Participant Attitudes

While the technical environment and tools play important roles in the design of flexible systems, no less important are the attitudes of project participants, who must have confidence in the concept of flexible design. The lead developers of UniverSIS had such confidence, having been involved in designing other flexible systems. The confidence of the UC business staff evolved over the course of the project, as the effectiveness and benefits of flexibility became clear.

To establish the needed confidence, an organization might want to begin with a demonstration project involving a significant but not mission-critical system. Those involved in the development of this system should agree to the following behavioral guidelines:

- *Be committed.* Everyone, from the project sponsors to the technical staff, must be committed to the goal of flexibility. Business and technical staff must ask two questions: "How might this change in the future?" and "How can that change be accommodated through changes in staff-maintained business rules?"
- *Be patient.* It may take longer to develop a flexible solution than a specific solution until developers have mastered the tools and approach.
- *Be persistent.* Under pressure, developers are tempted to go with a quick-and-dirty (inflexible) solution, with the intention of going back later to do it right. Resist this temptation. Eventually, the time saved with the quick-and-dirty solution is lost many times over to the effort required to maintain it. If a design is recognized as insufficiently flexible, redesign it. Everyone will learn from the experience.
- *Think in a new way.* This is essential and rewarding at the same time. The chal-

lenge of devising programs that adjust their behavior based on changes in staff-maintained business rules can release pent-up creativity. This is a far cry from the typical hostage situation where IT professionals have no time, no resources, and no charter to imagine anything new.

- *Develop general-purpose, reusable business processes whenever possible.* Separate business processes often follow a common pattern. Look for that pattern. Develop pattern-oriented processing tools rather than purpose-specific processing tools. Ask "How else might we use this tool?"

Having developed one flexible system that stands up well to changes in business requirements, both business and technical staff will gain confidence in the concept of flexibility and in their ability to design other flexible systems.

Flexibility in Practice

The following brief examples illustrate the positive impact of flexible design. In an inflexible system, the required modifications would have been costly, time-consuming, disruptive, or not done at all.

On-Campus Transfer

In the past, students who wanted to transfer from one college to another within UC were required to go through a procedure that involved paper forms and manual processes. Adding a single code to a staff-controlled table, for "on-campus transfer application," allowed the UniverSIS admissions module to be used for this new purpose. All necessary system modifications were performed by business staff modifying data. The paper process has been eliminated.

Tuition and Fees

The university was faced with a sudden and unexpected mid-year drop in financial support from the state. In the middle of autumn quarter, a decision was made to increase all tuition and fees as of winter quarter. All rate adjustments were performed by business staff in less than two business days. No IT intervention was required.

Correspondence Control

UniverSIS provides a correspondence management feature that can be used to generate personalized letters. The feature was initially developed to produce offer letters for the admissions office. Without any changes to program code, the Collections office was able to use the same feature to generate personalized letters to students with delinquent accounts. All correspondence-related work was performed by business staff modifying values in UniverSIS. The only change that involved technical staff was setting up a new batch job.

Financial Oversight for Athletes

UC, like other large universities, is frequently under scrutiny by the NCAA to make sure athletes aren't getting money they should not be getting. The athletic director ordered that the system be modified — immediately — to block refund checks for athletes until financial aid award requirements had been verified. Only 12 hours of programmer time were required to develop the required modification, which includes automated block of refund checks and automated e-mail notification of the Athletics department when a change in an athlete's enrollment or financial aid status occurs. The developers were able to adapt and reuse existing features, significantly reducing the need for custom coding.

Results Justify the Effort

Flexible features form the tool set for the business staff. As they learn the tools and gain confidence in them, they want to use them for every possible purpose. As new business requirements have emerged, business staff recognize how much direct control they have and appreciate the value of reuse and consistency. There have been situations in which the business staff accommodated a completely new requirement in minutes with an existing feature simply through modification of data. The UniverSIS developers have been surprised, and delighted, at the ways in which the system features are being used. In many cases, technical staff function only as consultants, helping the business staff identify which of the available tools would be most appropriate, but making no changes to programs or files.

The success of UniverSIS, or any flexible system, will be judged by how well it accommodates current and future business requirements, and how business staff take ownership of the system. At UC, the business staff own UniverSIS. The benefits of flexible design have been clearly demonstrated. *e*

Endnote

1. "Why Wait? Colleges Try 'Instant Admission'," *Christian Science Monitor*, December 11, 2001.

Robert Miller (robert.miller@uc.edu) is Director of Information Technology System Services at the University of Cincinnati in Cincinnati, Ohio. Bruce Johnson (mountainbruce@worldnet.att.net) is retired from a position as associate professor at Xavier University. Walter W. Woolfolk (woolfolk@fuse.net) is Information Systems Director for Taylor Distributing in Cincinnati, Ohio.

Suggested Reading

Johnson, Bruce, Walter W. Woolfolk, and Peter Ligezinski, "Counter-intuitive Management of Information Systems Technology," *Business Horizons*, March–April 1999.

Johnson, Bruce, and Walter W. Woolfolk, "Generic Entity Clouds: A Stable Information Structure for Flexible Computer Systems," *Systems Development Management*, October 2001.

Woolfolk, Walter W., Peter Ligezinski, and Bruce Johnson, "The Problem of the Dynamic Organization and the Static System: Principles and Techniques for Achieving Flexibility," *Proceedings of the 29th Annual Hawaii International Conference on Systems Sciences* (HICSS-29), Vol. III, January 1996, p. 482.

Woolfolk. Walter W., and Bruce Johnson "Information-Free Identifiers — A Key to Flexible Information Systems," *Data Base Management*, Part I July 2001, Part II August 2001.