



A Little DOPSS Will Do You

A New Approach to Training, Knowledge Management, and Problem Solving

by Steven C. Schatz

A great king needed to send a message across the desert—a seven-day journey. He called his advisors—the Vizier of Training, the Guru of Performance, and the Maven of Knowledge. The Training Vizier said, “Oh great one, the answer is obvious. We must make the messenger drink seven days’ worth of water before he leaves.” It was done and two days later, the messenger was dead. The Performance Guru said, “Ah, this shows the Trainer is lacking. You see, we must encourage the messenger to drink eight days’ worth of water by putting in some nice flavors and serving it in a lovely glass.” Two days later, the second messenger was dead. Up stepped the Maven. “It is obvious! We must gather all the water in the area and put it in a large holding tank!” The king asked, “And do what with it?” The Maven looked puzzled. At this point, the third messenger spoke up. “My lord, before your other esteemed counselors try to force me to drink that tank, perhaps we should just buy a canteen.”

The need is often not *more* resources, but the *right* resources, delivered when the user needs them. Training and knowledge management (KM) have limits that are inherent and fundamental. Of course, training and KM are both essential and powerful solutions in many situations. However, in a knowledge economy, where job roles and requirements change often, we must be able to offer new kinds of interventions. We need canteens—resource delivery systems that allow customization of information, functions, and contents *by the users*.

The Trouble With Training Is Training

It used to be that jobs were relatively stable. Needs and required tasks would stay the same while people came and left a job. In such a situation, training can be very useful. With static or slowly changing skill sets, the need is to train new employees to meet the needs of the job. In this circumstance, there is time to develop the training and to produce a good return on investment.

However, when the information needs for a job change quickly, as with many jobs today, training is not so useful. When the job involves a series of problems and those problems change regularly, training is at a disadvantage. There is a built-in delay when developing training. Before a solution is in place, developers must recognize a need, measure the performance gap, allocate budget, meet the design and production schedules, train the trainers, distribute the materials, and complete evaluations. It is increasingly rare to have the luxury of such an extended time between need and solution. With jobs that change with every project, training is often outdated before it can be completed.

Training is intrinsically a reactive solution. Think of the purpose of training: It is an attempt to send people down the correct paths, which have been blazed by experts who have gone before. The problem is that with a job that is in near-constant flux, by the time the solution becomes known, the need for training is often moot. While it is useful to share the solution in the event that a similar situation arises, it would be extravagant to develop formal training to support this sharing. In addition, when a new path has been blazed, others need to know *now*, not in several months. In these cases, the need for understanding supersedes the need for instruction.

The lack of time for those with critical knowledge creates another problem. A subject matter expert (SME) is required for the development of most training. An expert needs to decide the *what* of training while a designer focuses on the *how*. This works only when jobs and tasks remain relatively stable over time, because then there are SMEs available to guide training design and development. In rapidly changing situations, the only experts are the people actually performing the work, developing the solutions. They do not have

time to be SMEs. They solve problems and move on to the next set of problems.

Knowledge Management Isn't, Doesn't, and Won't

To develop solutions to problems, we often seek first to learn from the experience of others within our organization. Training is too slow, but perhaps KM systems hold the key. Or perhaps not.

Knowledge management has become a generic term for a great many ideas.

The term knowledge management is used loosely to refer to a broad collection of organizational practices and approaches related to generating, capturing, disseminating, and applying knowledge. Developing new knowledge, sharing knowledge, combining existing knowledge, and valuing knowledge are all part of what has been termed knowledge management, but emphases and interpretations differ... (MacMorrow, 2001, p. 382).

Prusak classifies KM projects into three groups: knowledge visibility, knowledge infrastructure, and knowledge culture. While a great deal of interesting and useful work continues on the nature of knowledge and organizational means of nurturing knowledge, too often the focus of KM is on the systems. They are easy to see, demonstrate, and show, so management sees where the dollars are being spent. Referring to an estimated 80% of the Global 1000 that have knowledge projects and 68% of the Fortune 100 that have defined knowledge projects under way, Prusak warned: "Quite a few of these projects include work focused on data warehousing, data mining, installation of Lotus Notes, building intranets, and developing document and intellectual capital applications—all of which bear a definite but somewhat distant relation to knowledge" (1999, pp. 3–4). These are KM *systems* and there are problems with looking to these systems to make up for the lag time in developing training.

There are three major problems with KM systems:

- Emphasis on warehousing, rather than use and retrieval.
- The one-size-fits-all outlook.
- Such systems do not manage knowledge, they manage information.

Let us look at each of these considerations.

Problem One: Just a Warehouse

As Prusak points out, many KM systems are little more than data warehouses. Information technology departments may run these warehouses with more interest in the technical challenges of how to collect and store this information than in its effective use. This system-centric design focus results in elegant systems with piles of information, but little utility for users trying to find answers to problems.

When systems are built in such a manner, it is rare for an unsophisticated user to be able to find what he or she needs in a timely, useful fashion to solve a problem or support performance. These systems are designed from the perspective of the *object*; that is, the focus is on how to capture and describe each document, from the standpoint of a single document. A more helpful approach for problem solving is a design based on users and, ideally, their processes and problems.

User-centric design is almost always more useful in terms of solving real workplace problems. However, it is tempting and easier to merely identify what kinds of information exist regarding a knowledge object, gather as much of it as possible, and tag the object with this information. Defining organization problems in terms of process and results, and designing a system that can address those needs, is almost always more difficult to accomplish. The focus of these systems needs to be on users, on their problems, and on what they need to solve those problems. Discovering and understanding this type of information is a skill performance technologists have honed. We need to participate in the design of KM systems. We cannot increase the performance of a person who is trying to dig a ditch with a spoon without addressing the problems of the design of the tool being used.

Problem Two: One Size Fits All

When a system is designed with a focus on the technical challenges—on the system and the objects instead of the end users' requirements, the problems the system will solve, and the type of solutions are needed—the end result is often a system that “works” for everyone, but does not help anyone. I have been in meetings where information technology developers have proclaimed, “Of course we can be all things to all people.” This belief highlights a critical difference between system-centric and user-centric or problem-centric design. If I am building a support tool for the Zowee 2000 mobile phone, the result will be different if I try to build a single tool to serve customers, technicians, and marketers instead of building a different tool for each group. For example, marketing personnel using the generic tool to find differences between the Zowee 2000 and the Zowee 2010 are going to be frustrated if they have to look through repair manuals developed for technicians. Of course, technicians will be upset if they are trying to find schematics and instead find sales literature. After a few such ineffectual searches, the users will solve the problem themselves—they will ignore the support tool. Unfortunately, this is often the fate of KM systems. They exist, but they are not used.

There is an added problem of meta tags for objects in these generic systems. Because the systems are built for everyone, the objects in the system are described for all uses and all users (in theory). This means there are extensive, but still rather vague, descriptions of the objects. Some tagging

schemata have 50 or more tags that can be added to each object, describing the object for retrieval. Describing an object using a tag set can be very time consuming. These are not easy decisions. The person tagging the object needs to view and understand the object. Even with a simple tagging set, such as the Dublin core (<http://dublincore.org/documents/dces/>), which only has 15 elements, making decisions about how to tag a knowledge object is both difficult to do and difficult to do with uniformity among the people tagging the objects.

For example, a common tag is author. Consider a web page with a picture of a famous statue. Who is the author? The owner of the website? The builder of the site? The photographer? The sculptor? There is no easy answer and no easy way to ensure that the way one person decides to tag the object will be the same way that others will tag it. As might be imagined, deciding on the tags for objects can be a challenging and time-consuming (i.e., expensive) proposition. Because of this lack of uniformity, when users try to find objects based on tags, they often find inappropriate information. For example, if I want a diagram of the Zowee 2000, a diagram means different things to me if I am a marketer (who can do on-the-spot upgrades), a customer (who wants to see if my current flash memory will work in the new model), or a technician (who wants to troubleshoot a bad component). So how should the person who is tagging an object decide, in a generic system, how to tag a diagram?

Developing a unique system for each group of users who share homogenous information needs can lessen this problem. By developing tag sets for the needs of individual groups, it is possible to develop a tagging set that is small and specific, and thus effective and relatively inexpensive to implement. However, when systems developers who want to fit specific users' needs to their existing templates build KM systems, then the one-size-fits-all approach means that no one is wearing clothes that really fit. Where we want a suit, we get a toga; it wraps around any size, but will not really meet our needs when we meet with the CEO.

Problem Three: Information, Not Knowledge

KM systems do not manage knowledge—they manage information. This is not merely semantics. There is an implicit promise in KM that a person will somehow be able to bring knowledge from a previous time and place and apply it to his or her problem and, as with vanishing cream, the problem disappears. *This cannot happen.* Knowledge is an event, an interaction *in time*, between information and mind(s).

The best KM system can provide the information and perhaps a summary of the circumstance. This is certainly better than nothing, but it is not knowledge. It might be better termed KL, knowledge leavings. Like the leavings from

horses and elephants after a parade, it is in no way as exciting or effective, but it may provide some stimulus to growth.

An Answer—DOPSS

There remain areas where training and KM are essential solutions. My contention is there is now a need for an additional class of interventions that I call DOPSS—Dynamic Online Performance Support Systems. These systems are online, computer-based systems that are custom-made for specific populations facing specific problems.

A group I am involved with at Indiana University has been working on developing systems for supporting the performance and problem solving needs of individuals sharing similar information requirements. Its projects have been focused on developing methods for design and implementation of DOPSS. For one DOPSS developed for maintenance workers involved with a single naval aircraft, work focused on methods to decide what functions were required to meet the needs of the target population. While this system was initially envisioned as a generic KM solution, onsite interviews and observation pointed to a greater need for communication tools and tools to aid in the location of personnel resources throughout the organization. While every group will have different needs for a DOPSS, this work has focused on developing design methods that can be used for all groups to decide what functions initially meet the group needs.

Within our projects, our team emphasizes that these are *initial* sets of tags and functions. Because DOPSS are dynamic systems that evolve with use, we expect the function and tag sets to change with time. The goal for the early design is to make a system that is both useful now and will guide the changes needed to make the system continue to evolve and remain useful. While each system must have a different function set (different communication tools, different approaches to matchmaking, different ways to add and retrieve objects) and a unique meta tag set depending on the needs of the target group, we can provide some fundamental guidelines that we believe will be common to all DOPSS.

What Is in a Typical DOPSS System?

The three categories of functions in most DOPSS are performance objects with unique meta tags to aid in fast and effective search and retrieval, communications tools, and matchmaking/connectors.

Performance Objects. An essential function of a DOPSS is the ability of the users to add, search for, and retrieve performance objects. It is useful to think of performance objects as a category of objects that may be either learning objects or information objects. Note that this distinction is merely an aid to understanding and designing the system. The two cat-

egories (described below) are not rigid and the system does not treat them differently. The difference is primarily in the intent of the object.

The system grows based on use; it is not a static system. Users direct its growth not only through suggestions, but also by having the power to add objects. This eliminates the time lag required for intervention by a designer or anyone else.

Learning objects seek to teach. Whether a document, print-based guide, workbook, animation, PowerPoint® presentation, video, or interactive web page, these objects aim to provide instruction. Learning objects may have, but do not require, an evaluative element. Within the work at Indiana University, we have adopted a design guideline that encourages learning objects to be chunked into very small bits. DOPSS are designed to support performance. They are not training systems. When someone wants to know how to format a letter, one does not want an entire word processing operating manual or a long tutorial on all the features in, say, Microsoft Word™. We encourage pieces to answer specific “How do I...?” types of questions.

Information objects are far more common and often will make up the lion’s share of the system. Information objects can include such content as shipping rates customized for a particular location, required forms, and other objects that help in the everyday functions of the job. However, most important will be objects brought in by users as the system grows and develops. A standard procedure when building other performance and instructional systems is to build all the content before the system is launched—developers try to discover what users may need and put that into the system.

By contrast, a DOPSS starts with enough information to encourage use. However, it is designed to grow by users and for users *as users wish*. Users working on a problem may request information, which will be tagged and added. Significant communications in chats, email, or message boards can be copied, added to the system, and tagged for effective retrieval. It is an important difference that users are able to add objects into the system. The users may also ask for information or documents to be located, tagged, and put into the system by the DOPSS support staff (or “gardeners,” as will be discussed on page 34).

Communications. The second major class of functions will be the most likely to vary, depending on the needs of the target population. It is difficult to overemphasize the importance of communication within these systems, for if the communications tools are useful and used, it will be an easy matter to capture new understandings as they occur, tag them, and put them into the system for search and retrieval. There are three general areas of communications tools:

- Synchronous two way, such as instant messaging or chat functions, which can include application sharing and

other real-time interactions (AOL and Yahoo both have messenger-type tools).

- Synchronous one way, such as webcasts, where a single speaker makes a speech to users via the web. These tools usually allow display of slides and voice either via a conference call or over the web.
- Asynchronous, such as message boards and listservs.

A design decision that guides this work is to view the DOPSS as a support tool, a custom help desk available when performers have a problem, not as a system that is constantly competing for desktop space. If a person is involved in a group problem, he or she may monitor a discussion board within DOPSS. If an important email is received, the information may be brought into a DOPSS and tagged for retrieval. A DOPSS is a system to support problem solving and performance. Its purpose is to allow users to create, file, and retrieve information that will further that purpose.

Connections. An essential function of DOPSS is the means to connect users with those who have the expertise or the authority to answer questions. When developing functions, we look for a “killer app”—an application or function that is so compelling users will be driven to use the system so they can access that function. From that compelling function, the users may try other functions. In one project, instead of immediately developing the knowledge portal that was requested, an organizational directory that allowed users to find experts for the myriad obscure parts still used was recommended. In jobs that are changing quickly, there often has not been time for someone to create an object that can provide training or guidance. In such situations, the need is for locating a person who has the necessary knowledge within a company. There is a famous saying at Hewlett-Packard: “We need to know what we know.” In a DOPSS, this connection function may be as simple as a directory and as advanced as the “gardeners” playing matchmaker based on request by users, as described below.

Design Consideration: A Gardener

While not a technical function, the role of gardeners is essential to the ongoing effectiveness of this type of system, not only because of the functions they fulfill, but also because of the underlying design decision implicit in planning for the role. Too often, technical systems are built, launched, perhaps debugged, and then summarily abandoned by designers. They are not systems that the designers will actually use. This must not be the case with a DOPSS. These systems need a human hand—an advocate to help the system grow based on the needs of the user. We cannot possibly design a perfect system, even with extensive interviews and observation.

To encourage the system to grow in the ways users desire, we introduce a continuing role we call the “gardener.” This name

was chosen to reinforce the difference between the old approach to designing technical systems and a garden, which requires some work to both ready and maintain. The needs of the DOPSS will change based on the needs of the users: Tags will change, information will be requested, old information will be archived, help questions will lead to changes and improvements. This characterizes a dynamic system, one that is continually growing and changing to reflect the changing needs of the users as they face new problems.

The gardener role is not a short-term position. It is an essential part of the system’s growth. We anticipate that the DOPSS designers will be gardening partners with the client organization’s staff during the first implementation period (planned to last six months), will begin to codify the gardener’s duties during a second period (three to six months), and will turn over the gardener role to the client organization following that.

Groups That Could Benefit From a DOPSS

It is crucial that each DOPSS system be customized. The function set is decided based on problems and needs expressed and observed. The needs of the group also decide the unique tag set for search and retrieval of the objects. This critical function allows fast and effective searches. In addition, because the tag set is designed for the needs of the group, it is small enough that users will be willing to take the time to tag and add objects. For example, if a sales group has developed a new, effective technique, a member may post this with tags that allow any salesperson worldwide who is trying to sell in a specific situation to find this solution. It is essential that users populate the system with information. In this way, DOPSS allow for training and information sharing to be the responsibility of those at the forefront doing the tasks.

While many types of groups could benefit from this approach, developing a DOPSS takes an investment and commitment of time, resources, and money. When looking for a group that may benefit from DOPSS, look for the following characteristics:

- Homogenous information needs.
- Dispersed either geographically or temporally.
- Easy access to a networked computers.
- Jobs that rely on problem solving in a quickly changing environment.
- Ability to operate under time constraints.
- Cost of labor that is high enough to justify the investment in DOPSS. The largest expense in a DOPSS is the observation and analysis required to decide functions and tags. The larger the target population, the more time and people required to do this front-end analysis. The technical expenses, such as developing the tag set and search/retrieval tool, as well as setting up the communications tools, are usually stable.

Some examples of groups that meet these criterion and could benefit from a DOPSS include marketers working on a worldwide product launch, lawyers working on a large case, researchers in different locations, bidding teams working on selling large systems over time, government agencies attacking a specific problem—such as security—and software development teams. There are certainly a great many more.

A DOPSS can be targeted to any size population as long as it has similar information needs. We are currently developing a DOPSS for a group involved in financial planning. While there are 28,000 individual performers, they all use the same filing system for their documents and have questions that fall into the same classifications. Therefore, it is possible to design a system that allows members to add documents and easily and quickly tag them so that others can just as easily find them. Members will be able to support each other in problem solving and performance improvement. Similarly, there were hundreds of maintenance workers in the DOPSS we designed for the U.S. Navy, but they were all focused on the repair of a single aircraft.

Final Thoughts

So far, we have developed and tested methods to select functions and tag sets within two DOPSS systems and are in negotiation to develop a complete DOPSS. As our knowledge grows, it is essential that we share our experiences with design, development, and implementation of this new class of interventions so we may all learn from each other. We emphasize that while the technical tools are all well-established, this is a different approach to design and performance support. It is a solution based on user needs, not on technical feasibility. It is filled with objects tagged and added *by users*. It grows based on evolving user needs. It places performance technologists in new roles—facilitating learners sharing knowledge, connecting people and information, and acting as performance librarians

who help companies discover and share what they know. As we design and implement more DOPSS systems, we hope to be able to further develop and test the design model that takes us from observation to developing functions and tags. At some point in the future, it may be possible to offer a self-customizing system, based on a series of guided surveys. However, we need more practice designing, building, and implementing these systems. For now, this new approach to problem solving, training, KM, and team communication offers a new, useful direction for those seeking help that training and KM systems have been unable to provide. 🏠

References

MacMorrow, N. (2001). Knowledge management: An introduction. In M.E. Williams (Ed.), *Annual review of information science and technology* (Vol. 35, pp. 381-423). Medford, NJ: Information Today.

Prusak, L. (1999). What's up with knowledge management: A personal view. In J.W. Cortada & J.A. Woods (Eds.), *The knowledge management yearbook 1999-2000*. Boston: Butterworth-Heinemann.

Steven C. Schatz, Canteen Consulting, helps organizations worldwide with solutions design at the intersection of people, performance, and technology. He is widely respected in both the public and private sectors for bringing a unique, user-centric perspective to projects. His client list includes Sprint, Visa, AMD, Acer Computer, Versity, Coca Cola, the U.S. Navy, and NCREL. His consulting group designs and develops systems that support performance, including web-based training, learning objects, and unique search/retrieval tools. He also presents workshops and seminars for clients worldwide and has taught instructional design and knowledge management at San Francisco State University and Indiana University. He has presented at conferences of the International Society for Performance Improvement, the American Society for Training and Development, Online Learning, Training, AERA, ADLnet Plugfest, ELearn, and Distance Training & Learning. He is completing his PhD in Instructional Systems Technologies at Indiana University and may be reached at schatz@powerstart.com.

Have you recently published a book about HPT or a related field?

Would you like us to consider it for a book review? Or would you like to volunteer to write a book review? If so, please contact Book Review Editor Erika Gilmore at 765.658.2431 or egilmore@oxauto.com.