

Performance Support for the Next Millenium: A model for rapidly changing technologies in a global economy

Gary J. Dickelman
Ashok Banerji

Consumers of popular desktop software have been amused, appalled, occasionally assisted, and most likely infuriated by a *Planning Wizard*, an *Office Assistant*, or other manifestation of an interventionist help apparition. If not, then they have encountered a *noninterventionist* form of assistance, such as a standard context-sensitive help file. These are the manifestations of years of developments in computing where the ultimate objective is to create systems that people can use with ease. Yet we are far from achieving the goal. Learning technologists, systems developers, human factors engineers, and others have focused on creating the precise support that a user (performer) needs at just the right time. The problem is seductive, as a significant gap always seems to exist between the features offered by software and a performer's ability to harness them. Those responsible for developing either the core applications or the ancillary elements of support share a common goal - ease-of-use - but opinions and techniques for achieving the goal vary widely.

In 1989 the concept of *Performance Support* was introduced to address ease-of-use and related learning and performance issues. While definitions vary, it is widely agreed that Performance Support Systems:

- enable people to perform quickly because they provide integrated task structuring, data, knowledge and tools at the time of need;
- do not tax the performer's memory, nor do they require performers to manipulate too many variables; and
- enable task completion with learning as a secondary consequence.

We will review a collection of existing performance support models. From these discussions we will abstract the kernel of performance support in the form of a new model. An approach to realize the new unified model will be shown indicating how it will serve the many stakeholders and stand the test of technological challenges in the global business environment of the new millenium.

Models for Performance Support

A number of models of Performance Support have emerged since 1989¹, all of which address some attributes of the performance problem space. They include:

- The Component Model;
- The Integration Model;
- The Learning Model;
- The Human Factors Model;
- The Hypertext Model;
- The Design Model; and
- The Funnel Model.

The models represent many distinct approaches that practitioners have developed to realize the performance goals.

It is our opinion that no one of these models adequately addresses seminal performance issues. Also they are not sufficiently extensible to continuously meet the needs of the rapidly changing technologies in our

¹ We have inferred the nature of these models from project experience, case studies, informal discussions, and the literature of the respective disciplines that comprise the Performance Support community.

computer-mediated workplaces. Still, each model has great merit in expanding our perspectives from the merging views of diverse disciplines that comprise the Performance Support domain.

Component Model

The Component Model is among the earliest and was expressed by Gloria Gery (Gery, 1991). The model is expressed in terms of the components that provide support to the performer. They include: (1) Applications Software, (2) Glossary, (3) Help Systems, (4) Infobases, (5) Intelligent Monitor, (6) Interactive Advisory Systems, (7) Interactive Reference, (8) Interactive Training, and (9) Video Databases. These components interact with the performer via the "Intelligent Monitor," which helps maintain context and enables tasks to be completed when stimuli are presented.

While the Component Model gives direction to the issues of Performance Support, it falls short as a model because, very simply, *the components are not the "thing."* Many early examples of Electronic Performance Support Systems (EPSSs) contained these components to accomplish the performance agenda. But the converse is not necessarily true: depending on the task at hand, an excellent EPSS may have none of these components. Donald Norman points out that when software includes such components it is usually an indicator of poor interface design (Norman, 1998). If proper user experience and usability engineering is applied to interface design, the result can be (and often is) an EPSS containing *none* of the components.

Developers often infer from the Component Model that an EPSS must have some or all of the components. This contributes to losing site of the performance goal, losing site of the performer, and/or actually *adding* complexity. Because of its heavy reliance on abstraction from EPSS tools of the late 1980s and early 1990's, the Component Model also fixes the underlying system architecture in the GUI genre of Apple and Microsoft. With the advent of browser-based interfaces in the Inter- or intranet world, the component architecture needs to be revisited (Nielsen, 1995 and Norman, 1998).

Integration Model

The Integration Model evolved through the mid-90s as practitioners recognized the benefits of addressing performance issues early in system design and hence embedding it directly into applications. Therefore effectiveness is measured this model by the degree to which the support can be integrated. Three terms have evolved - *intrinsic*, *extrinsic*, and *external* - to denote, respectively, the embedded support, the added support that has some explicit connection to the underlying application (albeit an afterthought), and the *unconnected* afterthought. Examples include an excellent user interface (intrinsic), context-sensitive help (extrinsic), and printed job aids (external). Intrinsic performance support is often the result of *performance centered design* being applied to the systems development lifecycle, whereas extrinsic and external support are labeled EPSS.

The Integration Model is perhaps guilty of a process / product fallacy. It reflects the discord that often exists between performance / learning technologists and systems engineers (i.e., the training department versus the IT community). While this model goes further than the Component Model in focusing the performance issue, it remains flawed as it perpetuates the notion that support is something separate from the application.

Learning Model

The Learning Model views performance support in terms of learning elements only. Typically, two approaches are followed: (1) the Computer-Based Instruction (CBT) is rendered more granularly to reflect specific task support and/or (2) the CBT is placed closer to the performance event. In this respect, the Learning Model provides mostly focused extrinsic support, albeit often external.

The Learning Model falls short as a comprehensive, extensible archetype because it views EPSS as separate from the work-enabling application, and overlooks many performance-enabling disciplines (e.g., human factors engineering) and techniques.

Human Factors Model

The Human Factors Model is based on applying the discipline to interface design through usability testing and inspection methods (Nielsen, 1993 and 1994). While the Human Factors discipline is necessary to ensure performance, it (like the Learning Model) is not sufficient. A likely outcome of an EPSS created with a Human Factors model is a *user-centered* system but not a *performance centered* system. Similarly, systems developed via goal-directed™ design, as advocated by Allan Cooper, are not necessarily performance centered (Cooper, 1995). Systems developers often engage the Human Factors community too late, and with too little emphasis or commitment. Developers often seek usability *validation* rather than formative evaluation through the usability engineering process.

Hypertext Model

The Hypertext Model derives from the notion that primary support vehicle is extrinsic EPSS in the form of a robust “help” facility. Many standards have emerged, such as conventional context-sensitive help, HTML help, cue cards, wizards, and assistants. We speculate that approximately 80% of current Electronic Performance Support Systems fall into this category.

An extremely positive aspect of the Hypertext Model is the discipline it imposes on information organization, search, retrieval, and maintenance (McKnight et. al., 1993). As the world of technology continues to make its Internet transition, the more useful the Hypertext Model becomes in terms of supporting performance. At some level, every interface is composed of information objects that are linked explicitly or implicitly, both structurally and semantically (Dickelman, 1997). But without a heavy emphasis on the Human Factors, the ability of hypertext structures to support performance becomes limited or even counter-enabling. With the proliferation of HTML authoring tools, many hypertexts have evolved according to an acknowledged principle that because the HTML hammer exists, almost every word is a nail. Applying the Hypertext Model is often motivated by the lack of scope or ability to integrate support.

Funnel Model

The Performance Support Leadership Council, a consortium of EPSS practitioners who, developed the Funnel Model. It views performance support as a sequence of “filters” or lenses applied to the work space that successively focus the user on the performance target. The appealing aspect of this model is that it addresses the outcome (performance target) and the development process. While conceptually pleasing, it falls somewhat short in terms of telling us how the filtering process occurs.

Synthesis

A common thread throughout all the models discussed above is the notion of supporting task completion (work performance) at the time of need and shedding the tired, ineffective practice of training people in advance of doing. Each of the models address the now famous statistic / anecdote that about 80% of what we learn about our jobs is accomplished on the job, not in pre-job training (Gery, 1991). All of the models address some important aspect(s) of performance support and the diversity of disciplines involved in the development process. Further, each of the models demonstrates practical successes in their applications. Still, none, in our views, get to the core of the performance issue. Further, there exists a dangling issue of ownership: IT/IS? Learning Technology? Human Factors Engineering? Ownership, and hence sponsorship, remains unclear.

Unified Model

We therefore suggest a *Unified Model*, based on the notion of a Human Activity System (Banerji, 1999). To begin constructing this model, consider the defining characteristic:

- Human activity is an essential component;
- Tasks are computer-mediated;
- Activities are routed through the computer;

- Successful task completion depends on the permeability of the barriers that exist between the person and the task.

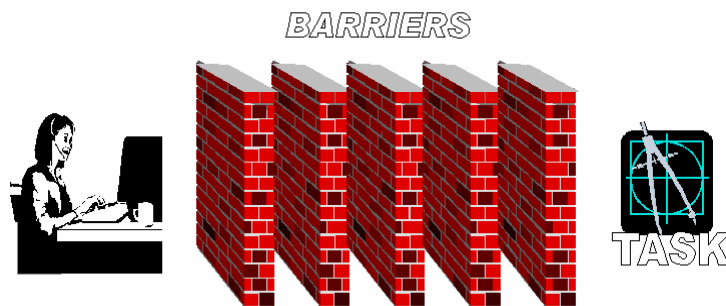


Figure 1: Barriers to performance

For the Unified Model, we define Performance as the absolute difference between the ideal state (excellent task completion) and the actual state (that which a performer accomplishes):

$$\text{Performance} = | \text{Ideal State} - \text{Actual State} |.$$

Measuring the quantity $| \text{Ideal State} - \text{Actual State} |$ requires monitoring the person and the state of the system. In this context, the gap (and therefore performance) is measured in terms of time, quality, and/or error. Barriers are defined in terms of:

- the gaps that exist in the performer's knowledge of the work/problem space;
- access to relevant information and data;
- the ability to make decisions, an understanding of work processes;
- the ability to establish the goal; and
- the ability to solve problems within the work domain.

Barriers are removed by constructing appropriate supports, which can be found in the products of diverse disciplines, as the multiple models of EPSS suggest. The disciplines include Business Systems Development, Data Modeling and Processing, Management Information System Design, Executive Information System Design, Decision Support, Expert Systems, Artificial Intelligence, Computer-Based Learning Systems, Multimedia Systems, Process Modeling / Re-engineering, and Virtual Representation, to name a few. Tools typically generated by these disciplines include:

- Process Support Tools
- Decision Support Tools
- Infobases / Reference / Hypertext
- Knowledge Management Systems
- Workflow Management

In many cases, the outcomes share common goals and functionality. The next step toward establishing the Unified Model, therefore, is to delineate where these diverse fields and tools address the same issues but use different language. The common objective, of course, is Performance Support; the exercise is to consider the development terminology in each field, then normalize.²

In the context of the Human Performance System, Performance Support is comprised of the dynamics around increasing the permeability of the barriers as the performer attempts to accomplish the task. Performance support therefore depends on

² This is left as an exercise for the reader.

- modeling the task;
- profiling the performer;
- measuring key performance indicators; and
- responding continuously with corrective stimuli.

The dynamics of increasing the permeability of barriers involve continuously measuring the human's state with respect to the task. This includes measuring the time it takes the performer to discover a solution plus the time to execute the solution. The support system must then compare the actual state with the ideal state and respond continuously with the right information to reduce the time intervals.

For true Performance Support, the human being represents an additional element of the 'system.' The ability for the 'system' to complete tasks more accurately and efficiently depends on monitoring, feedback, and newly applied interventions. The Performance Support System can therefore be viewed as:

Computer + Software + Human Being

where the human being is assumed incapable of completing the task initially. The feedback loop helps to increase competence continuously. The *total* system performance therefore improves continuously - which means that business performance improves continuously. In this sense, the Unified Model is an adaptive model, with the human being a variable component.

Realizing the Model

The main question for our model is, *How are human actions measured?* The question underscores a fundamental issue reflected in the earlier models: How can we measure or infer action within software without having access to the underlying source code³? One powerful answer lies within the operating system OS. Human actions within software can be represented as events according to the *messages* that are sent between the presentation layer and the application layer of the underlying software. These messages are the means by which user actions are communicated to the software logic.

Measurement in the Unified Model is therefore accomplished by comparing the messages generated by the performer with a pre-documented expert model of the application (also based on OS messages)⁴. When a gap is measured, the EPSS can then respond dynamically to the performer's specific need. Further, the message model allows for continuous profiling of performers. Thus the Unified Model provides a means to not only enable performance initially, but also to refine and improve performance continually.

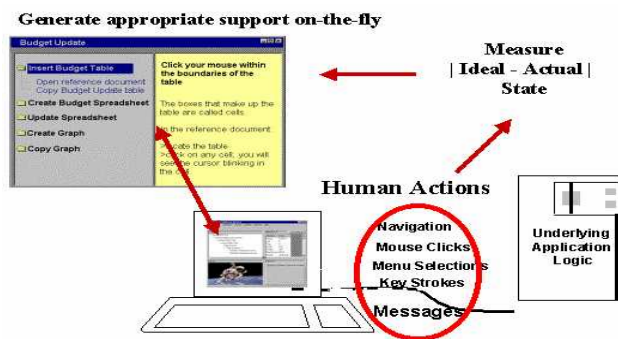


Figure 2: Measuring gaps via OS messages and event

The Unified Model provides a framework in which the various disciplines can influence performance. For example, a variety of support can appropriately be provided for a given performance gap to accommodate audience diversity (e.g., learning style, personal preferences). Learning Technologists therefore have the

³ Performance technologists are most often engaged *after* the underlying software is developed.

⁴ Event and message models are possible to construct and monitor in any operating system.

opportunity to embed nuggets of CBT to respond to specific gaps, whereas the Hypertext Engineering might provide an appropriate infobase. Usability engineering techniques can be used to position the various supports appropriately within the context of the Unified Model structure.

As our technological world migrates to the Internet, we find that new coding techniques - including dynamic HTML - coupled with message modeling and monitoring are supported by the Unified Model (Nielsen, 1995). The Model is independent of interface type, operating system, and system architecture. It is simply predicated on software events and message modeling. Performance is supported via monitoring and profiling, presenting appropriate interventions, decreasing the permeability of barriers that exist between the human and the task, and therefore increasing performance.

Conclusion

The Unified Model proposed in this paper embraces the many diverse disciplines that have traditionally comprised the field of EPSS. It is a model for the next millennium as it provides a performance-oriented framework, is inclusive of the evolutionary models, it synthesizes the vocabulary of EPSS constituents, and introduces a dynamical system that sees the human as a variable element. As such, the human is monitored and responded to so that the system behavior and performance increases continuously as well as adaptively.

References

- [1] Banerji, Ashok (1999), Performance Support in Perspective, Performance Improvement Quarterly, in press.
- [2] Cooper, Alan (1995), About Face: The Essentials of User Interface Design, Foster City, CA: Programmers Press / IDG Books Worldwide
- [3] Dickelman, G.J., Gershon Rides Again: Guidelines for creating web sites intended for use by human beings (Article, CBT Solutions Magazine, April 1997)
- [4] Gery, Gloria (1991), Electronic Performance Support Systems: How and why to remake the workplace through the strategic application of technology, Boston, MA: Weingarten Publications, Inc.
- [5] Laurel, Brenda (1993, 1991), Computers as Theater, Reading, MA: Addison-Wesley Publishing Company
- [6] McKnight C. & Dillon A. & Richardson J., (1993), Hypertext: A Psychological Perspective, Ellis Horwood Limited, Chichester, West Sussex, England
- [7] Nielsen, Jacob (1993), Usability Engineering, Boston, MA: Academic Press, Inc.
- [8] _____ (1995), Multimedia and Hypertext: The Internet and Beyond, Boston, MA: Academic Press, Inc.
- [9] _____ & Mack, Robert L. (1994), Usability Inspection Methods, New York, NY: John Wiley & Sons, Inc.
- [10] Norman, Donald A. (1988), The Design of Everyday Things, New York, NY: Doubleday
- [11] _____ (1992), Turn Signals Are the Facial Expressions of Automobiles, Reading, MA: Addison-Wesley Publishing Company
- [12] _____ (1993), Things That Make Us Smart: Defending Human Attributes in the Age of the Machine, Reading, MA: Addison-Wesley Publishing Company
- [13] _____ (1998), The Invisible Computer: Why Good Products Can Fail, The Personal Computer Is So Complex, and Information Appliances Are the Solution, Cambridge, MA: The MIT Press
- [14] Tufte, Edward R. (1983), The Visual Display of Quantitative Information, Cheshire, CT: Graphics Press
- [15] _____ (1990), Envisioning Information, Cheshire, CT: Graphics Press
- [16] _____ (1996), Visual Explanations, Cheshire, CT: Graphics Press