Design for Better Data: How Software and Users Interact Onscreen Matters to Data Quality

by Adrian Williams, RHIA

Well-designed software screens are more than just user-friendly. They encourage accurate, consistent data capture. An example from coding shows how.

Computer technology is supposed to make our lives easier, especially at work. Sometimes, however, it's easy to question if this is the case. Poorly designed software that confronts the user with confusing screens, excessive data entry fields, or unclear navigational tools can be frustrating—it also threatens the quality of the data that users enter. While software screen design may seem like a minor detail in your overall data quality program, taking the time to assess screen design when selecting new software applications or when evaluating current ones can be an important step in achieving your data quality objectives.

Principles of Good Design

Data integrity is the cornerstone of health information management. A patient's health relies on quality information—accurate, timely data supports good care. An organization's financial well-being depends on high-quality data, also; incorrect patient information can result in increased compliance risk, rework, and claim denials. When designing paper forms, HIM professionals have always worked to ensure that appropriate information is captured accurately. The same goal applies to software design. Software should simplify the work of gathering patient data while ensuring that the information captured is reliable and of high quality.

Software and users interact on the computer screen, and the image presented there is called a graphical user interface, or GUI—that is, the onscreen images that allow the user to interface with the software. Screen images include buttons, menus, and icons that permit the user to give commands to the software. It is a software program's GUI that can either frustrate or support the user as he or she accomplishes automated tasks. A GUI's key characteristics determine how easy a software program is to use and how well it can support data quality. Good GUIs pave a smooth, consistent road, provide plenty of road signs, offer travel tips and alerts, and make it easy to turn around and go back when necessary.

Navigating New Territory

Few things are more discouraging than not knowing where you are going. Stopping to ask for directions or trying to read an unfamiliar map can make travel frustrating and increase the amount of time it takes to get to your final destination. The same holds true with software. A poorly designed GUI can cause a user to spend more time figuring out how to navigate the program than the actual time spent completing the task. What are some of the key design features that help make a software program easy to navigate?

In the US we read from left to right and top to bottom. Our computer screen should follow the same pattern. The GUI's elements should be clearly marked. Buttons, icons, and links should be labeled with clear and consistent language so that the user knows exactly what action will result from clicking on a specific task.

It should be obvious how to move forward from one screen to the next and how to move back through previous screens, as well. Enhanced visuals can help users navigate a software program. Fields or buttons that are grayed-out can help prevent incorrect data entry, for instance. Images that depict a specific action rather than text that describe it can make software functions easier to identify. Color can draw attention to a specific field, focusing the user's attention on an important step in the process. Careful use of graphic images, language, color, and text in GUI design helps the user successfully navigate the program.

Keep It Consistent

When an unexpected detour changes your regular morning commute, the result can be disruptive. People like consistency. Uniformity in software screen design is an important factor to consider because it contributes to a consistent, reliable workflow.

Layouts should remain the same as the user navigates from screen to screen, with buttons, fields, and labels formatted uniformly, which make the user more comfortable with the software and avoid confusion. Standard formats also help reduce the amount of software training required. Consistency of design allows users to perform tasks efficiently with fewer opportunities for mistakes.

The Ability to Forgive

To err is human—especially when using computer software. Some software programs can be extremely unforgiving when a mistake occurs, making data entry tedious instead of efficient. Poorly designed screens can also prevent the discovery of errors until after information has been saved and sent to the next step in the process.

Good design makes mistakes easier to identify and easily corrected. For example, if a coder attempts to code for prostate cancer on a female patient, the software screen should provide an immediate alert of the error. By making it possible to catch this type of mistake right at the point of coding, a well-designed GUI can save time and promote data integrity.

Since it's human nature to want to explore when using software, software programs should allow users to push buttons and open windows without threatening accuracy. Simple ways to cancel out of a screen and undo actions will aid in collecting accurate patient information.

Balancing Speed and Accuracy

Hospitals and health systems face constant pressure to process information quickly, from making a patient's latest clinical data available to expediting the coding process and avoiding increased bill-hold days. Computer software should enhance workflow by simplifying data entry and streamlining repetitive tasks. If designed with the user in mind, software screens will limit the number of required key strokes or mouse clicks, allow for multiple tasks to be accomplished simultaneously, and make automated tools accessible with a simple click of a button.

However, HIM departments should carefully weigh the value of time-saving features with the need for data quality. Some software programs are designed to condense processes into a few simple data entry screens. This kind of simplification saves time, but at the expense of accuracy. The best design carefully balances the need for accuracy with the pressure for speed. For example, software that incorporates a variety of coding information in a single screen allows for thorough data capture but also speeds the process by making it possible to assign multiple codes at one time. In general, software screens should support data quality but should not be too cumbersome in trying to be accurate. It's a difficult balance, but an important one.

Virtual Memory for the User

Sometimes it seems as if rules and regulations change on a daily basis. Given the growing complexity of healthcare and increasing demands for data reporting, it's not surprising that HIM departments rely on computer software to track and store information. Software can offer users an adjunct memory, prompting them for more information, linking to online reference materials, checking for errors or warning signs, and providing edits to help guide the user along the correct road. Software screens that include these features and are regularly updated to reflect current regulations help ensure data quality and promote compliance.

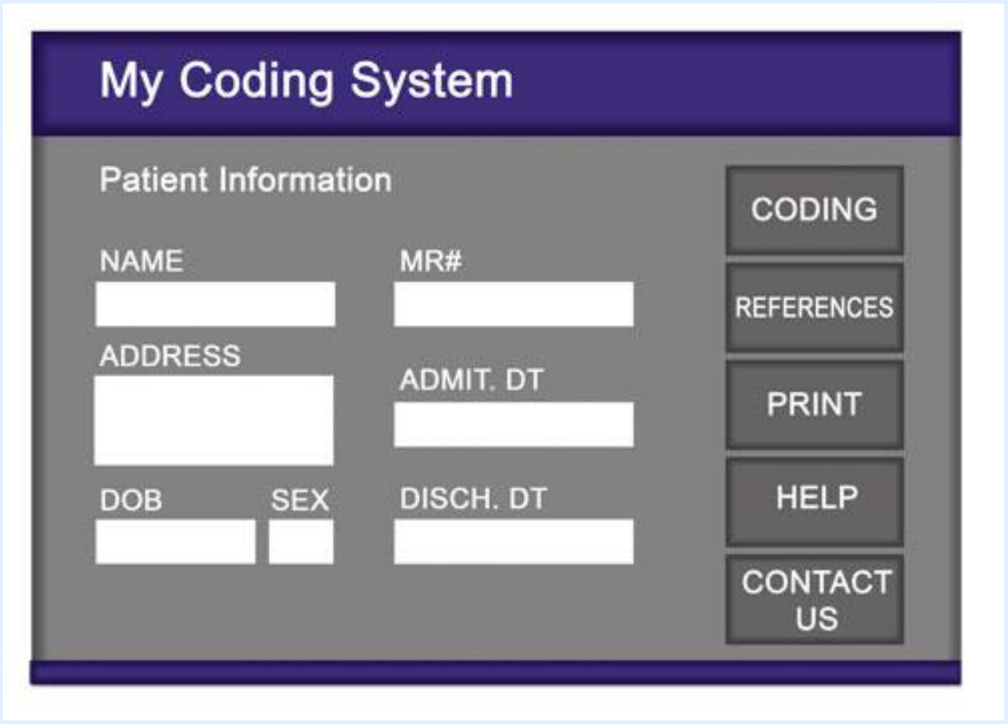
The examples shown below illustrate how the principles of good screen design can be applied to improve software usability and the quality of data captured. The examples walk through a sample coding program, but the principles apply to a wide range of HIM applications.

These principles can be used when selecting new software. They can also help evaluate training needs for current applications or identify causes of errors in current workflows. The illustrations show that even minor adjustments to the design of a program's GUI can make a significant difference in gathering reliable data by guiding the user through the program consistently and providing reference information to help their input choices.

Room for Improvement

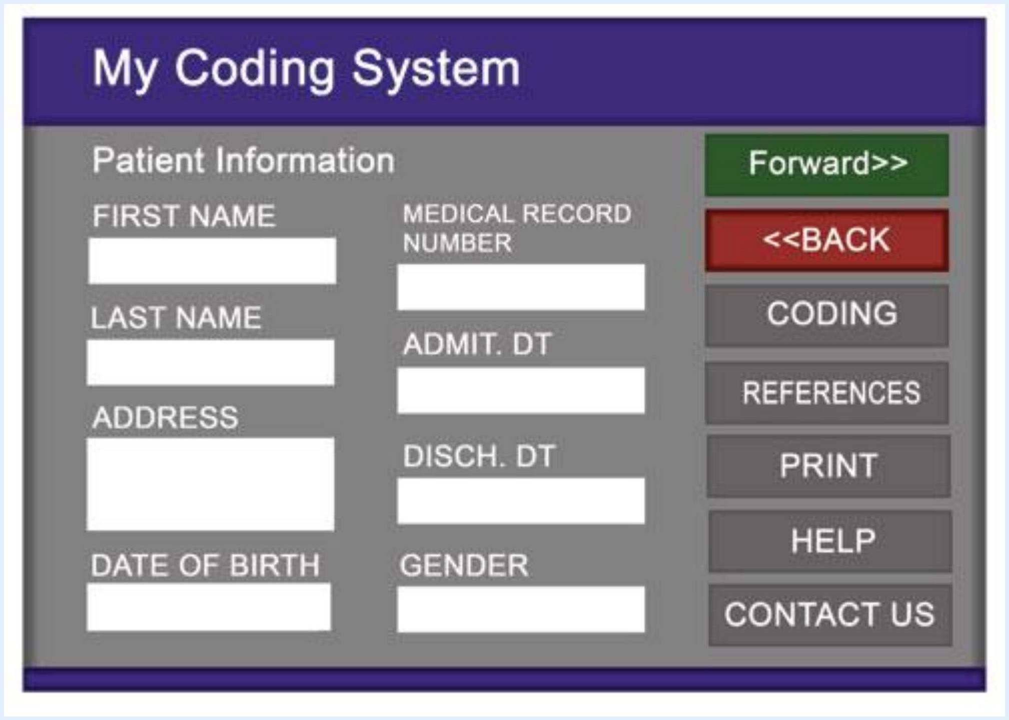
The examples here show how the principles of good screen design improve usability and boost the quality of data captured by a coding program. The examples are simple in design and do not reflect a fully functioning system. But they do illustrate that even slight adjustments to the design of a software screen can improve the consistency, accuracy, speed, and completeness of data capture. A coding program is shown here, but the principles are universal to software design and can be used to evaluate a wide range of

HIM applications.



Example 1 illustrates a generic coding software screen that would be used to initiate a coding session. At first glance, the items on the screen appear to be clearly labeled, and it is obvious where the coder should enter patient demographic information. However, there is room for improvement in the design.

Note that navigation between screens is not clear. There is no option for accessing the previous screen, and the user must assume that clicking on the button marked “Coding” will move the session forward to the next screen. The fields are labeled with abbreviations, and although most users will understand them, they present an unnecessary risk of confusion and inaccurate data entry.

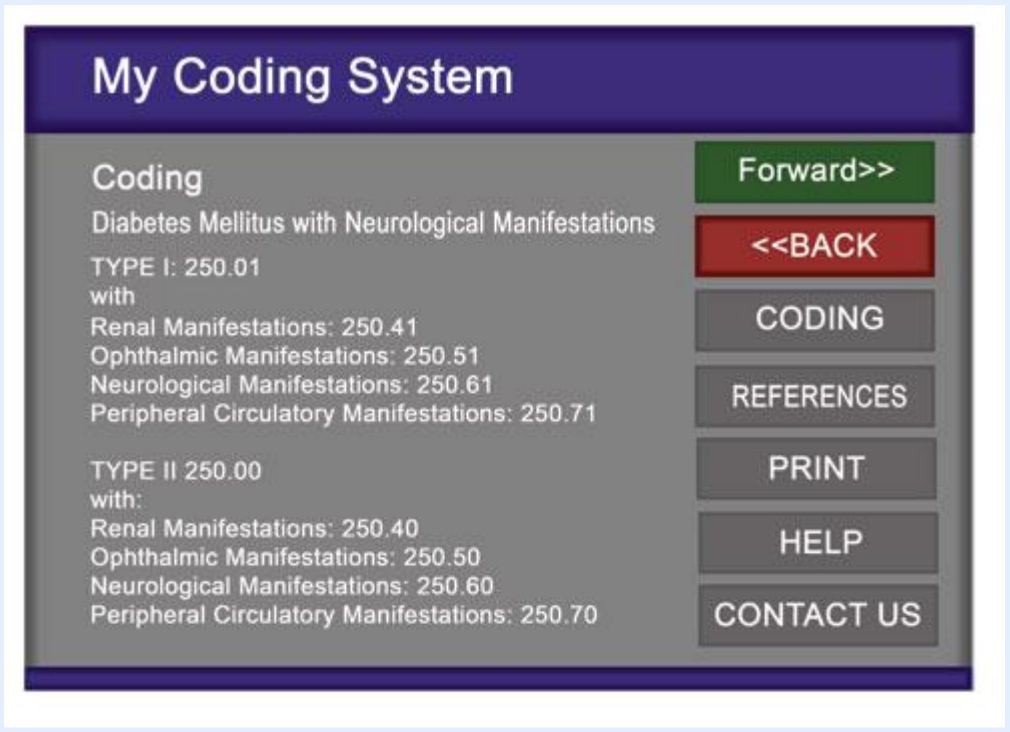


Example 2 illustrates several modifications to the screen in example 1 that improve the software's usability. New buttons clearly direct the coder to the next screen, as well as back to the previous screen. The buttons are highlighted with colors and incorporate directional arrows to help the coder easily navigate between screens.

Data entry fields are further clarified with the elimination of some abbreviations. The name field has been divided into two distinct fields for first and last names to avoid confusion, such as whether the first name should be entered before or after the last name. This enforces accurate and consistent data entry.

A gender field has been added, enabling the software program to alert the user when an incorrect code has been entered based on gender edits. A drop-down menu makes indicating gender a quick process. The "Admit" and "Discharge" date fields now include buttons that launch calendars, allowing the user to select dates, again reducing the risk of data entry errors.

Another feature that improves usability and speeds the data entry process is the addition of keyboard shortcuts. The text label for each button now includes an underlined letter that signifies the keystroke that will perform the button's action. The coder can opt to select the keystroke instead of using the mouse to point and click. Offering both options improves the functionality of the software for users who may prefer one method over the other. With just a few changes to the design of the data entry screen, the software has been made much more user friendly and does a better job of capturing accurate data.



My Coding System

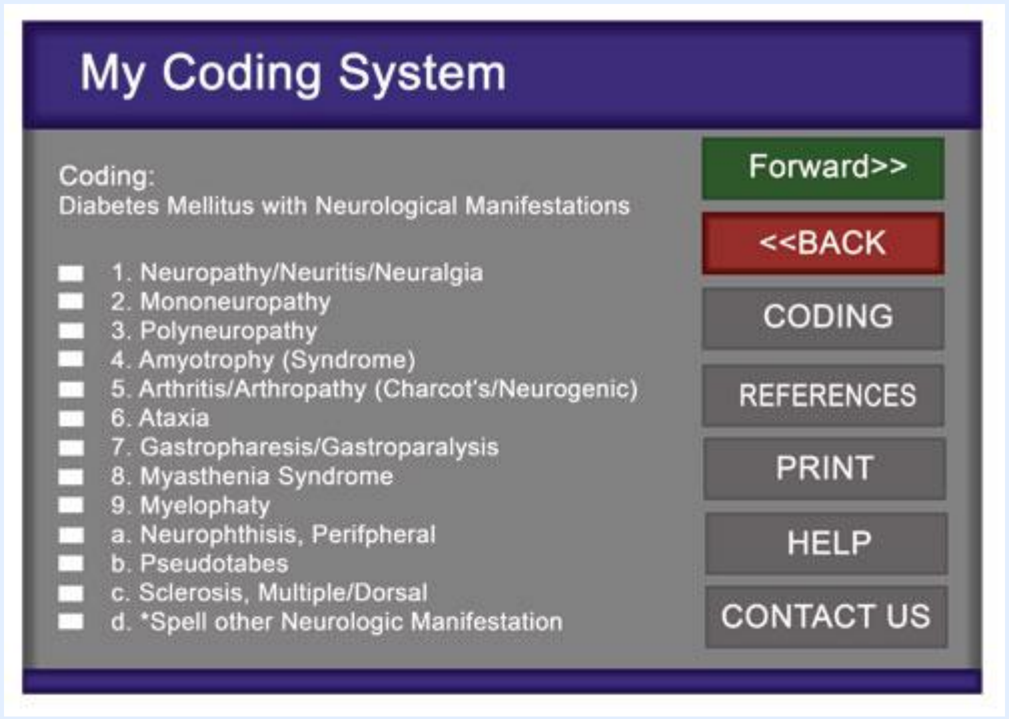
Coding

Diabetes Mellitus with Neurological Manifestations

TYPE I: 250.01
with
Renal Manifestations: 250.41
Ophthalmic Manifestations: 250.51
Neurological Manifestations: 250.61
Peripheral Circulatory Manifestations: 250.71

TYPE II 250.00
with:
Renal Manifestations: 250.40
Ophthalmic Manifestations: 250.50
Neurological Manifestations: 250.60
Peripheral Circulatory Manifestations: 250.70

Now let’s look at the design of a coding screen for diabetes mellitus in example 3. Although this screen occurs deeper in the coding process than the patient demographic screen in examples 1 and 2, the navigation buttons remain in the same locations and are labeled similarly, making the screen design consistent across the different software functions. As a result, the coder doesn’t have to adjust to a different format. All possible codes are provided in a single screen so that the user can easily assess different options and make a selection. Even so, this screen can be improved to aid the coding process and improve data quality.



My Coding System

Coding:
Diabetes Mellitus with Neurological Manifestations

- 1. Neuropathy/Neuritis/Neuralgia
- 2. Mononeuropathy
- 3. Polyneuropathy
- 4. Amyotrophy (Syndrome)
- 5. Arthritis/Arthropathy (Charcot's/Neurogenic)
- 6. Ataxia
- 7. Gastropharesis/Gastroparalysis
- 8. Myasthenia Syndrome
- 9. Myelophaty
- a. Neurophthisis, Perifpheral
- b. Pseudotabes
- c. Sclerosis, Multiple/Dorsal
- d. *Spell other Neurologic Manifestation

Example 4 shows how further enhancements to the software's screen design can assist the coder in selecting the right code for a patient's condition. After first selecting diabetes mellitus with neurological manifestations, the coder is presented with a screen that asks for more specifics about the neurological manifestation and offers a list of choices that will lead to the assignment of an additional code. Prompting for more information about the secondary diagnosis supports more accurate coding and more appropriate reimbursement. The incorporation of the "History" button allows the coder to view the entire series of screens in the session, review choices that have been made, and jump back to a previous screen if necessary.