Managing resident intern rotation schedules using an on-line, user-controlled system

Jay M. Lightfoot
University of Northern Colorado

Follow this and additional works at: https://scholarworks.lib.csusb.edu/jiim
Part of the Management Information Systems Commons

Recommended Citation
Lightfoot, Jay M. (1996) "Managing resident intern rotation schedules using an on-line, user-controlled system," Journal of International Information Management: Vol. 5 : Iss. 1 , Article 5.
Available at: https://scholarworks.lib.csusb.edu/jiim/vol5/iss1/5
Managing resident intern rotation schedules using an on-line, user-controlled system

Jay M. Lightfoot
University of Northern Colorado

ABSTRACT

Accredited medical clinics that provide resident programs are faced with the problem of scheduling interns for service rotation. These schedules must meet a complex set of criteria to assure that all services are covered and each resident receives appropriate training. This paper describes the design and implementation of a windows-based resident scheduling system that meets these goals. Following that, the paper discusses the advantages and problems of working in a visual programming environment and outlines future enhancements to the system.

INTRODUCTION

All medical doctors must complete a rigorous multi-year internship before they can become board certified to practice medicine. This internship is similar to a standard college curriculum in that experience of a specific type and duration must be acquired in a predetermined sequence in order to complete the program. Those clinics that provide this experience must be accredited by the ACGME (Accreditation Council of Graduate Medical Education) certification board. If a resident intern does not receive the correct experience during the internship, she will not be allowed to take the board certification examination. This would drastically reduce her employment opportunities and could cause the clinic to lose accreditation.

One of the major difficulties that residency programs face is how best to schedule the interns so that all residency requirements are met and all the basic clinic services are continuously available.

1 The feminine pronoun "she" is used throughout this paper in order to avoid the grammatically incorrect use of "they" and the confusion caused by alternating pronoun gender. The reader should also be aware that the words intern and resident are used interchangeably throughout this paper even though this is not strictly correct. An intern is in the first year of the resident program while a resident is in the second and third years.
covered. Unlike a traditional college, clinics that provide residency service depend upon the interns for the majority of services provided. The clinic could not operate without the residents. Consequently, if a medical service is not continuously covered by a resident, life-threatening consequences are possible.

This paper describes the design and implementation of a windows-based resident scheduling system that was custom designed for a medium-sized family medicine clinic. The system is designed to be operated by the residents themselves and support personnel. The primary goal of the system is two-fold. First, it should facilitate the input and validation of resident rotation schedules to assure that all services are covered by appropriate personnel. Second, the system must make sure that each resident receives the proper experience to allow board certification.

**SYSTEM OVERVIEW**

**System Environment**

The rotation scheduling system was designed for a medium-sized family medical clinic located in a semi-rural western community. The clinic is accredited by the ACGME and provides the standard three-year residency program required for board certification in family medicine. Prior to developing the system, all rotation scheduling was done by hand by the incoming 2nd and 3rd year residents. Both groups met at the end of the curriculum year to work out an acceptable rotation for the following year. The residents, rather than clinic administration, generated the rotation schedules because they know their personal needs best and have the most to lose from an incorrect plan.

The schedule creation process is very difficult because the final schedules must meet a complex series of detailed and often conflicting criteria. Specifically, the completed rotation schedule must:

- completely cover the three primary services of pediatrics (Peds), medicine (MED), and obstetrics (OB),
- cover these services using personnel with appropriate training,
- ensure that residents in the first, second, and third year (i.e., R1, R2, and R3) all receive service time in accordance with local clinic and ACGME regulations,
- allow residents the flexibility to take elective services in areas of planned specialization,
- allow residents the flexibility to take time off for personal reasons (e.g., having a baby), and
- provide R3 residents the first choice of rotation assignments due to seniority.

A complicating factor is the fact that the residents constantly rotate through the program; thus, any scheduling expertise gained during the internship is lost with the graduating R3 class. In effect, each group of new residents must re-learn the unwritten scheduling heuristics. These scheduling heuristics are combined with formal ACGME rules and local clinic policy to generate rotation schedules.
The project to computerize the rotation scheduling process was initiated because the clinic administration was frustrated with the time spent producing the schedules. The primary job of a resident is to practice medicine, not to learn how to administer a residency program. Likewise, many of the residents were not inclined to spend the time needed to understand ACGME policy. In addition, the clinic administration was unhappy with the inflexibility of manually generated schedules. Manual schedules were prepared one year in advance. Once a schedule was completed, changes were not allowed unless absolutely necessary. The administration felt that this needlessly restricted the program and its participants. Finally, a computerized system was desired to improve scheduling consistency from one year to the next. Encoding the scheduling heuristics into a computer program allows consistent schedules to be generated quickly.

Design Philosophy

The rotation scheduling system is the first phase of a larger project to computerize all facets of clinic scheduling. The design philosophy for the full project is stated below:

- The system should be windows-based and must use existing personal computer (PC) equipment.
- The system should be based upon a single logical database. In addition, a simple, consistent user interface is required.
- The system initially will be written for a single workstation, but it should be relatively easy to modify it to work in a multi-user environment.
- Scheduling should be automated as much as possible; however, the system will take advantage of human input when it is appropriate to do so.
- Flexibility should be designed into the system so that new schedules can be generated quickly and scheduling rules can be modified with minimal effort.

The decision was made to write the system using Microsoft Access® for database support and Microsoft Visual Basic® for special processing requirements. This allows a single database for all schedule processing (via Access) and the full range of user-interface options available from a modern visual programming language (through Visual Basic and third-party add-in components). Both packages are windows-based and can run on available PC hardware. Microsoft Access is a multi-user database product, so future expandability is possible.

The system runs on existing IBM PC hardware. This machine is configured with an 80486-33 CPU, 8 MB of RAM, 250 MB hard drive, and a 14-inch color monitor. Despite this minimal configuration, the system runs surprisingly fast. An HP laserjet 4 printer is available through shared network access. Other PCs on the network may eventually use the system, but the initial release assumes a single user (i.e., no concurrent access to the database).
IMPLEMENTATION DETAILS

End User Interface

The rotation scheduling system was designed to allow non-system personnel (i.e., residents and clerical staff) to input and generate schedules. This places a heavy burden on the user interface. The interface had to be simple and informative. It could not rely on arcane syntax or complex command sequences. In short, it had to be intuitively obvious. Fortunately, this is the forte of the graphical environment provided by Microsoft Access and Visual Basic.

Both Access and Visual Basic allow the programmer to build input/output screens (called forms) by physically manipulating tools with the mouse on the computer screen. These tools include the standard Microsoft Windows™ components (e.g., scrolling list boxes, click buttons, radio buttons) and more exotic special purpose tools (e.g., data grid control, timer control, communication ports). In addition to these built-in components, numerous third-party vendors supply a variety of tools called VBXs that go beyond the capabilities that Microsoft provides. The combination of visual form construction, built-in tools, and VBX extensions creates a powerful user interface development environment.

Data Input and Validation

Data input and validation are key components of the end user interface. Within this realm, the first major interface design decision concerned how to package the scheduling data to maximize presentation clarity. The rotation scheduling system requires that three separate rotation schedules be created—one each for the first-, second-, and third-year residents. Rather than combine these three categories into one large, complex form, it was decided to build three separate forms with identical visual characteristics. This simplified the interface considerably and allowed for future customization of each input form. It was also in line with the research literature concerning consistency in human-computer interface design (Tanaka, Eberts, & Salvendy, 1991).

The next interface design problem concerned how best to display the information on the computer screen. There are typically seven to nine interns in each year of the residency program (giving a maximum combined total of twenty-seven residents). Each resident is scheduled for a rotation task each month of the year. This implied that the built-in Microsoft data grid control tool be used.

The grid control provides a spreadsheet-like row/column display of data from a database table. Practically unlimited amounts of data can be shown using the attached horizontal and vertical scroll bars. Unfortunately, the grid control only displays data, it does not allow the user to edit existing data. Likewise, it does not allow drop-down list boxes or input validation. All three features are required for the rotation forms. The solution to this problem was to use a third-party VBX grid control. The control selected has editing and validation capabilities and provides the same visual appearance as the built-in control.
The final interface design problem was the most complex—how best to provide user control of schedule validation. Validating a schedule means that the input data is checked against a series of complex rules. When an error occurs, the user must be informed about the problem and given the chance to correct it. Ideally, the user should be able to correct the problem immediately and re-validate only that portion of the schedule directly affected. This allows the user to focus on one problem at a time. The user should also be able to validate larger portions of the schedule, up to and including the full data grid. Devising a simple means to graphically display this range of processing options was challenging. It involved research into the design of easily learned interfaces (Polson & Lewis, 1990) and the effective use of color in screen design (Tufte, 1989). The solution is shown in Figure 1 below.

Figure 1. Rotation Schedule Input/Validation Form
The box labeled 'Validate Options' provides the user with the ability to control the scope of the validation. For instance, if 'Validate Off' is selected, no validation takes place. This option is useful for efficient data input. If 'Validate Single' is selected, either a single row or column is selected for validation. If 'Validate All' is chosen, all rows or all columns are validated. Finally, if 'Validate Table' is turned on, all rows and all columns are validated.

Closely related to the 'Validate Options' selection is the box labeled 'R1 Validation'. The buttons in this box give the user control over the direction of the validation. If the user clicks the button labeled 'Row' while 'Validate Single' is turned on, the row upon which the cursor rests is validated. Likewise if 'Col' is clicked, the current column is validated. Clicking 'Row' or 'Col' when 'Validate All' is on performs validation on all rows or columns in the grid. Finally, clicking the 'Table' button performs the equivalent of an all row validation followed by an all column validation. The intent of this design was to provide the user with a fine degree of control in a relatively simple manner.

A key function of data input forms is to validate user input. Only those tasks that are valid for the type of resident being scheduled should be accepted. If incorrect tasks are entered, the entire schedule becomes invalid. Drop-down list boxes are attached to each cell in the scheduling grid to enforce input validation rules. Figure 1 shows a cell with its drop-down list deployed. In addition to limiting input to legitimate values, this design feature also speeds up data entry because the drop-down control automatically locates the desired task as the user types the first letters of its name. The list of valid tasks for each resident class is maintained in a database table. The system administrator can modify this task list (and other variable task characteristics) at any time.

The final user interface aspect of Figure 1 concerns the small 'face' pictures next to the validate buttons. These pictures are used to notify the user of the results of the validation process. The smiling face means no errors were found. Predictably, the frowning face means some errors exist. If a cell has changed since the last validation, then a question mark is displayed indicating unknown status. Finally, a neutral face (not shown in Figure 1) indicates that only warning conditions exist. The face pictures change automatically based on the cursor position in the grid; thus, as the user points to a cell she is instantly aware of the validation status of that position. Apart from the whimsical aspect of the face icons, there is good reason for their inclusion in the validation form. Research has shown that human performance is greatly improved by including displays that take advantage of our built-in pattern recognition facilities (Bennett & Rach, 1992).

If a frowning or neutral face is shown, the user may click on the picture and see a list of the errors associated with the grid position. A sample error listing form is shown in Figure 2. This form lists all errors found in the most recent validation run. If desired, the user has the option of moving from the error list to the grid control on the main form, correcting the errors, and immediately repeating the validation. In this way, she can systematically find and correct scheduling errors in a wholly graphical environment. The error form also aids the user by allowing her to display a detailed explanation of each violated rule. This provides built-in help that is context sensitive to the errors actually encountered.
Figure 2. Validation Error Listing Form

Scheduling Report Output

The input validation form is half of the end user interface. The other half of the interface is the output reports produced by the system. The rotation scheduling system currently produces three simple reports. Each report prints the scheduling grid for the resident class in a spreadsheet-like row/column layout. The rows of the report relate to the resident names and the columns represent the scheduling periods. The content of each intersection cell is the task assigned to the resident for the period. The reports can be viewed on the computer screen or printed in hard copy form. Other reports are possible, but none are currently planned or under development.
Internal System Details

The end user interface provides the abstraction needed to guide the residents and clerical staff through the complex process of generating valid rotation schedules. In essence, its goal is to be simple and direct. The internal processes behind this interface are quite complex and have a different goal. The major goal of the internal system behind the interface is to provide a flexible environment where changes can be implemented with little (or no) programmer intervention. This is accomplished through good normalized database design, modular rule sets, and user-controlled data tables. Each of these topics are discussed below.

Normalized Tables

A basic principle of modern database design is normalization. This is the process of dividing data tables into smaller pieces to ensure good operating characteristics. As a general rule, data tables in a system should be normalized to avoid the problems of update, delete, and insert anomalies (Kroenke, 1995). The rotation scheduling system is no exception to the rule. The diagram shown in Figure 3 depicts the structure (i.e., the conceptual model) of the scheduling system.

Figure 3. Rotation Scheduling System Conceptual Model
The conceptual model of the rotation scheduling system is very simple. Basically, the diagram depicts that residents perform many scheduling tasks and each task can be performed by several residents. Each resident can belong to a number of teams and these teams coordinate a variety of tasks in a valid schedule. From the diagram, it is obvious that the rotation scheduling system is not complex because of the underlying data structure. Rather, the complexity is due to the intricate validation rules that must be applied to generate accurate schedules.

When designing a system with a very simple conceptual model, there is always the temptation to short-change the normalization process. This is especially true when the anticipated data volume is low and the update activity is limited. This was the case with the rotation scheduling system. In the short run, it would have been much quicker and easier to compress the ROTATION and TEAM tables into the RESIDENT table. This would have left only two tables and one many-to-many relationship to deal with. In the long run, this would have been a mistake. Including the ROTATION data in the RESIDENT table sets the stage for very subtle update anomalies and limits potential reporting options. Placing the TEAM data in the RESIDENT table redundantly stores team information and locks residents into a fixed number of teams. Finally, neglecting the normalization process would make it difficult to fit the rotation schedule into the larger general scheduling system that will eventually be built. The lesson this teaches is that normalization is not just for big, complex systems. The rotation scheduling system is more flexible and robust because the time was taken to design it properly.

Modular Rule Sets

The ACGME certification board occasionally changes the standards that must be met to maintain accreditation. Likewise, clinics periodically modify residency programs to accommodate market demand and to provide a better learning experience for the residents. This means that a computer system that generates clinic schedules must be maintained on a regular basis to keep up with environmental changes. Maintenance is a problem with any computer system. The problem is magnified when the system embodies a complex set of interrelated rules and conditions. The text that follows describes how the rules in the rotation scheduling system were structured and implemented to accommodate easy maintenance.

Two types of rules are present in the rotation scheduling system. The first type of rule concerns the accreditation requirements and clinic coverage policy for each task. As an example of this, consider the fact that every second-year resident must have at least two periods of MED experience during the year. This is an accreditation requirement and is best visualized as pertaining to rows in the scheduling grid. Coverage rules relate to columns in the scheduling grid and are intended to ensure that all the key tasks necessary for the clinic to operate are continuously covered. An example coverage policy states that there must be two second-year residents on MED rotation each period of the year. Every task in the system can have different coverage and accreditation requirements. Fortunately, the regularity of these rules allows them to be processed by a single procedure. This procedure is part of the validation code in the input data forms and is driven by data found in the TASK data table (discussed below).
The second type of rule can best be described as a free-form rule. These rules are much more complex than the coverage/accreditation rules discussed above. Free-form rules can check any condition that is conceivable. Below is the text of a typical free-form rule:

Each R1 resident should have one period as OB intern with another OB intern, one period as OB intern with a GYN intern, and one period as GYN intern with an OB intern. It is illegal for two GYNs to be scheduled in the same period. Likewise, there should never be more than two OBs or any combination of greater than two OBs and GYNs in a single period. When two OB interns are scheduled in the same period, they should not be members of the same primary team.

This particular rule is concerned with both row and column conditions within the R2 data grid. Other free-form rules increase the complexity by referring to data found in the other data grids (i.e., R1, R2, and R3). The free-form rules are all unique and each is subject to change at any time.

There are two general approaches that can be used to handle free-form rules. The first is to embed the validation logic into a single monolithic validation subroutine. This subroutine would contain the code for all rules. The second approach is to build a modular rule processing structure capable of executing stand-alone rules which are implemented as functions. Each function would accept input arguments (from the data grid), perform a single rule, and return the validation state to the calling structure. The first option (i.e., large subroutine) would result in a long complex procedure that would quickly become a maintenance nightmare. The second option would be very flexible, but would be more difficult to implement. The modular option was chosen for the rotation scheduling system.

The modular rule strategy is less efficient than the alternative approach; however, it has the advantages of easy maintenance and flexibility. For instance, a rule can be disabled simply by removing its function call. A rule can be reused by passing it different arguments in different situations. The order of rule processing can be modified easily by rearranging a few lines of code. If desired, a scheme can be devised that performs certain rules when certain conditions exist and other rules in all other cases. The possibilities are impressive and, best of all, maintenance is minimized.

User-Controlled Data Tables

The final internal design feature intended to provide a flexible scheduling environment involves user-controlled data tables. A large number of parameters are needed to perform validation scheduling. For instance, each rotation task has a distinct coverage and requirement value associated with it for each class of resident. There are three general approaches to encode these parameters into the system. These are listed below:

- hardcoded the parameter values into each routine that needs them,
- define each parameter as a global variable which is set when the program starts,
- build a table to store the parameters as part of the database.
The first option is the easiest to do initially but has the least flexibility. The second option is significantly more flexible, but requires that a programmer make all parameter changes (since they are built into the executable program). The third option is the most difficult to implement, but has the most flexibility of the three. The third option was used in the rotation scheduling system.

By storing parameters in a table, end users can update processing constants via a form whenever they wish. It is even possible for users to perform "what-if" analysis on rotation schedules to test the effect of proposed policy changes. Building the parameters into a table is more complex because additional input validation is required and an extra level of indirection exists, but the resulting system flexibility is worth the effort.

Several parameter tables are used in the rotation scheduling system. These include tables for tasks, teams, calendars, general rotation parameters, and general system parameters. Each table has a corresponding maintenance form that allows users (with appropriate privilege) to modify processing parameters. The TASK parameter maintenance form is shown in Figure 4. Since the parameters are updated in real-time, a user can modify a parameter and immediately generate new schedules that are based upon the new value. This flexibility provides users with the control they need to maintain the system without programmer intervention.

Figure 4. Task Parameter Maintenance Form
DISCUSSION OF THE VISUAL APPROACH TO PROGRAMMING

Visual Basic and Microsoft Access both provide visual programming environments. Visual tools were selected to build the rotation scheduling system for three reasons:

- to build an exceptional user interface,
- to allow quick prototypes to be generated and approved by the end user,
- to improve programmer productivity by applying pre-tested programming components to a complex system.

While the visual environment was successful on the first two counts, it was not successful on the third. The tool components of Visual Basic and Access are geared almost exclusively toward the visual aspects of the system. For example, both packages provide scrolling list boxes, click buttons, text boxes, etc. Neither package has much to offer in the non-visual realm. There are exceptions to this (e.g., timer control, masked edit control), but these exceptions are very specialized and not of general use to the programmer. One potential solution to this problem is to purchase third-party VBX controls. Third-party controls exist for many tasks and can be purchased for a reasonable fee; however, if third-party controls are unavailable, detailed procedural code must be written by hand. Writing procedural code is far less productive than connecting pretested programming components.

A second reason that productivity was not significantly enhanced by the visual environment can be attributed to the programming environment itself. The visual aspects of Visual Basic and Access are excellent; however, the debugging support is limited. Common debugging features such as break points based upon watch values and automated single step execution are missing in the releases of the products used in this project. Some of these concerns will be addressed in the next release, but advanced support (such as backstep) will still not be available.

The final reason that productivity was less than expected has to do with the fact that neither Visual Basic nor Access are true object-oriented programming environments. This limitation made it necessary to build three distinct validation forms (one for R1, R2, and R3 scheduling) instead of building a single generic form and making minor modifications to three instances of that form. Lack of true object-oriented support significantly lowers productivity by forcing the programmer to duplicate coding effort.

In summary, the visual approach to programming is a good choice when the user interface represents the major complexity of a system. It becomes less of a good choice when complex processing lies beneath the interface. The environment is capable of supporting the processing, but a significant amount of programming is required and this programming is not well supported by the environment.
FUTURE ENHANCEMENTS

The rotation scheduling system is complete and operational. It is the first phase of a larger project. Phase two of the project will develop call and clinic scheduling. Call scheduling is an expanded version of rotation scheduling. While the rotation schedule determines which residents are assigned to tasks for a period (i.e., month), the call schedule assigns residents to every shift within each day of the period. The call schedule will be based on the data collected by the rotation schedule. Hence, it will build upon the valid rotation schedule produced in phase one. Complex rules must be checked in call scheduling to make sure that each resident receives the proper service time and appropriate rest time between calls.

Clinic schedules assign those residents not currently on call (or asleep) to various clinics that require coverage. As before, clinic schedules will require that valid rotation and call schedules exist. Any changes anywhere in the scheduling chain will trigger schedule regeneration 'down the line'.

Phase three of the project will generate faculty and individual schedules. The faculty schedule assigns the doctors who supervise the clinic to something akin to call duty. A qualified faculty member must be available at all times to oversee the resident program. The faculty schedule will determine which doctors are on duty. The individual schedule is the final piece of the system. This component will allow a resident (or faculty member) to log in to the system and see a calendar that details all her duties for a specified period of time. This information will be a compilation of data found in the other schedules. When phase three is implemented, the system will most likely be modified to a multi-user configuration and distributed over the local area network.

SUMMARY

This paper described the design and implementation of a windows-based resident scheduling system. The system is built using a combination of Visual Basic and Microsoft Access and is designed to run on a personal computer. The system incorporates an innovative interface designed to make it easy for end users to operate the system. Internally, the system is designed with flexibility and ease of maintenance in mind. Modular coding units and table driven parameters are employed to achieve this. The system is functional and operates as designed. The initial system will soon be expanded to include other aspects of clinic scheduling.
REFERENCES


