**Developments in Business Simulation & Experiential Exercises, Volume 9, 1982**

# SYSTEMS LEARNING SEQUENCE: AN EXPERIENTIAL COURSE MODULE FOR MANAGEMENT INFORMATION SYSTEMS

Philipp R. Heer, OCLC
Nancy Sampson and Haviland Wright, University of Denver

## ABSTRACT

This paper presents a pedagogic instrument, the Systems Learning Sequence (SLS), which provides students with an opportunity to operate and analyze a simple information system. The educational results of the SLS are evaluated using student audit reports of the SLS information system. The evaluation suggests that the SLS experience is an effective tool for communicating management information systems concepts. The SLS presumes no programming experience and functions under the concept that it is not necessary to be a skilled programmer to comprehend MIS technology.

## SYSTEMS LEARNING SEQUENCE

Programs in Management Information Systems necessarily include technical topics in computer technology and programming as well as managerial topics such as human information processing and implementation planning. The integration of these topics in introductory MIS courses is a formidable challenge. This paper presents a pedagogic instrument, the Systems Learning Sequence (SLS), which provides students with an opportunity to operate and analyze a simple, but complete, information system. The SLS reflects the fact that while few business school graduates become programmers or computer operators, many will be users and evaluators of information systems. The SLS has been used in elective and noncredit courses on MIS and is intended to be included in an MBA core course to meet AACSB standards relating to Information Systems. The purpose of the paper is to outline and explain the SLS and to evaluate the educational results of using the SLS as indicated by student audits of the SLS information system.

The SLS was designed to provide integrated, hands-on exposure to the more technical aspects of management information systems, with the expectation that after the SLS students would be better prepared to communicate with systems designers as users. This approach to management education complements Couger's studies[1] indicating low social needs (and therefore communication skills) of programmers and analysts. It may be more appropriate to train managers for communication with MIS personnel than to attempt redirection of MIS personnel. Specifically the SLS prepares students to work with analysts and/or programmers in troubleshooting existing systems and in better expressing requirements for new systems.

The philosophy underlying the SLS is that a student can develop a sense of systems technology, requirements, and limitations relatively quickly by working with a simple but complete computer-based information system. The sequence can be completed in eight to ten hours of class time. Programs in the SLS were developed using COBOL because of its predominant use in existing administrative systems. Furthermore it was presumed that a basic understanding of the language and some familiarity with available computer hardware is essential before actually using and operating an information system. Therefore, the SLS contains two modules, the COBOL Learning Sequence (CLS) and the Information Systems Learning Sequence (ISLS) . The ISLS was designed to be representative of a functioning system, complete with errors, logical inconsistencies and a documentation package, typos included, which has not been updated to reflect the current status of the system. Programs were written with minimal instructions in a straight-forward manner to reduce confusion for students trying to read them. This philosophy leads to the less desirable aspect of a system that gets something done but is not very user friendly.

### COBOL Learning Sequence

The CLS module is presented first in the SLS and consists of a four phase COBOL program development. Beginning with a simple program utilizing only internal processing, the CLS develops a program with input/output files and fundamental data control The CLS is designed to teach students to distinguish between language features, operating system features, and hardware features. By the completion of phase four students have had considerable experience with the COBOL language and know the difference between compiler errors and logical errors. The first program of the sequence utilizes only internal processing and produces a single line of printed output. Students are asked to change one line of code in the program before running it. The program is then stored for further modification. This program is used to introduce:

1) COBOL sections;
2) output;
3) operating system vs. applications messages; and
4) source program-compilation-object program-execution.

Students are then provided with the code for a more advanced program and asked to modify the original program to match the new code. Students continue modifying the program, sequentially adding features such as an input file, a logical loop, a stored output file, internal data storage, and a mathematical operation. During the successive program modifications the student learns to:

1) differentiate between data and procedure;
2) read elementary COBOL instructions;
3) correct compiler errors; and
4) add input/output files with required data specification and program logic.

During the CLS most students also begin to:

1) work out logical errors introduced during program modification; and
2) recognize the detailed thought processes involved in programming.

At completion of the CLS the students have written, checked-out, and executed the first program of the Information Systems Learning Sequence, and are ready to begin the next module.

[1] J. Daniel Couger and Robert A. Zawacki, Datamation, September 8, 1978, "What Motivates DP Professionals?"

Information Systems Learning Sequence

The ISLS module introduces students to a batch processing system which uses individual programs to accept data, to sequence data, to edit data, and to build and maintain a master file. This mini-system is similar to many business systems currently in operation. Students are provided with a system documentation package and form groups in which individuals adopt the roles of systems analyst, programmer, operator, and user. The four programs in the mini-system are presented one at a time to the groups until the system has been cycled once. Groups cycle the system three or four times in order to become familiar with all its features. The student groups, acting as system audit teams, present their findings, and generate recommendations for system improvements and documentation updates.

At the beginning of the ISLS students are provided a documentation package and access to the first program of the system. In discussions of the assigned task students are provided no other cues. They are reassured that they know enough data processing to perform an audit of the system and they are encouraged to run the first program when they have decided what it does. They are also told there is no existing master file. The first program accepts data, builds a work file, and produces a log of the input.

The second program of the ISLS is a sort which provides the basis for a discussion of batch versus online systems requirements. Students are asked to verify that their data is in fact in the correct order.

The third program is an edit which introduces the students to the concept and limitations of data verification. Student auditing of the edit phase is intended to force precise desk checking as the student must cross-reference the program with the input file, the edit report, the output file, and the system documentation. Once this phase is completed, the data and students are ready for file maintenance procedures.

File maintenance is performed in the fourth program. The file maintenance is transaction driven to emphasize the need to provide capabilities to add records, change fields, and delete records from a masterfile. Additional desk checking is required to validate the new masterfile by comparing the old masterfile, the transaction input and reported information. Follow-up discussions relate to systems backup and requirements to provide an adequate audit trail. Before students cycle the entire system, they are provided with detailed operator instructions for saving appropriate files and developing a historical backup.

As the students cycle the system they are encouraged to check everything they think it does, should do, and should not do. They are emphatically asked not to make any program changes, but they may run as much data and as many transactions through the system as is necessary to develop a complete set of findings and recommendations.

The ISLS module teaches students the essential parts of a computer-based information system and provides them with basic information necessary for effective communication with information systems personnel. The developed basis for effective communication includes a recognition of the attention which must be paid to detail and the high degree of coordination required between user, programmer-analyst, documentation, and actual operation of the system. Acting as audit teams students gain experience in identifying systems problems, sources of systems problems, and possible means of problem correction. The module also provides the instructor with opportunities to comment on systems security.

Educational Effectiveness

The primary student outputs from the SLS are the mini-system audit reports generated by the groups in the ISLS module. These reports reflect the information conveyed to the students during the SLS and the developed ability of students to use this information in an integrated analysis of an information system. Therefore these reports are used as data for evaluating the educational effectiveness of the SLS. Audit reports by groups of students in two graduate elective courses and four mid-career, non-credit courses representing 110 students have been assessed and compared to the learning objectives of the SLS as an indicator of educational effectiveness. Within each class student groups are randomly chosen to present one of the following areas: data acceptance, edit, file maintenance, documentation and overall system. Of particular concern in the analysis is the core of conveyed information as indicated by the intersection of audit report items and the correspondence of this core to the design objectives of the SLS.

Results

Table I summarizes major areas of concern. Learning objectives include the ability to identify problems and to correctly identify the possible cause or source of the problem based on the students' understanding of both the CLS and ISLS. While student groups create their own learning experience the quality of which is determined by their thoroughness and the number of transactions entered into the system, certain problem areas are absolutely critical to an audit. If these problems are not discovered it will be impossible to ever successfully rely on the results of the system.

As reflected in Table I, student groups are identifying between 60 and 100% of the critical problem areas. In all problem areas except one, student groups are also correctly identifying the source of the problems. The problem of input data conversion is an operating system/COBOL language difficulty. In considering this problem student groups related the problem to the first program of the ISLS where it is just encountered. However, the program itself is not the correct source. This situation represents the most sophisticated problem in the ISLS system and is well beyond the students' introductory exposure to operating systems and the COBOL language.

Sources of the other critical problems were all correctly identified indicating student understanding of:

1) undocumented User-Programmer interaction
2) program logical errors
3) not keeping documentation thoroughly consistent and up-to-date.
4) incomplete testing and verifying of data in the masterfile.

In order to correctly identify critical problems and their sources, the students must pay a great deal of attention to detail, recognize the key relationship between user and programmer/programmer-analyst, follow the program code, and understand the documentation package. The recognition-understandings exhibited by the student groups indicate a sound basis for effective communication with Information systems personnel.

Other frustrating, but non-critical, problems have been identified by student groups. Audit report results of problems identified by more than one group are shown in Table II. Problems 1 through 4 were most frequently related to a need for a more user

friendly system an aspect not discussed during the SLS. Problems 5 through 8 all relate to documentation. It is significant that half of the student groups identified a need for separate documentation for users, programmers, and operators. In the ISLS only programmer/systems analyst documentation was provided. All operator instructions were given in oral presentations. The last two errors listed in Table II relate to systems security issues. As a result of identifying these problems, student groups have gained working insight into the significant issues of user- oriented systems, adequate and reliable documentation, and security.

The results tend to indicate that students are recognizing and understanding the high level of integrated detail expectations for programs and documentation (which may be the design specifications). They are also aware of the need for complete, documented communication between user and programmer. In several instances they are projecting an understanding of what user-oriented systems, basic security issues, and documentation requirements for those involved with the system. Several groups noted that documentation language should be targeted for specific audiences, e.g. user, operator-programmer. Student groups have not indicated much understanding of the interaction of the hardware and operating system with the applications package. The educational effectiveness, however, of the SLS relating to applications package processes and requirements appears to be sound.

## CONCLUSION AND RECOMMENDATIONS

The evaluation of the SLS with regard to its objectives suggests that the SLS provides an effective technical supplement for an MIS course. Moreover, the successful use of the SLS to communicate key concepts in computer supported information systems supports the argument that extensive programming background is not a prerequisite for such learning.

As development of the SLS continues, future refinements and additions will include:

1) incorporating an audit trail in the file maintenance systems;
2) installing a program which integrates file maintenance procedures in order to provide a more realistic experience with such systems; and
3) extending the sequence to include data base management experience.

As a learning experience, the SLS begins with simple programming exercises and builds quickly to the introduction of more sophisticated systems concepts. In bypassing the usual programming introduction to data processing, the SLS focuses directly on those issues which will have the greatest managerial impact. Participants in the six SLS presentations evaluated have indicated significant levels of understanding and perspective with regard to management information systems. Through the development of the SLS and similar experiential learning tools, the information systems training of managers can be brought to bear more clearly on the realities of managerial practice.

TABLE I
SYSTEMS LEARNING SEQUENCE
DESIGN OBJECTIVES VS. CRITICAL AUDIT ITEMS REPORTED

| Critical Prob. Identified Yes or No / Correct Source Identified Yes or No | AUDIT GROUP FINDINGS A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1. Documentation/ Program Problem Data input specification per documentation not equiv. to data input specifications in program. | y / y | y / y | y / y | y / y | y / y | y / y |
| 2. Operating Sys./ COBOL Prob. Prog. format specifications result in alphabetic-numeric conversions of input. | y / n | y / n | / y | y / n | y / y | y / n |
| 3. Program Logic Prob. attempts to update records result in deletion of both the orig. record & the updating information. | y / y | y / y | y / y | / | / | y / y |
| 4. Documentation Prob. Documentation does not accurately reflect programs. | / y | y / y | y / y | y / y | y / y | y / y |
| 5. Prog. Operation Problem Prog. does not perform limit checks although such checks are suggested by the documentation. | / y | y / y | y / y | / | y / y | y / y |
| Total % critical problems identified. | 60 | 100 | 80 | 60 | 80 | 100 |
| Total num. of problems identified | 8 | 24 | 21 | 9 | 21 | 14 |

TABLE II
NON-CRITICAL PROBLEMS IDENTIFIED BY MORE THAN ONE GROUP

| PROBLEM SPECIFIED | GROUP IDENTIFYING PROBLEM | | | | | |
|---|---|---|---|---|---|---|
| | A | B | C | D | E | F |
| 1. File initialization program may be run only once. | X | X | X | X | | |
| 2. Error messages are hard to understand. | | X | X | | X | |
| 3. Hard to operate terminals. | | | X | X | | X |
| 4. Frequency of file maintenance runs per documentation not adequate for timely reports. | | X | X | | X | |
| 5. Although the documentation specifies a code with digits between 1 and 8 inclusive, the digits "0" and "9" may also be entered. | | X | X | | | X |
| 6. In file maintenance procedures, codes for leaving fields unchanged and leaving fields blank are not documented. | | X | X | | | X |
| 7. Although documentation refers to program listings, they are not provided. | | X | X | | | |
| 8. Documentation should be different for users and operators. | | X | | X | X | |
| 9. Deleted data in file maintenance is not listed for control. | | X | X | | X | |
| 10. The system does not restrict access for changing programs or data. | | X | X | | X | |