A Software Tool for Rapidly Prototyping New Forms of Computer-based Assessments

Irvin R. Katz

January 1998

GRE Board Professional Report No. 91-06aP
ETS Research Report 97-15

Educational Testing Service, Princeton, New Jersey
A Software Tool for Rapidly Prototyping
New Forms of Computer-based Assessments

Irvin R. Katz

GRE Board Report No. 91-06aP

January 1998

This report presents the findings of a research project funded by and carried out under the auspices of the Graduate Record Examinations Board.

Educational Testing Service, Princeton, NJ 08541
Researchers are encouraged to express freely their professional judgment. Therefore, points of view or opinions stated in Graduate Record Examinations Board Reports do not necessarily represent official Graduate Record Examinations Board position or policy.

The Graduate Record Examinations Board and Educational Testing Service are dedicated to the principle of equal opportunity, and their programs, services, and employment policies are guided by that principle.

EDUCATIONAL TESTING SERVICE, ETS, the ETS logo, GRADUATE RECORD EXAMINATIONS, and GRE are registered trademarks of Educational Testing Service.

Copyright © 1998 by Educational Testing Service. All rights reserved.
Abstract

This report describes the Free-Response Authoring, Delivery, and Scoring System (FRADSS). FRADSS allows test developers, who might have no programming experience, to put their ideas for computer-based test items directly onto computer. The system was used for the GRE Mathematical Reasoning pre-pilot and is currently being used for the GRE Engineering New Testing Initiative. The report begins with a description of the "niche" in the test development process that FRADSS is intended to fill, then presents several items authored using FRADSS. After describing the various components of the system (authoring, delivery, and scoring), the report discusses benefits of using FRADSS in the early stages of test development. These benefits were gleaned from interviews with item writers who have used FRADSS in their work.
Acknowledgments

I thank Daniel Zuckerman for assisting in the development and maintenance of FRADSS. Randy Bennett, Kalle Gerritz, and Randy Kaplan provided helpful suggestions on earlier drafts of this work. Finally, I thank the ETS test developers who volunteered to use FRADSS in their work.
A Software Tool for Rapidly Prototyping New Forms of Computer-based Assessments

Changes in the types of items delivered in tests have outpaced changes in test development and test production procedures—most procedures are still geared toward multiple-choice, paper-and-pencil tests. Yet testing programs are currently exploring new forms of computer-based tests that better assess skills in a variety of subject matter domains. Rather than simply expanding use of multiple choice to presentation on computers, testing programs seek innovative assessment types, including test items that involve complex, constructed (or free) responses. Without the proper tools, development of computer-based tests can be far more costly than that of paper-and-pencil tests.

When developing new types of items for paper-and-pencil tests, it is reasonable to expect that not all the items written will eventually be included in a test. Through the various review processes, some items are naturally discarded. As testing programs move to computer-based tests, the cost of discarding items increases because of the time and resources needed to implement items on the computer. But what if item writers could themselves create each preliminary item on computer, interact with the item as a candidate would, revise the item, and even collect some initial response data? Then, only the items that passed this type of rigorous screening would be implemented in an operational testing environment such as Educational Testing Service’s Open Systems Architecture (OSA). This report describes a software tool that meets these goals, the Figural Response Authoring, Delivery, and Scoring system (FRADSS). Using FRADSS, nonprogramming staff such as item writers can create dynamic, working prototypes of computer-based test items.

Despite being delivered as dynamic artifacts, computer-based items are currently designed on paper, a static medium. This disparity between the design medium and the working product is a limiting factor in the types of computer-based items that can easily be designed. Designing computer-based items on paper requires the test developer to be able to imagine user interactions. This is similar to the situation of architects who design on paper, but must “see” the 3D structure of a building in their minds. Architects spend many years developing this skill, and it is unreasonable to expect someone to imagine the workings of a complex item type just by looking at static item specifications. Further, different people may interpret specifications differently, leading to miscommunications among item writers and others.

The system described in this report, FRADSS, is intended to replace paper and pencil as the primary medium for designing computer-based items. FRADSS allows test developers, without programming, to construct working prototypes of test items and to interact with those items just as a candidate would. Because items can be quickly implemented on the computer (at least in approximate form), it is not necessary for item reviewers to imagine candidate interactions; the reviewers can actually evaluate the items’ interactions themselves.
Before describing FRADSS, I first introduce the types of items that can be currently created using FRADSS. As will be shown, the power in FRADSS lies its paradigm for prototyping. Instead of providing unchangeable templates into which item writers insert static text or graphics, FRADSS gives item writers a set of item "building blocks." The building blocks, which correspond to a variety of candidate response actions, may be mixed-and-matched along with user-provided text and graphics. In a template approach, each new template corresponds to a new item type. In FRADSS, each new building block corresponds to a whole set of item types, depending on how the blocks are combined.

**Item Genre: What Is Meant by "Computer-based Items"**

The items described in this report were motivated by the figural response item format. Figural response is a constructed-response item type in which candidates must demonstrate their knowledge by drawing a diagram or manipulating graphical material on the computer. For example, in some items, candidates are asked to draw the graph of a function; other items involve the arrangement of provided components into a logic circuit having certain characteristics. Figural-response items are currently used in paper-and-pencil format on the National Assessment of Educational Progress science exam, and have been piloted under various testing programs in domains such as architecture, biology, chemistry, physics, engineering, and mathematics.

As technological resources improve, so does the range of items types that can be computer-administered. The items presented below go beyond previous uses of the figural response format. Some items involve entering text or equations, or manipulating phrases on the computer screen. Although FRADSS has its roots in research on the figural response format, the types of items that can be prototyped, piloted, and analyzed extend prior definitions of "figural response" considerably.

Figure 1 is an example item and will be used to point out the various parts of the delivery interface—what the candidate sees. In each item, candidates are provided a set of buttons for performing the actions necessary to answer the item. Depending on what button is pressed, different help text is displayed in the area to the right, which tells a candidate how to perform the action currently selected. Candidates draw on or manipulate the objects provided in the lower area, which is called the background and solution area. The buttons to the right are used by the candidate to navigate through a test. Candidates may move to the previous or next item, or may skip to previous or next unattempted items. Alternatively, a menu is provided to allow candidates to jump directly to particular items. Items in this menu are checked off if the candidate has already attempted them.
The first item of mathematical reasoning (Figure 1) requires candidates to place one of several provided curves onto a coordinate system such that the resulting curve corresponds to a particular equation. The candidate clicks and drags one of the curves to move it, and the curve "snaps" to the coordinate system. This latter feature may be enabled or disabled depending on the item specifications. The click-and-drag action for moving objects is similar to the corresponding actions used in most commercial graphics applications.

The next item is an example of line drawing (Figure 2). To solve the item, the candidate must draw a line on the provided axes, subject to the requirements of the item stem. Drawing a line involves clicking and holding the mouse button down, then dragging the mouse to draw the line. The line "rubber bands" as long as the mouse button is down. Also, the line snaps to the gridlines of the coordinate system, which reduces the dexterity needed to respond. If the candidate makes an error, there are two ways to correct the error. The candidate may use the "Erase" tool, click on the line to erase it, then draw a new line. Alternatively, the candidate can move or resize the line using the "Move/Resize Line" tool by either clicking and dragging the line from its center (to move) or by clicking and dragging from an endpoint (to resize). These operations of drawing and resizing lines are almost identical to the corresponding user actions in most commercial graphics applications.
In the next item (Figure 3), the candidate must shade in a region corresponding to the area under the provided curve. Shading is performed by clicking in a region, which then is filled with a gray pattern. Clicking in an already shaded region causes it to become unshaded.

![Shade region item](image1)

**Figure 3:** Shade region item (left) and sample response (right)

The final item requires candidates to enter text into specific areas on the solution entry screen (Figure 4). The candidates click in the box into which they wish to enter text, then begin typing. Candidates cannot enter more characters than the size of the text entry area. As is the case with most commercial word processors, candidates may click on existing text to change the insertion point for text, or select a set of characters and type in order to replace those characters.

![Enter text item](image2)

**Figure 4:** Enter text item (left) and sample response (right)

In summary, this section has shown items demonstrating a number of different response facilities that FRADSS supports. These response types are by no means novel in that several other projects have explored them. The main point here is that even with a somewhat limited set of response "primitives," a variety of new test items may be generated. The next section shows how these items are created, without programming, by item authors.
Figural Response Authoring, Delivery, and Scoring System:  
System Overview

FRADSS was designed to aid the early stages of item development, when discarding and revision of items occur most. A primary use of FRADSS is therefore in the prepilot testing of items. FRADSS is a self-contained system that provides facilities for most phases of item piloting—authoring of items, delivery of tests, scoring of responses, and reporting of scores. FRADSS’s time and cost-savings for item piloting has already been demonstrated in a recent GRE initiative on Mathematical Reasoning (MR), which is described later in this report. This section describes the several components that comprise FRADSS.

Item Authoring

The authoring component of FRADSS makes it possible for a test developer with no programming experience to create graphical items on the computer, interact with the items as a test taker would, and immediately make necessary revisions. Graphics can be imported directly from commercial packages, and the test developer can specify what operations the test taker will be allowed to perform—for example, moving an object provided on the screen, drawing a line or diagram, or entering text. The system stores completed items and can compile a test including whichever items the developer selects for tryout.

The cost benefits of FRADSS at this stage of item development are substantial. The development of a new computer-based item type normally involves a test developer, test production staff, and programming staff. Numerous iterations are usually necessary before the item looks and performs to the satisfaction of all parties—a costly process. FRADSS places the entire authoring, reviewing, trying out, and revising process in the test developer’s hands and allows the developer to decide at this early stage which items to continue refining and which to discard.

This fast generate-test-revise cycle is among the key cost-saving features of FRADSS. In the usual cycle, item writers design a new item type, send the item to be implemented on computer by test production or programming staff, then review the result; FRADSS enables item writers to put items onto computer and make their own modifications. There is an analogy between this description of organizational change and the change that occurred when word-processing software became generally available. Before word-processing software became common, a staff member would draft a manuscript on paper and send the manuscript to be typed on computer and

---

1In this report, “prepilot” refers to a test administration having the goal of providing early information to test development staff on the feasibility of item types for large scale use (e.g., do candidates understand what they are being asked to do? Does the computer administration interfere with the candidates’ ability to respond?). Beyond these goals, a prepilot must be a quick and inexpensive collection of data. This type of administration is in contrast to a formal “pilot,” the goal of which is usually to collect item statistics. Thus, such a pilot would involve items and a testing situation as close to the operational version as possible.
formatted by the word-processing center. The manuscript’s author would then review a printout of the manuscript and make corrections, which were then sent back to the word-processing center. Now that easy-to-use word-processing software is available to all staff, authors draft, review, and revise their manuscripts directly, with concomitant time savings.

**Design Goals.** FRADSS’s design seeks to make use of test developers’ experience with commercial graphics packages, such as MacPaint and PC Paintbrush. Typical graphics applications use a “palette-canvas” interface metaphor—like an artist’s palette, the application provides a set of tools that determine what is “painted” on the canvas or how the current canvas is altered (e.g., moving or erasing objects). FRADSS generalizes and extends this metaphor into a prototyping system. The “palette” of each test item is editable by the item author, who selects the tools available to candidates in answering the item. The “canvas” is stocked with background material (e.g., the problem statement) and objects that the candidate manipulates with the provided tools.

![Figure 5: Sample elements of item-authoring environment](image)

The FRADSS item-authoring environment consists of three elements: widgets, tools, and allowed actions (Figure 5). **Widgets** are user-interface elements such as text fields, static graphics, and resizable lines. Widgets may have associated attributes, such as the font and size of text in a text field. **Tools** are the “verbs” that act on the “nouns” of the interface (widgets, screen). **Allowed actions** are the constraints that govern how tools may act on widgets. In contrast to a graphics package, FRADSS permits authors to customize each tool’s behavior for the needs of a specific test item. Finally, mediating between widgets and tools are **allowed actions**, which permit item authors to specify which tools may be used on which widgets. For example, certain graphics may be background information, so should be immutable by the candidate, but the candidate may manipulate other graphics (e.g., moving a curve onto a set of axes). Overall, authors create items by importing graphics or specifying other user-interface objects, choosing the tools that are available to students in responding to the question, and delineating—via dialog boxes and specially designed graphical objects—how the tools and provided interface objects interact.
Example of use. This section presents an example of using FRADSS to develop a simple free-response item. The screen layout of FRADSS's authoring component is essentially a blank delivery screen (the screen that a candidate sees when taking a test), with an additional set of palettes on the right (Figure 6). One palette contains the building blocks of an item, the tools that an item author can provide to test takers. A second palette contains several user-interface widgets, such as text fields and an on-screen keyboard, which may be added to the canvas of test items. Note that the system does not provide sophisticated graphic-drawing tools; instead, an author may use whichever graphics application he or she is most familiar with, and then copy-and-paste the graphic onto the item's canvas. Overall, authoring an item consists of (a) choosing the appropriate tools for students, (b) including any widgets needed for the item, (c) importing graphics from any commercial graphics application, and (d) if necessary, specifying restrictions on students' use of the provided tools (e.g., only certain imported objects are moveable). In the example below, it is assumed that any needed graphics already exist.

The item to be created requires candidates to choose the appropriate curve and move it onto the coordinate system such that the curve satisfies the stated equation (Figure 7). To get the background information for the item, the author switches to a graphics application and copies the information—that is, the coordinate system and item stem. The author then switches back to FRADSS, pastes the graphic, and positions it on the canvas. To continue creating this item, the author specifies the tools candidates can access by clicking on the appropriate button and placing that button into the delivery palette (Figure 8). The only tool needed is "Move Object." Now the author must separately copy each of the closed curves into FRADSS. The reason each curve is copied separately is that each must be separately manipulated by the
candidate. By default, FRADSS assumes that any imported graphic is part of the background and so is untouchable by the candidate. The author must specify that the closed curves are moveable by the candidate—double-clicking on a closed curve brings up the "allowed actions" dialog box (Figure 9). The author specifies that the curve may be moved, then closes the dialog and repeats this process for the other curves. The author then names the item and saves it onto disk, which completes the authoring of this item.

![Figure 7: Move object item](image)

Figure 7: Move object item

![Figure 8: Partially completed move object item. Move Object tool is selected for placement.](image)
After an item is saved onto the disk, the author may test the item, bringing up a sample delivery window with the actual working item. The testing facility allows item authors to try out the item and, if desired, print out the item (or a solution) to facilitate review by others. If the item does not perform as expected, the author returns to the authoring system to edit the item as needed.

Test Compilation and Delivery

When a set of items have been authored, item writers can compile the items into a test, determining which items should be included in the test and in what order. Figure 10 shows the Test Compiler window. On the left is a scrolling list of all of the items created by the authoring system (the item pool). The list on the right (empty in the figure) contains the items, in order, as they will appear on a test. Items are added to the end of the current test by selecting an item from the item pool list and pressing the "Move" button (Figure 11). As with individual items, a test is given a name by the author when the item is saved onto disk.
Using the delivery component of FRADSS, the test developer can send the compiled items to a pre pilot test site for administration to a group of candidates. The test takers interact with the items as they would if the items were part of an operational test. The delivery component allows candidates to navigate through the test freely, to keep track of which items have been attempted (i.e., a response or partial response has been entered to an item), and to move through the test looking only at those items not yet attempted. In addition to candidates’ final solutions, the system records each test taker’s path toward that solution, so that test developers can later review the actual interaction as if they were looking over the candidate’s shoulder.

Response Scoring

The scoring component of FRADSS currently supports manual scoring (automatic scoring facilities are under development). The system presents each item and the human judge indicates his or her rating of the response directly on the computer (e.g., Figure 13). Judges may view candidate solutions either “by candidate” (i.e., score all solutions produced by one candidate) or “by item.” In by-item viewing, the judge rates the responses to one item generated by all candidates. Even though
different candidates' solutions are scored, the scoring component stores each candidate's scores separately from those of other candidates.

Score Reporting

When all responses have been scored, FRADSS can generate various types of compiled score reports. The format of these reports is flexible and can be altered to facilitate analyzing the compiled scores using statistical software.

Benefits of FRADSS for Test Development

FRADSS is in current use by ETS staff developing items for the GRE Engineering modular test, and has been used for the GRE Mathematical Reasoning prepilot data collection. Increasingly, other groups within ETS use FRADSS for creating innovative assessments in various domains (e.g., economics, German). Informal interviews with staff using FRADSS reveals some positive effects on the typical test development process. This section describes some of these benefits.

Currently at ETS, when a test developer has created (and reviewed) several items, test production staff draw the necessary graphics and implement the item in the ETS operational testing environment called the Open System Architecture (OSA). If any item requires new facilities in OSA, either the item is reworked or OSA programming staff add the required facilities. In either case, it often happens that the resulting item is not quite as the test developer had intended. For example, a graphic might have been misdrawn, the new OSA facility might not work as expected, or, having viewed the item on computer, the test developer may realize that his or her specification to test production staff was incorrect. When this happens, the test developer must inform the appropriate staff member, who then corrects the error. This cycling between test development and test production staff can increase the time it takes an item to be correctly implemented.

FRADSS facilitates communication among test development, test production, and programming staff. Because test developers implement prototype items on the computer themselves, the review cycle is accelerated. When an item is implemented incorrectly in some way, the test developer using FRADSS can edit the item immediately, without having to pass corrections to others. As a result, more time can be devoted to reviewing and revising items, resulting in more refined items being given to test production staff when moving to operational implementation. More refined items means potentially fewer incidences of test development changing an item after seeing it implemented in OSA.

Because test developers have implemented the items themselves, they have already explored a range of possible response types for a particular item, and are in a good position to talk with test production staff about the specific issues involved in implementing that item (e.g., issues of screen layout, response types). Thus, having test

\[2^2\] This further development of FRADSS is funded through the relevant testing programs.
developers use FRADSS to implement items themselves is a process of education. Test developers learn firsthand the issues and pitfalls associated with placing the item onto computer. As a result, when the item is ready to be transferred to OSA, the test developer has a better chance of being aware of what would be needed to make the item usable under OSA. This experience also allows test developers to make more informed decisions if changes are needed in the move from FRADSS implementation to OSA implementation of an item.

In the process of implementing items in FRADSS, test developers also note problems that were not raised in reviews of the paper versions of items. Issues such as screen layout or details of candidate interactions are difficult to resolve unless one actually tries implementing an item. Other problems may be discovered with an item when it is pilot tested. With FRADSS, test developers can quickly implement items and collect preliminary data on items, and then use this information to decide which item types to pursue and which to abandon.

Another benefit is that the FRADSS implementation of an item acts as a “living specification” of how the item should look and work in OSA. Rather than specifying an item on paper, a test developer can give each FRADSS item directly to test production (or programming) staff with instructions such as “I want the OSA item to work like this item, except the OSA item should ...” That is, the specifications are provided in comparison with a working prototype rather than being written from scratch. Working with such a living specification reduces misunderstandings between test developers and test production staff because both parties have a concrete example of the item to which they can refer during discussions and implementation for operational use.

As implied by the benefits described above, FRADSS may be seen as a visualization and communication tool. It is difficult for test developers to imagine how an item will look to a candidate and how it will behave just from drawing out a sketch on paper. FRADSS gives test developers help in visualizing an item; even if the FRADSS-implemented item is not exactly what the test developer wants, he or she is still aided because it is easier to imagine a slightly different behavior from an actual working prototype than to imagine the whole thing from static paper and pencil. In addition, when discussing and reviewing items in a group, with a live prototype everyone reviews the same item. With paper and pencil, different people may imagine subtly different behaviors that could create confusion during (or after) a review session.

In addition to improving communication among test developers and between test development and production staff, FRADSS may support test developers’ creativity. That is, FRADSS not only helps to accelerate the test development process, but also provides the potential for more creative and better test items being generated by test developers. One way that creativity is enhanced is because FRADSS allows test developers to more easily explore a wider variety of response types than they could using OSA. When developing new types of items, test developers might invent new ways for students to respond to test items. If not already a part of FRADSS or OSA, these new response types must be created by technical staff (i.e., via programming).
Furthermore, programming staff must add the new response types to the system in such a way that nonprogramming item authors can manipulate how and when candidates use the new response type. For example, MR items required responses in which candidates shaded arbitrary regions on the provided graphic. This facility (the "Shade Region" tool) had to be added to FRADSS by programming staff.

OSA is an operational environment and so has many more constraints on its design compared with FRADSS, which is a prototyping environment. Changes to OSA must work smoothly with the other pieces of this large system. OSA must always be "bulletproof" in the sense that the system should be practically 100% reliable. All new response types must be scorable by computer. For computer-adaptive testing, all scoring is performed in real time (as the candidate waits), so any new response type must have scoring that can be completed quickly and efficiently.

Programming by FRADSS developers is less constrained because the system is used for item prototyping, not for high-stakes delivery. The entire system is relatively small compared with OSA, so there are fewer components that a new response type needs to take into account. Finally, because the system is not used for actual examinations, the "look and feel" of the interface need not be as professional looking as OSA. The system is freed from constraints of scoring, and even when scoring is needed, it can be implemented with less regard to the stringent memory constraints imposed by standard ETS testing hardware. Overall, owing to these relaxed restrictions, changes to FRADSS can be implemented relatively quickly, easily, and inexpensively.

Evaluating FRADSS: Is It Useful?

Evaluation of FRADSS is ongoing; however, preliminary evidence suggests that FRADSS successfully meets its goal of providing a usable rapid prototyping environment to nonprogrammers. FRADSS has been used by eight test development staff at ETS, who all volunteered to use the system in their daily work. These authors were trained to use the basics of FRADSS within a few hours (2-3 hours). After that time, authors could create items largely on their own. Using FRADSS, these item authors have implemented approximately 230 distinct items, which can be classified into approximately 70 different item types (e.g., graph a function, arrange words in a sentence, indicate an area on a graph). Anecdotally, authors report that they explore a wider variety of item designs using FRADSS than when creating items on paper (which would be put onto computer by production staff). Similar to programmer-oriented prototyping software, FRADSS appears to accelerate the item creation and revision cycle, giving authors more time to refine items.

These items, in turn, have been shown to be usable by their target user population—test takers. Several items have been delivered in preliminary tests to student volunteers; as described below, more than 200 such volunteers have taken a test on FRADSS to date. The usability of FRADSS's delivery system is suggested by results of a questionnaire administered after a prepilot of new items types in mathematics
In this questionnaire, 75% of students reported that they found the computer test-taking environment easy to use; 62% thought that they would perform equally well on a computer-delivered test or a paper-and-pencil test.

The next section describes examples of FRADSS's contribution to item development for GRE testing programs. These examples serve as case studies of the types of productivity increases that FRADSS can facilitate.

**FRADSS Support of GRE Research and Development**

FRADSS has been used by test development staff on three GRE projects to date: (1) an exploratory study of figural response items for the new GRE Engineering Subject test, (2) development of a more refined pool of new items for the GRE New Testing Initiative (NTI) in Engineering, and (3) a prepilot of items for the Mathematical Reasoning initiative.

**GRE Engineering Exploratory Item Development and Experimentation.** FRADSS was used in the development of new figural response items suitable for the new modular GRE Subject Test in Engineering. As the items were designed to assess discipline-specific engineering skills, most of the items were considerably more complex than the mathematics items presented earlier. Three types of items were developed: design, graph, and vector. All of the items were implemented by one person (the first author); these items were authored at the same time that FRADSS was first being developed.

Design items (Figure 14) present candidates with the specifications for some artifact (e.g., a circuit) along with appropriate components for building that artifact (e.g., circuit components). Unlike the "move object" item described earlier, the components represent "bins" of components. That is, when the candidate moves a component, a copy of the component is manipulated by the candidate. As a result, although only one component is visually provided on the screen, candidates have access to as many of those components as they need. The design items typically also involved the "rotate" tool, which allows candidates to rotate a selected component in 90° increments.

---

3. This work was conducted in collaboration with Debbie Friedman of ETS Research and John Economou, Michael Kaplan, and Ray Thompson of GRE Test Development.
Graph items (Figure 15) present candidates with a set of axes and a diagram of a physical situation or electrical circuit. The candidates' task is to draw the appropriate curve (or curves) on the provided axes. FRADS does not provide facilities for drawing curved lines; instead, candidates are asked to approximate their curves using several straight lines. Although this approach to curve drawing may not be ideal, it is easier to learn and perform than other approaches available in commercial graphics packages.

Vector items (Figure 16) present a diagram of a physical situation and direct candidates to indicate particular forces acting on parts of the diagram. Candidates use the "Draw Arrow" tool to create their vectors. This tool works identically to the "Draw Line" tool.
FRADSS was also used in collecting and analyzing data for this project. Data from 23 subjects were collected as they solved approximately 24 items in mechanical or electrical engineering. A portion of the analyses included investigating the solution strategies candidates used in solving the items. The time-stamped interaction log, automatically collected as candidates interact with FRADSS, facilitated these analyses by providing an alternative to repeated viewing of videotapes of candidate performance.

**GRE Engineering Demonstration Item Pool.** Based on the results of the exploratory development of figural response items for engineering, a second pool of items was created. This item pool will be used in demonstrations to engineering educators to garner support for the NT1 Engineering exam. The items will also be used in a prepilot, similar to the data collection done for the Mathematical Reasoning initiative, which is described in the next section. The authored items consisted of a combination of existing GRE engineering items and newly commissioned items written by engineering educators. Item-writing guidelines were sent to each outside item writer; the constructed-response portion of these guidelines were based on the lessons learned from the exploratory item development described above.

The new engineering items consist of 60 constructed-response and multiple-choice items designed to assess knowledge of basic engineering concepts, mechanical engineering, or electrical engineering. All items were implemented using FRADSS by an individual test developer. Although the test developer was learning FRADSS as he implemented the earlier items, he was able to implement the items in approximately 75 hours over a six-week period. Although an average rate of one item per hour may not seem high, keep in mind that some of the items involved complex constructed responses (e.g., build an electrical circuit). Additionally, the items generally represented distinct item types—few items were simple variants of others. These results demonstrate the productivity that can be obtained using FRADSS.
Mathematical Reasoning Pre-Pilot Test Administration. FRADSS's contribution to the November 1993 pre-pilot of GRE Mathematical Reasoning items provides a model for cost-effective and efficient pilot testing of new items. Item writers used FRADSS to develop and review items directly on computer. GRE graduate assistants collected data from 180 candidates, employing the delivery component of FRADSS. Candidate responses were scored on-line in FRADSS by human judges, and the resulting scores were provided to data analysts without the need for an additional data-entry step. A timeline of events suggests the benefits offered by FRADSS for such pilot testing:

September. Before the decision to use FRADSS was made, GRE item writers developed on paper 30 new mathematical reasoning item types. Test production staff then created the necessary computer graphics, and the items were placed into FRADSS, using its authoring component, by one FRADSS developer. Simultaneously, two item writers received training in FRADSS.

October. Items were reviewed (on computer) by the item writers. At first, all revisions were carried out by FRADSS staff or by item writers with the aid of FRADSS staff. During October, as item writers became more proficient in FRADSS, they assumed more responsibility for implementing item revisions. Eventually, all revisions were performed directly by item writers, without the aid of FRADSS staff. FRADSS staff implemented only those revisions that required programming (e.g., whether or not elapsed time is displayed as a student takes a test). These programming changes amounted to fewer than 10% of all revisions.

November. During this month, FRADSS was used by the GRE graduate assistants to collect data. As FRADSS currently works only on the Macintosh, it was necessary to rent computers for the pre-pilot. Before sending the machines to the graduate assistants, a number of ETS test developers, unrelated to the Mathematical Reasoning project, reviewed the items and provided feedback on the tutorial materials. The machines were sent to assistants who had never used FRADSS before. Nevertheless, the minimal instructional material appeared to be sufficient as no questions regarding the use of FRADSS were raised by the assistants and the data from all 180 subjects were collected successfully, with no losses due to technical problems. Graduate assistants mailed diskettes back to ETS that contained the collected data for each week.

When the data arrived at ETS, they were scored by two ETS research assistants who employed the scoring component of FRADSS. That is, all candidate responses were scored manually, directly on the computer. The compiled scores were sent directly to statistical packages without the need for an additional data-entry step. In

---

*This work was conducted in collaboration with Jackie Briel and Rob Durso of GRE Programs and Matthew Bell, Daryl Ezzo, Jutta Levin, and Mary Morley of GRE Test Development.

*Planned enhancements to FRADSS include facilities for providing a library of standard graphics used in various testing domains (e.g., math: coordinate systems, curves). Additionally, as more items are constructed using FRADSS, item writers will be able to "borrow" graphics from previously written items—FRADSS makes it easy to copy graphics from one item to another.
addition to scores, the system reported seconds-to-solution for each item answered by each candidate.

_December._ During this month, staff analyzed the time and score data and reviewed the comments made by candidates about the items and computer delivery system. In addition, plans were made for a larger scale pilot test, utilizing Sylvan test centers (i.e., closer to an operational test).

_Summary._ Items went from paper to computer to pilot administration to analysis in a little under four months. Early estimates had suggested that in the same amount of time, the original items could only have been implemented in OSA. Given the results of the pre-pilot, items are further refined; instead of implementing in OSA all items before any testing, it is now only necessary to implement those items that “pass” the rigorous screen of the pre-pilot. For example, because of this pilot test, MR staff decided to eliminate an item type from further testing. Had the pilot test not occurred, this rejected item type would have been implemented in OSA, further instances of this item type would have been written and reviewed, and the items would have been included in a much larger scale pilot test under OSA. Thus, because of the use of FRADSS, the GRE Program avoided the considerable costs of implementing, writing, reviewing, and testing items of a type destined to be eliminated.

**Conclusions**

Using FRADSS, a test developer can create dynamic, working prototypes of computer-based test items. Collecting pilot candidate responses, scoring the responses, and creating score reports (including each student’s time to complete each question) are all facilitated by the system. Although other software tools such as Visual Basic and Hypercard can be used to create test items, these systems require some programming ability. To our knowledge, FRADSS is the only system that is specifically designed to allow nonprogrammers to develop free-response test items on computer and to administer the items in a test.