

## REDEFINING USER SUPPORT: SHIFTING PARADIGMS INSTEAD OF BLAME

*John R. Cordani*  
*Department of Business and Technology Education*  
*Eastern Michigan University*  
*Ypsilanti, Michigan*  
*313-487-4330*  
*be\_cordani@emuvax.emich.edu*

*Kirk P. Nagel*  
*Division of Academic Affairs*  
*Eastern Michigan University*  
*Ypsilanti, Michigan*  
*313-426-5020*  
*daa\_nagel@emuvax.emich.edu*

### PROLOGUE

Effective user support is not a product, but a process; there are stages of technological development every user goes through that need to be more formally acknowledged and nurtured. We will propose five basic levels of users and their needs. Five is not a magic number; we find it a convenient number for general grouping. An argument could be made that there are as many levels as there are users, but it would be specious; conversely, isn't the convention of grouping so many levels of skill sets under the aegis of "user" equally so?

To effectively redefine user support, we stepped outside the bounds of the computer community. Since computers are pretty much mainstream these days, one big redirection necessary is to make user support more mainstream in the eyes of those outside the computer community. We feel it's imperative to let the community of users define what user support is/should be rather than continue to have the user support community decide what level of user to satisfy.

We used two basic tools in concocting our modest proposal: the 80/20 Rule and the Pathfinder's Principle. The 80/20 rule is amorphous, yet any given context immediately defines it. For instance, in the context of application software the 80/20 rule says that 80% of the time 20% of the features are used. In the context of troubleshooting, 80% of the problems stem from 20% of the possible causes. In the context of our proposed user community, 80% of the users get 20% of the support.

The Pathfinder's Principle states that before you can get to where you want to go, you have to know where you are and where you've been. And this gives us a rather clever segue into the meat of our article.

### WHERE WE'VE BEEN

In the good old days, computing was a shared scarce resource. There was precious little difference between users and support staff. Users were highly-trained, primarily quantitative, technical resource people in their own departments, while user support staff

were concerned with managing the physical resources available--disk space, job execution, etc. User support staff were the gods of the system who dictated protocols to the users--demigods--while both types were pretty much left alone by mere mortals because all this computer stuff was just too weird for mere mortals to even remotely understand let alone use.

Within the computing community, user support was necessarily a team effort; the computing resource was too cryptic to be sufficiently used and understood by any one person. Teams were formed to describe the computer's rules to people who wanted the computer to do work. More teams were formed to explain application rules to people who wanted the computer to do work, but lacked the wherewithal to write their own programs. The perpetuation of these teams was assured by the lack of help within the system.

In the 1980s, there was a shift from shared scarce resources to single, isolated abundant resources in the form of personal computers. Rather than being part of a cooperative collaborative gestalt, the personal computer user became a "microsystem." Each failure of a microsystem had little impact on the community. Several microsystem failures had little impact on the community. Therefore organizational resources devoted to supporting these isolated communities could be even less, since the isolated or individual failures were so inconsequential.

Since communication in isolation is difficult at best, resource decision makers were blissfully unaware of the magnitude and nature of the problems. The size of this isolated community and its problems continued to grow, unbeknownst to the resource allocators, far beyond any resource needs projections which were still based on the traditional shared resource base. What factors could explain such growth, growth that changed computers into commodities?

The growth of computing into a commodity can be explained by following the growth patterns in PC hardware and software. Whichever family of microcomputer chips is examined the pattern is markedly similar. The following table lists both the Intel (I) and Motorola (M) families:

*Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.*

© 1995 ACM ISBN # 0-89791-704-9/95/0010 \$3.50.

<b>Introduction Chip</b>	<b>Power</b>	
1979	8088 (I)	0.33 MIPS
1979	6800 (M)	
1982	80286 (I)	1.2 MIPS
1982	68010 (M)	
1984	68020 (M)	
1985	80386DX (I)	2.5 MIPS
1987	68030 (M)	
1988	80386SX (I)	2.5 MIPS
1989	68040 (M)	
1989	80486DX (I)	20 MIPS
1990	80386SL (I)	4.2 MIPS
1991	80486SX (I)	3 MIPS
1992	80486DX2 (I)	41 MIPS
1992	80486SL (I)	15.4 MIPS
1993	80486DX4 (I)	60 MIPS
1994	Pentium (I)	100 MIPS <sup>1</sup>

Both families have increased their processing power by a factor of about 300 in 15 years. This increase in processing power did not come with a comparable rise in PC costs. Indeed the cost of a unit has remained very stable over time without adjusting for inflation. Adjusting for inflation, costs of PC power have decreased.

This increase in hardware capability has been coupled with an increase in software capability. Most software vendors have used the increased power of the PC to provide a platform for increased sophistication in their software products. The increase in sophistication of software is illustrated by the increased use of complex graphical user interfaces (pre-1985 interfaces on microcomputers were text-based, line-oriented affairs).

With the increased capability of hardware, help followed the trend into graphical interfaces, complete with drop screens and cascading menus. Every type of software followed this trend. The emergence of the GUI approach to interfaces in the PC clearly shows the trend to increased dependence upon hardware to allow a richer and more "friendly" interface for the computer user.

Over the last decade and a half major developments in software have been concentrated in the microcomputer area. Software houses, with the goal of penetrating the single user market, scrambled to curry favor with the mass market of PC users. Developers of the 80s and early 90s strove to design mass use programs which allowed the user to operate the computer in isolation, without the need for central resources.

Users then began to demand hardware/software computer systems

which they could operate independently. System design was driven by mass market appeal, with a corresponding depth and breadth of isolated systems. Microcomputing had moved out of a niche market for the technologically elite and into the commodities market. As a commodity, users demanded systems which they could purchase, install, and operate largely by themselves. And vast numbers of systems were purchased as commodity items, and lo, these systems became vast numbers of isolated microsystems.

In the 1990s, linking these isolated microsystems back into the fold of a shared resource base via networking brought these "inconsequential" failures back into sharp focus for centralized resource allocators. Not only had the number of users sprung unexpectedly on these resource allocators, but the expectations of these users had qualitatively changed. The user expected the machine to behave as a commodity--it is supposed to function flawlessly in task performance. Because of the marketing and associated conventional wisdom of the new "user-friendly" machines, users perceived performance failure as the fault of support staff. Although this perception is seriously flawed from the point of view of the computing community, in one fundamental and overlooked way, it IS entirely the fault of the computing community.

Willfully or not, by being isolated from the large number of microsystems, the help/support function has remained primarily a reference source for the technologically adept.

## WHERE WE ARE

Today, computing is a household commodity. Shifts in hardware capability, increases in software performance, price/performance ratios, and the relativistic ease of use have all contributed to bringing computing to everyday folk. But where the technology is more accessible than ever before, using it seems, to today's casual user, just as arcane as it always was.

There are many reasons for this. Technology always seems to change faster than people. The support function (along with hardware requirements) has never received a level of support commensurate with system demands. Resources for providing documentation have at best been given grudgingly--if at all. Too often, those given the task are considered the least adept by their own supervisors. Application and product developers are often heard commenting on the pain and drudgery of providing documentation--it detracts from their *raison d'être*, confounds their ability to concentrate on the task at hand, and sullies the pristine elegance of their coding solutions.

And the reactions of the computing community to these seemingly timeless conditions themselves smack of timelessness. Wouldn't life be better if everyone went over to UNIX? If only everyone would become a systems programmer..If only everyone would forget about these toy GUI interfaces and stick with good old command line operations..If only everyone would sit down and write their own login.com's of only several score or a few hundred lines..If only...

We have kept up with hardware changes, we have kept up with software changes, but we have neglected to keep up with the fundamental changes within the user community. We still, if not consciously at least subconsciously, expect and demand the same level of skill and competence in users that existed back in the early days.

## Where We Are In The Eyes Of Today's Users

In constructing our new user support paradigm, we spent a lot of time cruising and schmoozing within the user population to get back in touch with where computing is today from a human perspective, not from the computing perspective. We asked people to define user support according to their needs. We asked them to tell us how we could be more effective given the traditional constraints of resource allocation. We asked for honest feedback on all the user-friendly tools at hand.

Repeatedly, one message from this community was clear. Support continued to be technology driven and not commodity driven or, from their perspective, the user served the machine/system rather than vice versa. So it is fundamental to keep computing as a commodity in mind. Mr. Webster defines commodity as "1. any useful thing 2. anything bought and sold...basic items or staple products..."

Commodities in our culture include automobiles and, we propose, computers. These items share qualities which user support specialists need bear in mind. Buying and running an automobile has much in common with buying and using a computer. The new buyer shops for a machine in a dealership, negotiates on features and price, and finalizes a purchase. The buyer gladly accepts a few moments of orientation to the new machine, but fully anticipates turning the key and driving the vehicle away with full use of all its functions.

The owner expects to return to a dealer or service center only when the vehicle needs service or repair. The owner does not expect to spend months learning how to use the automobile. The owner does not expect to become a certified mechanic to use the automobile. The owner does not expect to become an automotive engineer to use the automobile.

Today's computer users bring the same expectations to computing. They do not expect to become programmers. They do not expect to become analysts. They do not expect to become administrators. Users fully expect to "turn the key and stick it in 'D'".

The most important thing we discovered was how sorely we have abused the term "user." It encompasses every possible skill set imaginable--from the peerless to the clueless. And this is the fundamental problem with business-as-usual user support.

To those within this "new" user community, the traditional term "user" functionally defines a tertiary stage of computing development and all the user-friendly tools at hand were functionally designed for this segment of the population--in short, the level that needs the least amount of support. The best example given was what one person referred to as "...the VAX Virtual Help System. If I'm lucky enough to NOT get 'Sorry, no documentation on....' it tells me what something is, how it works, what switches are available, everything except WHAT IT DOES AND HOW TO USE IT!"

These same people gave us the five levels of today's computing community. Before we list them in developmental order, remember that we had asked the population to define us in their terms, not ours. We weren't insulted by the terms and hope you won't be--they are, after all, merely labels of convenience and have served us quite well in reaching the segments of the population that most need reaching. They are:

1. Novice
2. Neophyte
3. Weenie
4. Nerd
5. Geek

Most of our people put themselves in the first two categories and placed support staff in the latter two. Cordani, not surprisingly, was considered a nearly perfect example of a Geek; Nagel was felt to be quite the Weenie.

## HOW THINGS REALLY WORK

There are so many things at play simultaneously it's impossible to give a neat hierarchical analysis. We are talking about cycles and processes with interplay and interdependence rampant throughout. But all can be boiled down to the timelessness of the human condition. There is little difference between the five developmental stages we just listed and the developmental stages of a human: prenatal, infant, child, adolescent, adult. We all started in the womb and gradually, at least physically, grew into adulthood. We all crawled before we walked, and walked before we ran.

But we don't really have the same situation in traditional user support. We have created something more similar to the life cycle of fish. Hundreds of thousands of eggs are laid, with the bulk of them becoming food for others. Of the tens of thousands of hatchlings, thousands survive to become fry. A few thousand fry survive to become fingerlings. A few hundred survive to adulthood. Many of the adults survive by consuming the eggs, hatchlings, fry, and fingerlings of their own species.

Look seriously at how the user community receives traditional support. Novices and Neophytes, unwillingly thrust into the harsh world of computing, find themselves at the untender mercies of Nerds and Geeks who teach introductory computing courses, often on microcomputers. Their help comes from grad assistants who are often barely more conversant with computing than they, or from student assistants who have just completed their own minimalistic introduction to computing. The lucky survivors go on to become new generations of Weenies, Geeks, and Nerds. They in turn perpetuate this cycle because they know of no other way because their teachers knew of no other way because their teachers knew of no other way because there weren't any teachers to begin with. The horrific attrition rate of students, student assistants, and grad assistants seems to bear this out.

If we remove the centralized, traditional support structure from this scenario, it turns out that there is quite an effective underground support structure at play. Novices glom onto Neophytes who glom onto Weenies who glom onto Nerds who glom onto Geeks. Communication in this situation proves to be remarkably effective within adjacent levels.

The structure of this pyramid remains largely untouched by traditional support structures because of the nature of communication between the levels. The further up you are in the user hierarchy, the more removed you are from the lower levels. What is considered basic common knowledge in the realm of Geekdom is considered unattainable by the Lower Realms.

The physical presence of user support exacerbates an already sorry situation. Typically, computing centers are located away from the

bulk of today's computing activity. Remember that today's computing is largely PC-based, distributed physically in classrooms, labs, and offices throughout the organization. With the boom in telecommunications, more and more computing activity is taking place within user's homes.

## WHERE WE NEED TO GO

We'll confess to deliberately using inflammatory rhetoric. Our intent was not really to inflame, but to try to shake the large amount of complacency we find endemic to many traditional support structures. Too many times we have witnessed support staff blaming users for not already possessing the knowledge the users are trying to acquire.

Although we are returning in many ways to the Golden Days of centralized systems in the form of networks, Pandora's box of distributed computing has been opened. From the centralized standpoint, users are dispersed, poorly educated in the art of computing, and operating what they consider a commodity. Users now possess enough power to wreak havoc in a network environment, and often do.

From the users' perspective, they've become accustomed to operating without the "help" of user support--mostly because they've found support to be of little help to begin with and removed in time and place from their needs. If they must participate in a network environment, they expect it to function, at least within their world view, as easily as their GUI machines. They can, will, and do resent the "elegant" command line structure underlying the GUI interface. They don't WANT to know about FTP, telnet, finds, seeks, greps, awks, pings, and fingers. They expect it all to work seamlessly from whatever platform they happen to be on because they KNOW it works seamlessly. In many cases, they've been using "point and click" freeware and shareware packages that do the work for them.

So what do we do? We should forget about hoping for changes in the external conditions affecting the support function and look inwards to how we can change what we have the power to change--ourselves and the way we interact with users.

If allocated resources are going to continue to be just barely sufficient to maintain the centralized hardware and software needs of the organization let alone provide effective user support, then just concentrate on maintaining the hardware and software.

We propose acknowledging, formalizing, and nurturing the already existing underground support structure--we call it SneakerNet--we mentioned earlier. In the eyes of the user community at large, SneakerNet already is the only effective user support that can be relied on due to its very nature. It not only delivers support directly to the site, but a physical presence is constant.

For example, a faculty Geek visits the computing center to solve a specific problem. There will be natural social interaction between fellow Geeks and Nerds. New developments, upgrades, protocols, etc. are hot topics of conversation. After assimilation and, hopefully, problem-solving, the faculty Geek wanders back to the department and disseminates this new knowledge to colleagues and students who are often less well-versed in the nuances of Geekdom. These same colleagues and students belong to the larger community of users who avoid the computing center because they don't understand what the staff at the center is trying to tell them.

They do understand their colleague/professor.

This is the beauty, strength, and elegance of SneakerNet. It is self-supporting and requires no resource allocation. It promotes a type of computing socialism where people do what they like to do.

We need to look at the user community in a different light. We must quit trying to make people fit into OUR existing systems and find ways to support (and reward?) the people in THEIR existing systems to function more easily and efficiently in meeting the real needs of today's user community.

## ENDNOTE

- [1] Rosch, Winn. Hardware Bible, Third Edition. Brady Publishing. 1994. pps 110- 114.