PRINCIPLES OF HUMAN-CENTERED INFORMATION SYSTEMS

JOHN J. LEGGETT, PETER J. NÜRNBERG, ERICH R. SCHNEIDER
Department of Computer Science, Texas A&M University
College Station, Texas 77843-3112 USA

ABSTRACT

The paper discusses two main principles of human-centered information systems design – namely, the integration of space and time at a fundamental operating system level. These principles will allow all applications to use the notions of time and space in their designs and will give humans a working environment that is more familiar. A research project based on these principles is described and computing infrastructure requirements are discussed.

1. Introduction

Future information systems will have little in common with those of today. Ubiquitous access to high-capacity computer networks will lead to massive data and process distribution, and entirely new forms of personal and collaborative work will emerge [Leggett et al., 1993]. In the future, information will be produced, transmitted and consumed in electronic form. Our static, paper-based literature with its archaic indexing schemes will be replaced by a dynamic, digital hyperliterature, containing flexible and efficient mechanisms for locating, organizing and personalizing vast amounts of multimedia information [Egan et al., 1991; Schatz, 1993]. Scholarly work of the future will be mediated through coordinated access to shared information spaces. Users will organize their own private information spaces, collaborate with colleagues through shared information spaces and have access to vast amounts of multimedia information in global public information spaces. New media, new data types and ubiquitous access to high-speed computer networks will revolutionize our conceptions of books, libraries, scholarship and learning. The societal impacts are as obvious as they are overwhelming to contemplate.

But what will become of the human in this future scenario? Will we still be at the mercy of systems that have a computer-centered design, or will the human be augmented and empowered with new computing paradigms? Our thesis is that, indeed, a paradigm shift from computer-centered design to human-centered design of information systems should and must occur if we are to obtain the most benefit from future computing technologies.

How will we design such human-centered information systems? The issues and possible technologies are legion. In this short paper, we will concentrate on a few basic principles that allow for the personalization and sharing of scholarly information spaces. We will introduce the notion of integrating space and time at a fundamental level in the operating environment and show how this enables a more natural working environment for humans. We also describe a research project involving these notions and discuss the computing infrastructure necessary for its support.

2. Concepts

A fundamental question in current information systems design is: "Who owns (controls) the machine's resources?" Unfortunately, the most common answer to this question in current designs is: "the computer (or operating system, or windowing system)." Consider the following question. Who owns the areas outside of the windows in a multiple, overlapping window graphical user interface? *We* do not. If we did own that screen real estate, then we could write notes to ourselves outside any application, allowing personalization of our information

spaces. Instead, at present, the windowing system owns this space, and it decides what can be placed there.

While it is true that many applications have the notion of *space* in their interfaces, it is not true that operating systems support the notion of space throughout the computing environment. Yet, this is exactly what we need for the scenario above. Although, as humans, we are used to working in a continuous space, when we work with our computer-centered information systems of today, we must give up this notion and work in confined little boxes that are defined by and convenient for the computer. What we need, of course, is a free reign to use the space provided as *we* see fit – furthermore, we need the computer to maintain these spatial relationships persistently.

As humans, we also work in an environment in which *time* is a fundamental notion. Our telephones record messages with timestamps, our clocks ring bells at a time we set in the future, and we are accustomed to reconstructing and associating events in the past by locating artifacts in our spatial environment [Friedman, 1990]. For example, we remember actions in the past by colocating with the physical objects we were using when the activity occurred.

While it is true that some applications have the notion of time in their interfaces, it is not true that operating systems support the notion of time throughout the computing environment. When we work with our computers we are always working in the present. What we need, of course, is a free reign to move through time as *we* see fit – furthermore, we need the computer to maintain *our* notion of time, its past, present, and future, allowing us to personalize and share our information spaces and our work products.

Why is it that we do not have these capabilities in our systems today? One reason is that we have concentrated more on efficiencies for computers than for humans – that is, computer-centered instead of human-centered designs. A second reason is that the computing infrastructure necessary to implement time and space at a fundamental level in computing environments is not trivial. In the following section, we describe a research project which is investigating these issues and discuss the required computing infrastructure.

3. The MA Project

MA is an example of the type of human-centered design discussed above. MA is a time-based and spatial graphical user interface metaphor. The name was chosen from a philosophy of time in which time does not exist unless an event is happening. This *notion of time* allows one to compress the *representation of time* in user interfaces (see Figure 1). The MA project seeks to implement this metaphor in prototypes of personal and collaborative graphical user interfaces. The term MA will be used in reference to the metaphor, as well as particular prototypical implementations of the metaphor.

3.1 Features of MA

Time. Infinite time exists in disjoint time sequences based upon the occurrence of events. Time flows in a direction and speed specified by the user. Time is represented by space in the user interface. This allows one to refer to the time sequences as time spaces.

Active events will have a current trace in a time space. Completed events will have an iconic representation in a time space. The iconic representation may be inspected to determine attributes such as trace in time space and versions of computational and data abstractions. Time spaces will be compressed for the benefit of the user.

Space. Infinite space exists in at least two dimensions.

Function. Time and space belong to the user. Time belongs to the user in the sense that the user can freely move in time through the past, present, and future. Space belongs to the user in the sense that the user can place any object or cause any event to occur at any location in space.

Moving back in time, the user can inspect previous work. Moving forward in time, the user can set events to happen when the time arrives. When using MA, the user's daily work is automatically archived in time through no additional effort.

Moving around in space, the user is the owner (in a general sense) of all that can be seen in the user interface. The user may freely write notes or annotations or cause events to happen at any location in space. When using MA, the user's daily work is automatically co-located in space through no additional effort.

3.2 Computing infrastructure requirements

MA requires several capabilities to be present in the core operating system, or operating environment, if it is to be implemented in a straightforward manner.

Simple and composite object management. The storage subsystem of the operating system must provide support for both simple and composite objects with arbitrary attribute-value pairs instead of a simple flat filespace.

Flexible and efficient version control. All objects in MA are automatically versioned with respect to time. Some may of course be versioned with respect to other criteria, such as authority. The versioning subsystem must be extremely efficient with respect to both time (since every object is versioned) and space (since many versions of an object may be stored).

Event mechanism with flexible scheduling and notification. Placing arbitrary computational objects, including (as of yet) non-existent ones, in future time spaces requires flexible scheduling. Also, the operating system must asynchronously signal, or notify, other computational objects, including the MA interface, when events (such as new pre-scheduled application startups) occur.

Control mechanism for computational abstractions. Interaction with abstractions representing events or results of computations must be supported, if the placing of events in time spaces is to be implemented in a straightforward manner.

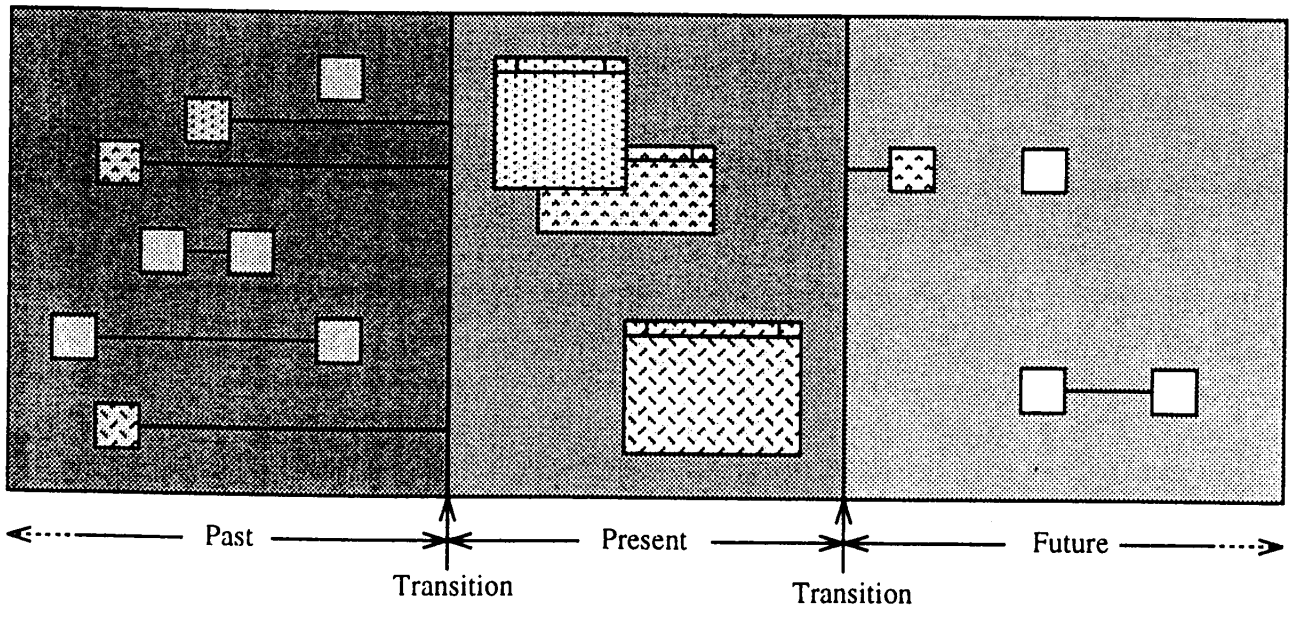

Figure 1. A conceptual view of the MA graphical user interface.

3.3 An Implementation Environment: HOSS

HOSS is a computationally-oriented hypermedia system [Nürnberg et al., 1996]. It consists of a hyperbase layer, a structure processing layer, a metadata manager layer, and an application layer. The main difference between HOSS and other hypermedia systems is that HOSS is an entire operating environment. It provides file system, memory management, and scheduling features. HOSS is best thought of as a hypermedia-aware operating system. An immediate result of this is that HOSS, as any operating system, admits an open set of application processes. Furthermore, just as all applications in a real-time operating system may take advantage of real-time awareness on the part of the operating system, all HOSS applications have immediate access to hypermedia functionality. The entire environment is immersed in this functionality.

3.3.1 How HOSS meets the needs of MA

Various aspects of HOSS can be used to meet the computing infrastructure requirements of MA. These are briefly described below.

Simple and composite object management. HOSS uses a *hyperbase* as a back-end storage layer. A HOSS hyperbase process consists of two threads: a Versioned Object Manager (VOM) and an Association Set Manager (ASM). The VOM provides simple and composite object abstract data types.

Flexible and efficient version control. The VOM thread of the hyperbase also provides versioning support for its object abstractions. Delta storage algorithms may be specified on a per-object basis, allowing delta storage for each object to be optimized with respect to time and space for the appropriate object or media type.

Event mechanism with flexible scheduling and notification. Two HOSS tools, HyperActive and AcTool [Sánchez et al., 1994], provide an interface and infrastructure for defining and manipulating agents, which can be used to schedule events and asynchronous event notification.

Control mechanism for computational objects. Recognition of hypermedia structure by HOSS is what differentiates it from other operating systems. An aspect of this recognition of structure is the explicit control of behaviors over these structures. HOSS schedules such behavior computation more efficiently than a general purpose operating system. The ASM thread of a HOSS hyperbase provides both the structure and behavior abstract data types, allowing the straightforward manipulation of behavioral computational objects.

4. Summary

This paper has argued for a human-centered paradigm for information systems design. To allow for the personalization and sharing of scholarly information spaces, two principles of human-centered information systems design should be followed – namely, both time and space should be integrated at a fundamental operating system level. This will allow all applications to use the notions of time and space in their designs and will give humans a working environment that is more familiar.

Acknowledgements

This research was supported in part by the Texas Advanced Research Program under Grant No. 999903-155.

References

Egan, D. E., Lesk, M.E., Ketchum, R.D., Lochbaum, C.C., Remde, J.R., Littman, M., and Landauer, T.K. [1991] "Hypertext for the electronic library? CORE sample results." *Proceedings of the Third ACM Conference on Hypertext*, (San Antonio, Texas, December), 299-312.

Friedman, W. J. [1990] *About time: Inventing the fourth dimension.* Cambridge, MA. MIT Press.

Leggett, J. J., Schnase, J. L., Smith, J. B., Fox, E. A., Eds. [1993] "Final report of the NSF workshop on hyperbase systems." (Washington, DC, October 15-16, 1992). Department of Computer Science Technical Report TAMU-HRL-93-002, Texas A&M University, College Station, Texas, June.

Nürnberg, P. J., Leggett, J. J., Schneider, E. R., and Schnase, J. L. [1996] "Hypermedia operating systems: a new paradigm for computing." *Proceedings of the Seventh ACM Conference on Hypertext*, (Washington, DC, March), forthcoming.

Sánchez, J. A., Leggett, J. J., and Schnase, J. L., "HyperActive: Extending an Open Hypermedia Architecture to Support Agency", *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 4, (December), pp. 357-382, [1994].

Schatz, B. R. [1993] "Building an electronic community system." In *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-human Collaboration*, R. M. Baecker, Ed., Morgan Kaufmann Publishers, pp. 550-560.