A Better Mythology for System Design

Jed Harris
Pliant Research
978 Cragmont Ave.
Berkeley, CA 94708
+1 510 524 4350
jed@pliant.org

Austin Henderson
Pliant Research
PO Box 334
La Honda, CA 94020
+1 650 747 9201
henderson@pliant.org

ABSTRACT

The past decades have seen huge improvements in computer systems but these have proved difficult to translate into comparable improvements in the usability and social integration) of computers. We believe that the problem is a deeply rooted set of assumptions about how computer systems should be designed, and about who should be doing that design.

Human organizations are continually evolving to meet changing circumstances of resource and need. In contrast, computers are quite rigid, incapable of adaptation on their own. Therefore when computer systems are incorporated into human organizations, those organizations must adapt the computers to changing circumstances. This adaptation is another human activity that technology should support, but our design philosophies are oddly silent about it.

This paper explores the origins of these problems in the norms developed for managing human organizations, proposes partial solutions that can be implemented with current systems technology, and speculates about the long-term potential for radical improvements in system design.

Keywords

System evolution, accommodation, mythology, pliant systems.

PROBLEM

The social value of computing is so great that use of computers in developed countries is already pervasive and well on its way to becoming universal. Computer systems are increasingly part of our interaction with most companies and many individuals; indeed, they shape our experience in these interactions.

From 1980 through 1998, the performance of personal computers has increased by at least a factor of 1000, and the size of typical personal computer applications has increased by about a factor of 16. Communication between computers has speeded up by a factor of more than 400, and changed from a difficult, occasional act to a largely transparent and often continuous process.

However, in spite of the enormous improvements in computer performance and functionality, computers, both personal and institutional, are still very difficult for most people to use, and are becoming more, rather than less, difficult to manage as they become more complex.

Institutional computer systems present apparently intractable problems of ossification, as highlighted by the recent “Year 2000” problem, and the spectacular failure of multi-million dollar projects to upgrade the US air traffic control system, the Internal Revenue Service, and major systems at the Bank of America, among many others [2].

Even when institutional systems are “successful”, they often cause serious problems. For example, while workflows mechanized by computers may be greatly accelerated and made more reliable, they are also largely “frozen”. Because they are often frozen before experience is gained they are often wrong from the start. Because the world changes, the frozen workflows tend to quickly diverge from the real demands of the business process.

Personal computer systems are subject to another sort of ossification. Most people have great difficulty updating their computer systems, or figuring out how to fix them if an update goes wrong. People become very cautious about changing or updating their environment, with good reason. As a result, users are trapped in a network of assumptions and practices that they cannot easily change.

Because computer systems are pervasive, on the way to becoming universal, the social effects of these problems with computing are also pervasive and will be universal. This is a situation that deserves deeper consideration.

Why are we having such a hard time?

Fortunately, there is a research community of computer scientists, designers, cognitive scientists, and others, who are working on the general problem of “making computers easier to use.”

Unfortunately, this research effort is not moving nearly as quickly as the underlying technology. While our hardware technology has improved by orders of magnitude, and our software has grown comparably more complex, the relationship between people (individually or in groups) and computers has only improved incrementally. In some cases, it has even deteriorated.

Why have the huge improvements in computer systems proved so difficult to translate into comparable improvements in the usability (and more generally, the social integration) of computers? We believe that the problem is a deeply rooted set of assumptions about how

Proc. Human Factors in Computing Systems (CHI'99), Pittsburgh (USA), pp. 88-95, ACM Press, 1999.

computer systems should be designed, and about who should be doing that design.

In this paper we propose an explanation for this odd state of affairs, and suggest a research agenda and design approach that can begin to bring our computer systems into better alignment with our social context.

MYTHOLOGY OF SYSTEM DESIGN

Any community, including communities of practitioners, has stories it tells within its ranks about how its activities are conducted. In a community of practice these are largely stories about how the practice is conducted, when it succeeds and when it fails. As with any stories, these are selective and emphasize especially significant or unusual aspects of the practice. They do not directly drive the practice, but they do shape it by helping each participant construct and frame their account of their practice.

We are calling this set of stories the *mythology* of the practice. To understand how our mythology as system designers affects our practice, and how it might be improved, we need to examine our mythology more closely, and briefly examine its roots.

Standard Mythology of System Design

Our mythology of system design takes some basic principles for granted—so much so that they are rarely stated. The following principles are widely accepted:

· Define clear system requirements.

· Define a clean architecture that can meet all the system requirements.

· Define clear choices for users at each point where they interact with the system.

· Maintain consistency throughout the design, for both ease of maintenance and ease of learning.

Underlying these principles are even more fundamental assumptions:

· The parts of the system must interact according to a pre-established harmony defined during its design.

· The job of a designer is to discover, clarify, and when necessary invent the rules that define that harmony, and then embed them into the computer system.

· The users must interact with the system in terms of the language or ontology that these rules create.

Our first—and probably most important—point is that these **are** assumptions, and that they could be **different**. We have some proposals about specific **better** assumptions, but before we get to those, let’s look at how we got our **current** assumptions.

Coordination

People working together are always caught in the tension between their particular ways of understanding the world, and the explicit shared regularities they must maintain in order to coordinate their activities.

We must respond to *particularities*—the details of the case at hand—to achieve appropriate, creative action in response to a changing world and changing goals. These responses can generate unpredictable and unbounded diversity.

Humans in groups depend on shared *regularities* —expectations, norms, conventions, assumptions—to coordinate their activities. To maintain group coordination in spite of adverse circumstances, such as distance, many participants, external hostility, and so forth, we need to support these shared regularities through discussion, teaching, monitoring and enforcement. In other words, the regularities must be made *explicit*. We most often encounter these explicit regularities as various sorts of “rules”, including organizational regulations, rules of thumb, and “manners”.

Particularities and explicit regularities tend to conflict, because individuals can often see ways to respond to particular circumstances that are not anticipated by or compatible with the regularities required for coordination.

Bureaucratic Organizations

Since at least the advent of large-scale agriculture and armies, over 5,000 years ago, people have been struggling with this tension between particularities and regularities, especially in large groups.

Over these thousands of years people have codified this struggle in military, religious, governmental, and business organizations. Relatively recently this continuing codification has produced *bureaucracy* as we know it today. As Joanne Yates has shown,

> During the period from 1850 to 1920, formal internal communications emerged as a major tool of management, exerted toward the goal of achieving system and, thus, efficiency. By the end of that period, control through communication was a fact of life in the workplace [7].

In such a bureaucracy, all “official” activity is viewed as being conducted according to explicit regularities, captured in rules, guidelines, etc. Bureaucrats are trained and monitored to make sure that they “follow” the rules and interpret them consistently. More precisely, they must obey the following key norms:

· Focus exclusively on **identified** regularities in situations.

· View all the particularities in a situation in terms of these regularities.

· Consistently act in terms of pre-defined rules about these regularities.

· Keep records to allow management and enforcement of the bureaucratic norms.

By respecting these norms, a well-run bureaucracy can coordinate the work of huge numbers of people and can minimize corruption, disorder and mistakes.

Computer Systems: Perfect Bureaucratic Tools!

Viewed in terms of the standard mythology of system design, computer systems ought to be the perfect support for bureaucrats. Computers can only work in terms of the regularities they have been built to handle. They can only respond based on the way situations fit these pre-defined regularities. They always follow pre-defined rules, and they do not (at least in the myth) make mistakes. Finally, computers do not change spontaneously, and so they do not require constant management and enforcement of these bureaucratic norms—they obey them automatically.

All of this is no coincidence. Computer design practice and mythology arose from practices such as telephone systems engineering, ballistics calculation and metamathematics which were pursued by communities intensely dedicated to the bureaucratic norms.

PROBLEMS WITH THE STANDARD MYTHOLOGY

It's hard to get perspective on design assumptions embedded in the mythology and practices of such a pervasive tradition. To create a vantage point from which we can see potential alternatives, let's take a more careful look at how bureaucracies actually work.

Bureaucratic norms arose in the process of trying to control people's tendencies to creativity, subjectivity, and "excessive" responsiveness to particularities. These tendencies may be exercised in the service of self-interest, but they may also be simply the result of trying to do a better and more fulfilling job.

These efforts at control are never completely successful, and in fact total success would be disastrous for the organization. The "*ad hoc* elaboration of rules in use" to fit them to individual cases requires a significant level of human creativity and responsiveness to particularities [6, 9]. We call this process *accomodation*. Furthermore, changes in the explicit regularities will be required, and successful change will also demand these abilities.

As a result, real bureaucratic behavior is a dynamic balance shaped by both the underlying human tendencies and the norms. The norms themselves have evolved to make this dynamic balance effective.

Bureaucratic Organizations, Reconsidered

The dynamic balance found in real bureaucracies has several important characteristics that are not captured by the standard mythology:

- Particularities are observed and accommodated.
- The practices of fitting particularities into the rules evolve, changing the interpretation of the rules.
- Practices of rule application are accumulated and codified, often leading to explicit changes in the rules.
- Changes in the rules are designed and implemented. Each change triggers a new round of accommodation as workers adjust their practices to the new regularities.

All of these activities take place in the context of common purposes or missions, which help to guide the accommodation and change, and allow the individual bureaucrats to act appropriately even in the absence of explicit regularities. When members of a bureaucracy do not subscribe to common purposes, the norms can only be maintained by enforcement. Since enforcing obedience to the norms is costly, those who maintain the integrity of an organization must be sensitive to dissent and must make pragmatic tradeoffs to keep the organization viable.

The mission of the organization underpins several further norms that are often important in real bureaucracies:

- Apply the rules with the mission in mind.
- Notice conflicts between the rules and the mission.
- Change the rules so that they serve the mission better.

Computer systems: Less Than Perfect…

Unfortunately all of these additional characteristics of real bureaucracies, and the corresponding demands they place on bureaucrats, are not supported by current computer systems, precisely because computer systems were created using the standard mythology of system design.

In particular, computer systems require their users to map particularities into regularities, and only then can the systems proceed entirely based on the regularities. The systems never deal with the particularities themselves or with the process of accommodation. This has unfortunate consequences:

- Particularities cannot be accommodated by the systems or discussed within them.
- New regularities and difficulties with old ones cannot even be noticed by the systems, since these arise in the process of mapping particularities into regularities.
- Changes are very difficult, slow, and expensive since the system will not notice or accommodate to problems, so all the implications of new regularities must be anticipated by the system designer.
- Most profoundly, computer systems don't share the mission of the organization; all they have is their explicit regularities. The burden of adapting as necessary to carry out the mission falls entirely on users of the systems.

All of this is no coincidence. Practices of accommodation and change are rarely if ever discussed explicitly in bureaucratic mythology, perhaps because the norms are focused on constraining human tendencies, not enabling them. In Yates' extensive study, for example, there is essentially no discussion of the practices through which organizations handled accommodation or change, although some of the technological mechanisms are mentioned in passing [8].

NEED FOR A NEW MYTH

Thus the standard story of bureaucratic rationality that is built into our design practices for computer systems is a story which provides a very partial and rather damaging view of how "good," "rational" organizations work.

Once we recognize the limited and inaccurate perspective on work and technology imposed by the standard myths of both organization and system design, we can start to search for more effective approaches and write better myths around them. That is the focus of the next major section.

However, before we leave our analysis of the problem, let us briefly deal with one seductive alternative.

Smart Systems are Not the Answer

Many have found it very tempting to try to solve the problems described above by making systems "smart"—so that they can notice new regularities, for example. We believe that this is a fatal blind alley. Not only are current "smart" system technologies grossly inadequate to the task, but the mythologies behind many of these "smart" technologies tend to simply reinforce the standard bureaucratic mythology that caused the problems in the first place. They do little or nothing to integrate systems with the mission and values of the organization, or to support the continual reassessment, reinterpretation, and reconciliation that are necessary in a changing world.

Classical Artificial Intelligence (AI) is the source of most ideas about how to make systems "smart." However, AI was founded on the conjecture that a sufficiently large, complex, and well-designed "perfect bureaucracy" could duplicate human flexibility and adaptability. Not surprisingly, it has run into intractable problems that are essentially variations on the ones mentioned above. For example, typical AI systems require examples to be "translated" into an appropriate input format—which conveniently maps the particularities of the examples into the regularities the system is designed to handle. As we might expect, such systems cannot be extended to deal with the untidiness of real situations.

In the last fifteen years, various forms of "soft computing" (for example, neural networks) have been explored as alternatives to classical AI. These techniques can indeed help in some areas where flexibility and accommodation are required, such as handwriting, speech recognition, and recommending books and CDs. However as yet no one knows how to build large systems using these techniques, and in fact we believe that there are deep problems with "scaling" these approaches. As a result, these techniques tend to be used as "rubber bumpers" on systems otherwise built according to the standard mythology.

As we shall see in the next section, a much more realistic and fruitful alternative is not to make systems smart, but to make them better vehicles for the **users'** intelligence.

A BETTER DESIGN MYTHOLOGY FOR TODAY

We believe it is feasible to do much better in designing computer systems if we aim to better integrate human organizations and computer systems, by making the computer system help the humans do the **whole** job, not just the part that fits the standard myth of bureaucracy.

In this section we will explore whether we can move toward such integration without radical changes to the current technology of computing. Perhaps surprisingly, our tentative answer is that we can actually move quite far. In the subsequent section, we speculate on what might be possible with radical changes in computing technology.

To help humans do their whole job, we need to honor the particularities they must handle, the accommodations they must make in nearly every case, the organizational purposes that underpin the coordination, and the process of negotiation and change that keeps the organization viable. Let us examine each of these in turn.

We Must Honor Particularities

The standard mythology simply assumes away particularities that do not fit the regularities. The designer's motto might be "If it's not captured by the regularities, it's not important." As a result, whenever the computer system fails to capture important particularities, users must create parallel records to capture what the system ignores, and then they must coordinate the computer system with their separate records.

For example, in a British printing shop studied by Graham Button [3, 4], a "complete" workflow system was put in place. As we would expect, it could not record any information about jobs beyond its predefined attributes (i.e. its regularities). Of course, these predefined attributes were nowhere near rich enough to capture everything the people working in the print shop needed to remember about a job—in case, for example, it needed to be re-run "the same way." Even if, by some miracle of design, the predefined attributes **had** been adequate, new jobs or work practices would have demanded new attributes very soon.

As a result, the print shop workers maintained the old manual job-tracking system **in parallel** with the computerized workflow system, and coordinated the two systems themselves. The old manual system had a job tracking form similar to the computer system, but in the manual system, the form had **margins**—workers could write notes whenever important features of the job didn't fit into the predefined fields. Unfortunately, typical electronic forms don't have margins, so the workflow system, far from helping the workers, imposed an extra burden.

Fortunately, there are many ways to extend computer system designs so that users can track particularities **within** the system. Simply allowing annotation or "margins" on electronic forms, for example, moves a long way in this direction. Recording annotations and displaying them when the form was retrieved would have greatly aided the workers in Button's print shop. Furthermore, once annotations are captured in machine-readable form, extensions like content search, summarization, etc. become relatively easy to add. Moreover, the accumulated annotations are a valuable resource for suggesting and evaluating extensions to the system's regularities that will allow it to support more of the particularities its users actually encounter.

Margins are only an example, though a very useful one. There are many ways to help users to track the particularities they need to handle, and often they arise fairly naturally from the way a given computer system is used. The key is simply to remember that the system will never be able to capture all the important particularities in terms of its regularities.

We Must Honor Accommodation

Typical computer systems require users to translate particularities into the regularities that the system can handle, and then apply predefined behaviors based on that translation. This translation must produce terms that will lead the system to "do something reasonable." To produce "good" translations, users must "reverse engineer" the system's behavior to come up with encodings that produce the right results. Any particularities that can't be so encoded must be handled outside the system.

In Button's print shop, for example, while workers often ran portions of the same job on multiple machines, the workflow system had no way to describe this, so users had to find ways to convert the actual machine usage, operator time, etc. into terms that the system could handle.

Another example, with more fruitful system design implications, comes from Fikes and Henderson's discussion of the work in an order center for copier supplies [5]. At one point a clerk, attempting to get a "ship to" address for an order, was told that the copier was on an ocean-going barge which called at several different ports, and therefore the address depended on the date the supplies would arrive. Not surprisingly, there was no pre-defined way to fill in the "ship to" field with an address that would produce the correct results.

The clerk invented an ingenious solution on the spot. Instead of a valid address (from the system's perspective), the clerk entered a phone number, and instructions to "Call Bob". This solution highlights several troubling issues, but it also suggests an approach to supporting accommodation.

This solution depended on two things. First, the "ship to" field was flexible enough to accept what was clearly not a valid address. Second, the field ended up being interpreted by a human being on the shipping dock, who could understand the instructions.

If the supply center had been using a "sophisticated" computer system, this accommodation would probably have failed for two reasons. First, the clerk probably would not have been allowed to put the order into the system with a "Ship To" field that the system thought was invalid. Second, the field would probably have been used to automatically generate a shipping label, quite possibly with bar-coded routing information, and no human would have had an opportunity to notice and interpret the instructions.

Note that in this case, annotations could be helpful but they are not sufficient. At a minimum, the system must allow information in essential fields to be incomplete or invalid, and must "call for help" when it needs to carry out some operation that depends on the "invalid" information.

Of course this sort of accommodation mechanism may interfere with the functioning of the system if it is used inappropriately, and it may be subject to abuse. This is just the sort of concern that the standard bureaucratic mythology tends to raise.

However, in reality, building mechanisms for accommodation into a computer system is much more likely to solve problems than to create them. People **must** make these accommodations somehow, unless they are willing to violate or severely limit the mission of the organization. In the print shop it was not feasible to run every job on only one machine, and in the supply center the supplies could not always be shipped to a fixed address. The alternative to system support is for people to cobble together accommodation mechanisms that are completely outside the system, and are effectively invisible from within the system.

If the system supports accommodation, accommodation activities will generally be much more available for audit than the methods outside the system that will otherwise be used. Furthermore, the record of cases where accommodation was necessary can help system maintainers to identify places where more automated support is needed. Overall, accommodation support is likely to produce a net improvement in system performance and security.

Again, even the most basic support for accommodation moves us a long way, but it opens the door to further improvements based on analysis of the accommodation that is actually being done, semi-automatic codification of accommodations, and so forth.

We Must Honor Purposes

In the standard bureaucratic myth, purposes are not **part of** the system. The designers of the system lay down rules, and the bureaucrats follow them. The designers need not explain their purposes, and the bureaucrats don't need to understand the purposes, since they just have to follow the rules. The purposes, just like the rules, are not affected by the activities of the bureaucracy.

Actually, of course, any viable system, including a functioning bureaucracy, depends on broad agreement on purposes, and this agreement evolves over time. Accommodation, for example, is guided by the underlying purposes of the regularities it is interpreting and extending, at the same time that it also extends and reinterprets those very purposes.

In computer systems, purposes show up mainly in the context of help systems, templates, wizards, etc. These mechanisms are designed to support users who have a purpose they are seeking to fulfill, but don't know how to map this desire into the system's regularities. These user support mechanisms provide various types of maps from purposes to system behavior.

These mechanisms typically provide one way communication, from the system designer to the user, consistent with the standard mythology. However this need not be the case, especially in a networked world. Since we are looking for ways to make the system a better vehicle for users' intelligence, we can see that it **should** not be the case. Once we have shifted our point of view, we can see that there are many opportunities to do better.

One useful option is suggested by email support forums, which typically have FAQs (lists of Frequently Asked Questions, and their answers). These forums are typically excellent vehicles for users' intelligence. Users ask questions, and system experts (in many cases more experienced users) answer the questions. Over time, the questions and answers are gradually codified into a FAQ.

If we extend a typical help system with a support forum, and consolidate answers provided in the forum into the help system, we get a mechanism that has the potential to honor purposes within the system. User questions that cannot be answered within the help system become part of an ongoing conversation about the structure of the system and how it supports its users' needs.

As with honoring particularity and honoring accommodation, this type of mechanism also supports the evolution of the system itself.

We Must Honor Change

The most fundamental of these four issues is support for change. Change is extremely difficult to handle effectively in systems designed using the standard mythology, since it is completely outside the mythology. Computer systems designed using this mythology create an impermeable wall between the users of the system—who know "where it squeaks"—and the maintainers of the system—who see only the (by definition) consistent view from inside. The need for change, the reasons for the need and the specific situations where the need arises have to be communicated entirely outside the system, creating another parallel activity that must be coordinated with the computer system, primarily by social means.

As a result of this exclusion of change management from the computer system, the process of changing the system is severely hobbled. Decisions about changes are not directly driven by the needs of the users, so changes tend to be relatively slow and poorly matched to users' priorities. Even more seriously, design decisions are decoupled from the specific knowledge of the users, so design tends to be based on an inadequate understanding of how the system is really used, and how changes will affect users.

The other three areas we have discussed—particularities, accommodation, and purposes—could be considered specific mechanisms to support change. Each one captures information from the users of the system in a form that bears directly on the changes needed in the system. In some cases, such as aggregating accommodations, a computer system might be able to flag and even mechanically abstract needed changes largely automatically.

Undoubtedly, using this perspective, we can come up with many more ways to incorporate the change process into the system itself, and help the users and designers collaborate effectively on evolving the system to better support their joint purposes.

However, the ability to support change in the use of the system is not enough. Change raises the original challenge that drove the development of bureaucracy in the first place: How can the institution ensure that the local changes combine to support the mission and maintain enough coherence to preserve the organization? In the examples above we have left this burden on the shoulders of the system maintainers. In the following section we consider how computer systems can help deal with the tension between change and maintaining coherence.

A BETTER MYTHOLOGY FOR THE LONG TERM

In the previous section, we accepted the basic concept of bureaucracy, although we adopted a new view of what really goes on in bureaucracies. We observed that the standard bureaucratic myths failed to deal with the need for accommodation or change, because they were focused entirely on constraining people's natural tendencies, which might otherwise result in the organization sliding toward chaos. We suggested that once we recognize this, we can find ways for computer systems to help people manage accommodation and change.

In this section, we question the long-term value of organizations as we know them today. Bureaucracy reflects the inherent problems of building large organizations entirely out of humans. Socio-technical systems composed of both computers and people have problems too, but they are very different problems, and we believe such systems can become dramatically superior to any purely human organization. To achieve such superior systems we will need radically new myths.

However, our entire view of the world is filtered through concepts rooted in organizational structures and concepts designed to constrain and regularize human tendencies—from scientific "laws", to the great chain of being, to "canonical" works in literature, art, and music. Such a pervasive framework makes imagining radical alternatives very hard.

To begin this difficult process, we can observe that evolving systems need to maintain both agility and coherence. If a system isn't agile, it cannot respond effectively to local variations or global change. If it loses its coherence, it cannot act effectively as a system at all.

The bureaucratic solution is to maintain coherence by imposing rules that guarantee coherence through pre-established harmony. Bureaucratic logic then requires managing all proposed changes through a central authority (designers, the systems department). This attempt to preserve coherence implicitly depends on underlying human tendencies to provide adequate agility. Unfortunately, as organizations grow, this approach tends to greatly reduce agility, and over time it also loses the ability to maintain coherence, since it becomes incapable of meeting the organization's needs.

Our goal is to find new *pliant* modes of human organization that can sustain high levels of both agility and coherence as they grow. While we are still far from an engineering theory for creating such pliant organizations, we do see four transitions in computer technology that we believe will help us move in this direction:

From Programs to Patterns

Computer systems today typically execute programs which specify processes completely deterministically. The set of programs in a system must mesh according to a pre-established harmony, since the programs have no ability to adapt to each other. This leads to a system that is incredibly coherent while it works, but also incredibly lacking in agility. Furthermore, this coherence is fragile, since systems that depend on pre-established harmony descend abruptly into chaos if the harmony fails.

In pliant systems programs will be augmented or replaced with process descriptions using *patterns*, as described by Christopher Alexander [1]. (Note that this is very different from using patterns to help people design and build software, and will be much harder to implement.) Such patterns specify processes partially and open-endedly. Any actual process is the result of many overlapping and interacting patterns. The interaction of the patterns is not based on pre-established harmony, but is worked out in the process of using them and sometimes requires search and discovery. The system may need to request help if it cannot get its patterns to mesh well enough; conversely, people may intervene to push the patterns into specific relationships.

From Execution to Enaction

In today's systems, programs are executed —deterministically elaborated in a pre-specified relationship to a pre-defined context. Again, this has all the problems of pre-established harmony. In particular, it assumes that the structure of the context for executing a program can be completely known in advance.

In pliant systems patterns will be *enacted*—interpreted by finding a fruitful way to relate the patterns to the specifics of the case at hand. The process of enaction determines what aspects of the case are relevant and how they should be organized to the meet the needs of the patterns being enacted. As a result, enaction can support accommodation much more effectively than can program execution in current systems. Again, a pliant system may ask for human help if it encounters problems in enacting patterns, and people may intervene to influence the way the system enacts patterns in specific cases.

From Monoliths to Collaborating Activities

A computer system based on current technology defines a single, consistent, monolithic perspective which encompasses all of its information and actions. Once again, this provides incredible coherence while suffering from very poor agility and extreme fragility. However, all organizations contain multiple shifting points of view whose interplay is vital to the actual work underway. As a result, the apparent coherence imposed by the computer system is illusory and stands in the way of assessing and responding to the factors that affect coherence in the organization.

Pliant systems will consist of multiple *collaborating activities*—both cooperating and competing—which contain many partially reconciled perspectives that are only consistent enough for the purposes at hand. Enaction of patterns can continue to function effectively in a partially inconsistent environment, but excessive inconsistency will ultimately lead to loss of coherence. To maintain adequate coherence, enaction must also monitor the consistency of the process being enacted, at many different scales, and work to increase consistency if coherence becomes inadequate. Users have partially reconciled perspectives of their own and will need to be active participants in the process of maintaining coherence.

From Design to Evolution

Current computer systems are updated through re-design and re-implementation, resulting in a transition from one universal perspective to another. Designers usually must maintain some compatibility between the old perspective and the new one, to allow existing data and applications to survive the transition. However, in practice, because of the strong requirements of pre-established harmony, implementing changes and maintaining compatibility are often in deep conflict, and sometimes necessary updates become impossible.

In pliant systems, collaborating activities will evolve largely by incrementally consolidating enaction of their patterns. Such consolidation is similar to just-in-time compilation—once an activity has successfully enacted some patterns it can succeed more quickly in future similar cases, and may gain the ability to handle some more difficult cases. If the system needs help to succeed in one case, it may be able to succeed in subsequent similar cases without help. The knowledge acquired through successful enaction of patterns can be recorded by extending and differentiating the patterns themselves. In addition to this incremental consolidation, the system will need to refactor its patterns as it evolves, to prevent unbounded growth in complexity and to improve its ability to generalize to new cases.

SUMMARY AND CONCLUSION

Let us review some broad characteristics of this shift in mythology:

Different Stance

We begin by recognizing that no attempt to fit the world into a neat set of categories can succeed for long. We will always encounter inconsistent, ambiguous, messy bits that don't fit. Standard system design, which depends on making the world fit a neat set of categories, will naturally have trouble.

Co-Production

Trying to solve this problem by making machines "smart" isn't feasible. It's a lot easier and more fruitful to help **people** be smart, and make machines a better vehicle for **human** intelligence. Working together, people and machines can handle problems better than either can alone, but only if they honor each other's strengths.

Multiple Truths

There is no single consistent perspective within which we can frame all the information or activity in any given complex system, much less the wide variety of systems that we encounter daily. Instead, we are always faced with multiple interacting perspectives which cannot be reduced to one another. We need to find ways for machines to help us work with these multiple perspectives.

Dynamic Balance of Change and Coherence

Effective systems must be sensitive to needs for local change and at the same time must maintain adequate coherence, at many different scales. The dynamic balance between change and coherence is driven by, and judged against, each organization's need to achieve its mission.

We Can't Get Out of the River

There is no way for a designer to stand above the fray, as an observer standing on the bank of the river. Even reflection is a part of the action. Ultimately there cannot be a role of designer prior to and distinct from user. We are all in this together, and designers will be more successful if they see themselves as part of the whole socio-technical system.

ACKNOWLEDGMENTS

Early work leading to these ideas was done at Apple Computer, and we appreciate its support. The other members of the team at Apple were Dave Curbow, Alan Cypher, Paul Dourish, Tom Erickson and Don Norman. In addition, we benefited from consulting by Niklas Damiras, Xin Wei Sha and Brian Cantwell Smith.

Since May, 1997, some members of this team have continued to work on these ideas as Pliant Research http://www.pliant.org/.

REFERENCES

1. Alexander, C. The Timeless Way of Building, Oxford University Press, New York, 1979.

2. Flowers, S. Software Failure: Management Failure: Amazing Stories and Cautionary Tales, John Wiley & Sons, 1996.

3. Bowers, J., Button, G. and Sharrock, W. Workflow from Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor, in Proceedings ECSCW'95, (Stockholm, Sweden), European Foundation for Cooperative Work Technology, 1995

4. Button, G. and Sharrock, W. The production of Order and the Order of Production, in Proceedings ECSCW'97, (Lancaster, UK), European Foundation for Cooperative Work Technology, 1997

5. Fikes, R.E. and Henderson, D. A. Jr. On Supporting the Use of Procedures in Office Work, in Proceedings of the First Annual National Conference on Artificial Intelligence, American Association of Artificial Intelligence, Menlo Park, CA, 1990.

6. Suchman, L. *Plans and Situated Action: The Problem of Human-Machine Communication*, Cambridge University Press, Cambridge, 1987.

7. Yates, J. *Control Through Communication: The Rise of System in American Management*, The Johns Hopkins University Press, Baltimore, 1989; pp. xvi-xvii.

8. Yates, J. Op cit. pp. 68, 72.

9. Zimmerman, D. H. and Wieder, D. L. Ethnomethodology and the problem of order: Comment on Denzin. In J. D. Douglas (ed.), *Understanding Everyday Life: Toward the Reconstruction of Sociological Knowledge* (pp. 285-298) Chicago, Aldine, 1970. Cited in Suchman, L. and Trigg, R. Artificial intelligence as craftwork In Chaiklin, S. and Lave, J. *Understanding practice: Perspectives on activity and context* Cambridge University Press, 1993.