

---

## Core Value 4 — The Blame Game

---

If you are a computer programmer, consultant, or project manager, you have played the “Blame Game.” To compete, avoid blame at all costs. To win, place blame on anyone but yourself. Let us try a round at the Blame Game together.

Your customer, end user, or boss asks you to implement a new computer application. The first thing you do is survey the users and document the business requirements. Moving forward, you develop a functional design document that highlights what the system will do. After obtaining your user’s sign-off on the back page of the 250-page document, you develop a detailed design document that describes how you will implement the system. At the completion of the detailed design phase, you sigh in relief. At last you can get on with the real work. You begin coding the system.

Coding proceeds smoothly. Of course, you need to alter the functional design a bit as you learn more about the limitations of version ‘X’ of tool ‘Y’. After all, when you presented the design to the user you were sure that tool ‘Y’ could handle vertical buttons with pink dots. Oh well, good thing you caught it in time. At the completion of coding you test the application. It works. You are proud. Now it is time to present the program to your user.

As you proceed through the presentation you wait anxiously for praise. Instead you hear, “This application is not what I asked for. It will not work. What have you been working on for the last 15 weeks?” Immediately you whip out the signed functional design document. You point out that any changes you made were caused by limitations in tool ‘Y’. Sure there are a couple of minor bugs but you can have the bugs fixed by next week.

For the next week you work 16 hours a day to correct bugs in the system and you return the following week for a new end-user presentation. The user takes one look at the system, and before you even get to the parts that you fixed, the user declares, “I’m done. I’m going back to the old system.” What happened? Did this project fail because the end user was incompetent? Didn’t the user sign the specification? Did the user sign it without reading it? Maybe it failed because tool ‘Y’ caused you to change the design. To win at the blame game, squarely place the blame for the applications failure on external factors. Never ever accept the blame.

There is a popular best seller titled, “Men are from Mars, Women are from Venus,” by Dr. John Gray. In this book, Dr. Gray uses the metaphor of populations from different planets to illustrate common conflicts between men and women. He proceeds to explain that many of these conflicts are the result of a different set of perspectives and priorities among the sexes. In essence, our problem is much the same. Programmers are from Mars and end users are from Venus. If we fail to appreciate how different these two groups of people are, we will fail to successfully build systems. Knowing this, how do we proceed?

First and foremost, do not assume that because you have asked a user to read a document it will be understood or even read. This is not because end users are illiterate. More likely, the failure to communicate

complex system specifications is a failure of our skills. Many of us have gone into the computer programming business because of our problem solving and technical skills, not because of our communications skills. While we can write, we may not be able to write clearly and concisely. To ensure clear and complete communications with our users, we should review all written material in design walk-through meetings. At these design walk-through meetings we review the written design documents with our users, page by page. This is time well spent. The more time we spend with the users, the more likely it is that we will accurately capture the system requirements.

Secondly, use pictures whenever possible. Two steps we use to estimate programming efforts are as follows: First, partition the problem into understandable components. After abstracting the components, that is to say looking at the forest and not the trees, compare each component to a similar component you have completed in the past. Once you can compare the new component you are estimating to something you have done in the past, estimating the new effort becomes very straightforward. Likewise, to communicate the design of a new system to an end user that is unfamiliar with software engineering, the problem needs to be abstracted and partitioned. The easiest way to do this is using pictures.

To begin, draw a picture of the overall application. This picture is often called a context diagram because it highlights how the current application will fit with existing applications. You should complete the context diagram as part of your requirement document, which at Metamor we call a Business Function White Paper. Next, partition the problem into smaller pieces. Each piece could be represented by a block diagram, data flow diagram, or entity relationship diagram. These diagrams form a critical part of the Functional Design Document. The process of diagramming your application will help you partition the system into logical sub-systems and then properly abstract them. At the completion of each document, the Business Function Specification, Functional Design Specification, and Detailed Design Specification schedule a walk-through with the user. During these walk-through meetings, review both the pictures and the words.

The next step of the quality design process is to compare your new system design to any legacy systems that it is replacing, or to the manual procedures it is automating. Ensure that your new application provides complete coverage of existing functions or procedures. Do not forget to interview and review your design with the widest possible group of application experts. In many organizations, these experts include the direct users of the application, the user's supervisor, senior corporate management, and any MIS professionals who worked on the legacy application you are replacing.

This brings us to a very interesting question. When, in the design process, should we design the screens and review them with our end users? In the past I have argued that screen and form design is part of the detailed design process. It is part of the 'how' but not important to the 'what'. This argument assumes that all users need to know about each screen during the functional design review process, is what fields will be available on each screen. However, I believe that it is in your best interest to create the screen designs as early as possible. Why? Quite simply, it is much easier for many users to understand how a system will work when they can look at screen layouts instead of lists of fields. As computer technologists we are accustomed to thinking of fields as

representations of data elements. End users are not. In fact, end users often have a limited grasp of the concept of data independent of fields.

The risk of reviewing screens early is that users, after seeing the screens, will assume that the system is almost done. To alleviate this shortcoming I describe the screen layouts as “visualizations.” In essence, the screen layouts are just another picture used to communicate the system with the users.

In conclusion, it is my belief that when a system fails to meet the user's expectations the blame for the failure should be placed on the developers. We are the data processing professionals. The burden of clear communication lies solely on our shoulders. If you are careful to use design walk-through meetings, pictures, reviews of legacy functions and processes, early screen visualizations, and communicate with your user community, you can avoid playing the Blame Game.