



## **Achieving Performance-centered Design**

---

**By John Lovgren  
Kalin and Lovgren Associates**

### **Introduction**

Performance-centered design is a way of looking at the design of systems that focuses on what the person using the system needs to get the job done. In most system design projects all kinds of statements would be made during the design and development of the system. "That can be handled in training." That's a documentation problem." They can get that information from our help system." What all of these statements indicate is that the focus of the design is on the system, not doing the work. If we focus on the work we say things like "How can we make it obvious how to do that task?" "Will people be able to figure out what to do next?" "When we observe people doing this task, what did they need at this point to get it done?" "What does somebody need to know to do this? Can we provide that on the screen?" This change in focus is as significant as the switch from procedural to object-oriented design. It has the opportunity to significantly reduce training costs while increasing the quality of the work being done and the job satisfaction involved with doing the work.

### **Goals of Performance-centered design**

People will be able to get their work done with no training on how to use the system.

The system will provide the right information at the right time to accomplish the work

Reduce time to perform to an absolute minimum

Assume that people may not know the work

Both tasks and systems will be changed as the work is understood

### **How is this different?**

Most of the focus today is on ease of use of the interface. The interfaces are mostly data-centric. This means that the task resides in the head of the person doing the work. They have to know what function to invoke next. If the person doesn't know exactly what the task is the interface will probably not help them get the job done. The concerns for data-centric interfaces have to do with such things as screen layout, labeling on buttons, menu hierarchies, etc. The focus is on the design of individual screens and getting at the data. I saw a

demonstration of an object-oriented design done by a major OO consulting organization. They were very proud of the design principle that you could get to any information or function from one screen. This screen had a large number of buttons to allow for this kind of flexibility. Of course the system provided no clue as to what you might need to do first or when you might be done.

In a performance-centered design the emphasis is on providing support for the structure of the work as well as the information needed to accomplish it. Some examples of performance-centered design exist in commercially available software. One of the most interesting ones is TurboTax ®<sup>1</sup>. It is interesting for several reasons. If you look at the goals of performance-centered design, this software packages meets all of them.

### **People will be able to get their work done with no training on how to use the system**

People trying to do their income tax have no interest in taking any kind of training. They want to get their taxes filled out correctly and quickly, getting all the deductions they are entitled to. These packages, over the years, have moved the interface from a forms based one, where the user had to know what forms were needed to an interview based one that fills out the forms automatically as you answer questions. The design of the interface assumes no particular computer expertise. The controls used are designed to be as self-evident as possible. Also, by making this assumption the designers are free to improve the controls from year to year. The carryover is minimal since most people only use this system once a year.

### **The system will provide the right information at the right time to accomplish the work**

At each step in the process the system only asks those questions that are relevant based on previous answers. The taxpayer is free to ask for more detail or may proceed through a dialog that asks more detailed questions if the taxpayer doesn't know the answer to the higher level question. If a taxpayer says they are married filing jointly then all the questions having to do with other forms of filing will never be seen.

### **Reduce time to perform to an absolute minimum**

The only assumptions made in these packages are that you have the necessary financial and personal information to be able to answer the questions in the interview process. Once you install the software you are able to start doing your taxes immediately.

---

<sup>1</sup> TurboTax is a registered trademark of Intuit

**Assume that people may not know the work**

The interview process makes no assumptions about the taxpayer having ever done this before. Knowledge of forms is not required, nor is expertise on the particular forms.

**Both tasks and systems will be changed as the work is understood**

This is made obvious by looking at TurboTax over time. When I first used TurboTax 6 years ago I found myself using some of the interview mechanism. Often I had to go into the forms themselves. Doing my taxes generally took about 2 days. Each year I found my need to go to the forms to be less and less. Last year it took me about 2 hours to do my taxes and I only looked at the forms when I printed out the final copy. This is an excellent example of the payoff in understanding the work and co-evolving the work and the system to meet the objectives of Performance-centered design.

**How do we achieve systems with these characteristics?****Understand the work**

If you don't have a good understanding of the work people are doing then there is no way to build a performance-centered system. The techniques we use are referred to as contextual interviews. There are excellent articles about this topic by Karen Holtzblatt. These articles are listed in the bibliography. The intent of contextual techniques are two fold, first is to work with the people doing the work to understand it. You work almost as an apprentice. This technique is then used to produce a set of models. A model of the physical environment, a model of the flow of information, a model of the task steps or flow, etc. These models become the basis for discussion with the people doing the work, the developers, the trainers, the documentation people, and the business owners. The design of the system is driven by these models.

**Embedding the work**

This is the hardest part of the design process. When I first worked with Gloria Gery, one of the originators of the field, I found myself wanting to put all kinds of text in the interface telling the person doing the work what do to next, or how to do this step. Gradually over time I moved to putting those things needed for this step of the task in the interface with navigation to the next and previous steps. In addition there were ways to get deeper if more was needed to accomplish this step. If the steps were not really linear then there were interface constructs such as check lists that might provide both navigation as well as status for what needed to be done and had been done.

## **Prototyping and Iteration**

In doing Performance-centered design you have to constantly take your prototypes and models back to the people doing the work to validate what you have done. I have learned that you never get it right the first time, or the second. It is only through extensive iteration that you can achieve a Performance-centered design. The purpose of prototyping is to get feedback, not to build the system. Prototyping must be done quickly and with easy ways to create multiple versions so that several possibilities can be explored. It is fine to start with paper prototypes, although you must eventually move to screen based prototypes.

## **Design Constructs**

Over time three major concepts have emerged in the Performance-centered design field. The first is the concept of Intrinsic. An intrinsic is a mechanism or interface control embedded directly in the interface. The person using the interface does not have to break the work context to use this control. The next concept is an Extrinsic. These are mechanisms that still support doing the work, but require a context break. The person is using the extrinsic as opposed to doing the work. Cue cards are a good example of an extrinsic. They support the work, but are not part of doing the work. The third concept is an External. Externals are mechanisms that require a complete context switch. Most help systems, CBT tutorials, training classes, and paper documentation are externals. A design goal for a Performance-centered System is 80 % intrinsic. This means the system directly supports the person doing the work as opposed to having to go to other constructs to find out how to do parts of the task.

## **Team Structure**

The team needed to do performance-centered design must have a wide variety of roles. Development, training, UI design, prototyping, as well as performance-centered design. The most critical role is filled by the people who actually do the work. Except for the last role people on the team can fill multiple roles. One of the biggest mistakes made in putting teams together is trying to find people who used to do the work and having them fill the worker role. They have usually forgotten many of the details of the work. They also tend not to remember the exception conditions. These can be critical to having a successful performance-centered design. People frequently forget how to handle exceptions, which can waste time and cause errors to be made.

## **Conclusion**

Success in performance-centered design is seen when the person doing the work no longer thinks about the using the system, because the system seamlessly supports the accomplishment of the work by having the right thing available at the right time. People no longer view the system as an obstacle to overcome but rather as a natural tool in the context of doing the work.

## **Bibliography**

Beyer, H. and Holtzblatt, K., "Apprenticing with the Customer: A Collaborative Approach to Requirements Definition ," Communications of the ACM, May 1995.

Dickelman, G. J. (1995). Things That Help Us Perform: Commentary on Ideas from Donald A. Norman. Performance Improvement Quarterly. v8 n1. p23-30.

Gery, G. J. (Fall 1993). Reengineering Performance Development: The Broader View. Journal of Instruction Delivery Systems. v7 n4. p3-6.

Gery, G. J. (1995). The Future of EPSS. Innovations in Education and Training International. v32 n1. p70-73.

Gery, G. (1991). Electronic performance support systems: How and why to remake the workplace through the strategic application of technology. Tolland, MA: Gery Performance Press.

Gery, G. (1995). Preface to the special issue on electronic performance support systems. Performance Improvement Quarterly, 8(1), 3-6.

Gery, G. (1995). Attributes and behavior of performance-centered systems. Performance Improvement Quarterly, 8(1), 47-93.

Gery, G. (1995). Intrinsic, extrinsic, and external. CBT Solutions, 1995(June/July), 26.

Gery, G. (1995). EPSS: What it means for training. INFO-LINE At Work, 1995(Winter), 1-2.

Holtzblatt, K. and Jones, S., "Contextual Inquiry: A Participatory Technique for System Design," Participatory Design: Principles and Practice. Aki Namioka and Doug Schuler (Eds.), Hillsdale, NJ: Lawrence Earlbaum Pub. 1993.

Holtzblatt, K. and Beyer, H., "Contextual Design: Principles and Practice," Field Methods for Software and Systems Design. D. Wixon and J. Ramey (Eds.), NY, NY: John Wiley & Sons, Inc. 1996.

Raybould, B. (Feb. 1995). Making a Case for EPSS. Innovations in Education and Training International. v32 n1. p65-69.

Stevens, G. H. (Feb. 1995). Designing EPSS Tools: Talent Requirements. Performance and Instruction. v34 n2. p9-11.

**About the author**

John Lovgren, Partner of Kalin and Lovgren Associates, is an expert in performance-centered interface design and software and Web usability. With over 28 years in the areas of software architecture and interface design, Lovgren helps companies improve website and software usability through performance-centered design techniques and methodologies.