DOCUMENT RESUME

ED 388 221 IR 017 370

AUTHOR Tanaka, Yuzuru
TITLE From Augmentation Media to Meme Media.
PUB DATE 94
NOTE 7p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Augmentative and Alternative Communication; Authoring Aids (Programming); Creativity; Editing; Foreign Countries; Microcomputers; Multimedia Materials; Research and Development
IDENTIFIERS *Computer Users

ABSTRACT

Computers as meta media are now evolving from augmentation media vehicles to meme media vehicles. While an augmentation media system provides a seamlessly integrated environment of various tools and documents, meme media system provides further functions to edit and distribute tools and documents. Documents and tools on meme media can easily replicate themselves, recombine themselves, and are naturally selected by their environment, namely the society of their authors and users. Their accumulation in their users' community will form a meme pool, which will bring rapid evolution of documents and tools. The IntelligentPad architecture provides a standard framework called a pad that works as a meme medium. When applied to microworlds, it will bring more flexibility to the environments and more opportunities of creative thinking to the user. (Contains 10 references.) (Author)

From Augmentation Media to Meme Media

Yuzuru Tanaka
Electrical Engineering Department, Hokkaido University, Sapporo, 060, JAPAN
tanaka@huee.hokudai.ac.jp

Abstract: Computers as meta media are now evolving from augmentation media vehicles to meme media vehicles. While an augmentation media system provides a seamlessly integrated environment of various tools and documents, meme media system provides further functions to edit and distribute tools and documents. Documents and tools on meme media can easily replicate themselves, recombine themselves, and are naturally selected by their environment, namely the society of their authors and users. Their accumulation in their users' community will form a meme pool, which will bring rapid evolution of documents and tools. The IntelligentPad architecture provides a standard framework called a pad that works as a meme medium. When applied to microworlds, it will bring more flexibility to the environments, and more opportunities of creative thinking to their users.

Media for Creations and Evolutions

Today's personal computers are considered as augmentation media that provide us with various tools for entertainment and thought. They provide tools to augment our capabilities. However, augmentation alone can not stimulate our creativity. Creativity requires new media different from augmentation media. What kind of media can stimulate creativity? This question has been motivating the IntelligentPad research project in Hokkaido University since 1987. Creation yields evolution, and no other than evolution can activate further new creation. Biological evolution is based on genes. We require similar genetic media both for the cultural and technological evolution of creative works on computers, and for the educational evolution of their users' creativity. Such media should be able to replicate themselves, to recombine themselves, and to be naturally selected by their environment. They may be called meme media since they carry what R. Dawkins called "memes" (Dawkins, 1976). Their environment here means the society of their producers and consumers, namely, authors and users. M. Stefik pointed out in 1986 the importance of understanding and building an interactive knowledge medium that embodies the characteristics of memes (Stefik, 1987).

Children playing with the toy blocks Lego easily create various types of vehicles, towns, and amusement parks. As G. Tarde, a French sociologist in the 19th century, thought, creations start from imitations in a broad sense (Clark, 1969). Lego has been keeping to provide new customers with various new construction kits for vehicles, towns and amusement parks so that they can learn construction patterns through their playing experiences with these kits. After the imitating stage, they gradually become able to recombine partial imitations of different originals, or further to add their original patterns to these recombinations. The standard connection interface of Lego blocks allows arbitrary combinations of various blocks from different construction kits. Thus, the world of Lego blocks rapidly expands and evolves themselves. Creation and evolution require such meme media that make both the decomposition of existing objects and the composition of new ones as easy, as direct, and as instantaneous tasks as the editing of documents.

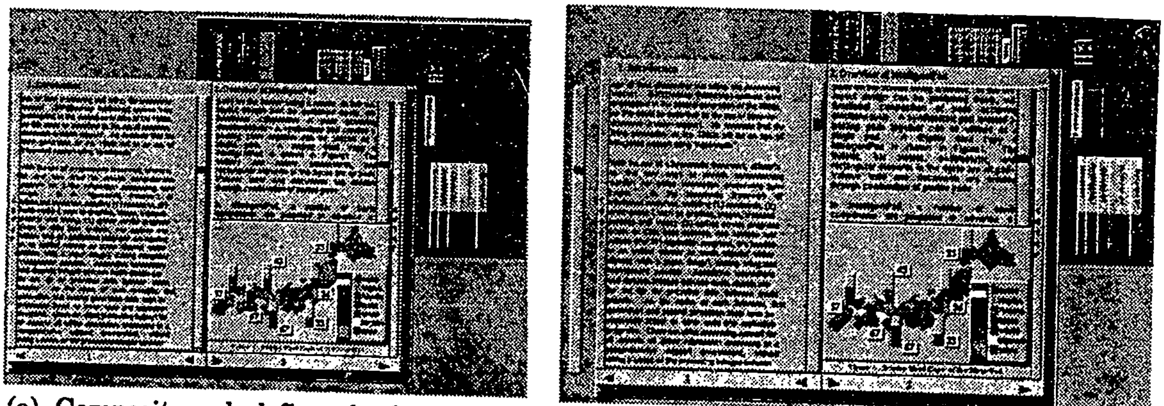
While personal computers have dramatically simplified the editing of multimedia documents, they can not yet allow us to easily edit existing tools to create new tools. The IntelligentPad system architecture that has been developed at Hokkaido University for these 7 years uniformly represents both documents and tools as pads, i.e., reactive media objects that look like paper sheets on the display screen (Tanaka and Imataki, 1989; Tanaka, 1989, 1991; Tanaka et al., 1992). Pads work as meme media for the editing, the replication, and the natural selection of various types of documents and tools.

IntelligentPad as Meme Media

A pad in the IntelligentPad consists of a display object and a model object. Its display object defines both its view on the display screen and its reaction to user events, while its model object defines its internal state and behavior. Wide varieties of intellectual resources can be represented as pads. Among them are multimedia documents, system utilities, application programs, and user environments. Among system utilities are E-mail utilities, file utilities, and database utilities.

In an IntelligentPad system, you can easily compose any document or any tool by directly pasting some pads on top of another. Such a paste operation simultaneously defines both the layout of its components in the composed pad and the functional linkage among component pads. Composite pads are also simply referred to as pads. When the distinction is necessary, pads that are not composite pads are referred to as primitive pads. You may use paste operations in arbitrary ways, for example, to overlay multiple translucent pads of the same size, or to arrange multiple pads on the same base pad. Users can easily replicate any pads, paste pads on another, and peel a pad off a composite pad. These operations can be equally applied to both any primitive pads and any composite pads.

Figure 1 (a) shows a bookshelf and books that are all constructed by pasting various primitive pads. The open book defined as a composite pad shows a text with a scroll bar on the left page and a



(a) Composite pads define a bookshelf and books.

(b) The scroll bar in the text page is replaced with a bar meter in the map.

Figure 1 The editing of an existing composite pad.

map of Japan with several bar meters on the right page. This scroll bar pad and these bar meter pads actually share the same function that detects the mouse location and sends the underlying pad a value between 0 and 1 depending on the detected relative location. Therefore, you may replace the scroll bar with a copy of these bar meters (Figure 1 (b)). When you paste a bar meter on the text pad, you have to connect this bar meter to the text-scrolling function of the underlying text pad. You can specify this connection just by selecting the scroll slot from the list of slots defined by the text pad.

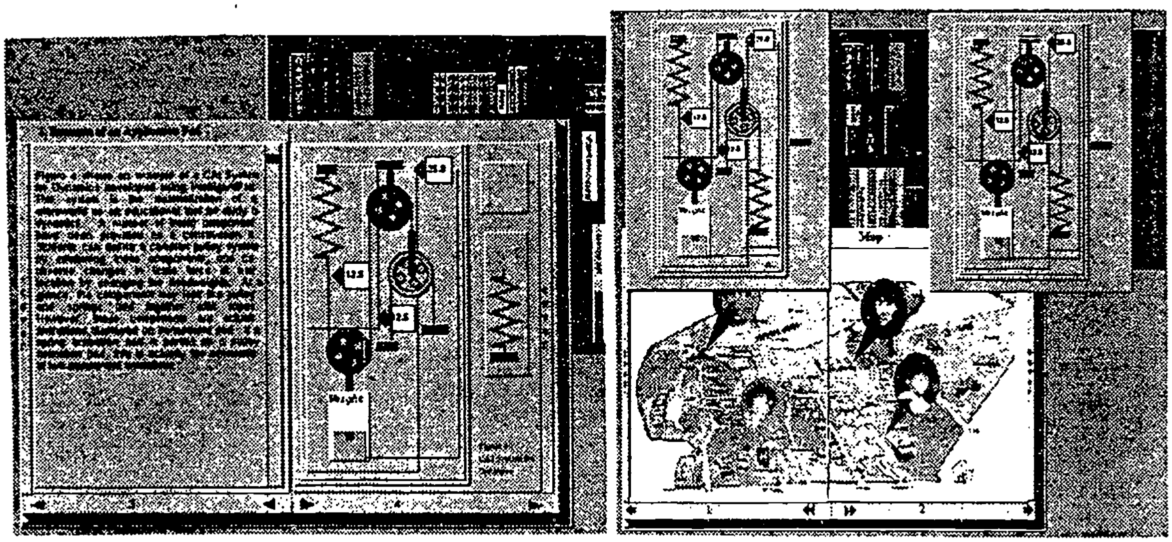
The application-linkage interface of each pad is defined as a list of slots. Each slot can be accessed either by a set message **set** <*slot_name*> *value* or by a gimme message **gimme** <*slot_name*>. Each of these two messages invokes the respective procedure attached to the slot. Each slot s_i may have two attached procedures, $proc_{i,set}$ and $proc_{i,gimme}$. The default for $proc_{i,set}$ stores the parameter value into the slot register, while the default for $proc_{i,gimme}$ returns the slot register value. Its slots and attached procedures define the internal mechanism of each pad. They are defined by its developer.

When a pad P_2 is pasted on another pad P_1, the IntelligentPad constructs a linkage between them. This defines a dependency from P_1 to P_2. The pad P_2 becomes a subpad of P_1, while P_1

becomes the master pad of P_2. No pad may have more than one master pad. If P_1 has more than one slot, we have to select one of them to associate it with P_2. This selection can be specified on a connection sheet. The selected slot name is stored in a standard slot of the subpad P_2 named *connectslot*. A subpad can send either **set** ↑*connectslot* <**value**> or **gimme** ↑*connectslot* message to its master, while the master pad, when its state is changed, can send some of its subpads an **update** message without any parameter to propagate an update event. The up arrow before *connectslot* means that the slot name stored in *connectslot* becomes the real parameter. IntelligentPad allows us to disable some of these messages, which can be specified on the connection sheet. The two messages **set** s v and **gimme** s sent to a pad P are forwarded to its master pad if P does not have the slot s.

Besides the three standard messages, any pad can send some standard messages for geometrical operations to its master as well as to its subpads. Among them are move, copy, delete, hide, show, open, close, resize, and paste messages. The set of these standard messages defines a standard interface between pads.

Figure 2 (a) shows other pages of the same book. Pulleys and springs on the right page are



(a) Springs are pulleys that are connected to each other are animated by transparent pads.

(b) A pulley-and-spring pad is put on a FieldPad, and taken its shared copy. Picture pads on a campus map can send any pads to a priori specified workstations.

Figure 2 A microworld of springs are pulleys and the event-sharing by the FieldPad.

animated by transparent pads. By pasting these pads together, you can easily connect animated springs and pulleys. Two pads pasted together automatically readjust the location of their animated parts so that these parts may work as if they are directly connected. These springs and pulleys constitute a construction set for a microworld (Papert, 1980), where a learner can not only play with a given set of objects, but also decompose these objects and recombine their components to create new objects.

Any pads accept copy and shared-copy requests. Shared copies of the same pad share the same state, i.e., the same model object. The state of a composite pad is defined as the state of the base pad. Shared copies, however, can not share a user event applied to one of them unless it only changes the shared state. A user event may change the pad view without changing its state. For example, it may change only the location of a component pad in a composite pad. Hence, we require an event sharing mechanism as an independent primitive function. In the IntelligentPad architecture, every independent function should be implemented as an independent pad so that its generic function may be applicable to any pads. The pad on the right of the pulley-and-spring pad in Figure 2 (a) works as such an event-sharing pad. We call it a FieldPad since it represents the field of user events. In Figure 2 (b), the pulley-and-spring pad is pasted on the enlarged FieldPad, and the whole pad is duplicated. The two copies share every user event applied to it.

In Figure 2(b), a campus map of our university is pulled out of the bookshelf. It has pictures of several persons at different sites. Each picture is actually a pad clipped into that shape. This clipping facility is also provided as a pad called a MaskPad. When a pad is pasted on a MaskPad, it is clipped into the shape of this MaskPad. Each picture pad on the campus map can transport any pad to an a priori specified IP address. If you want to send one of the shared copies in Figure 2 (b) to one of the persons in the campus map, you can just drag this copy to the picture of this person and drop it there. The copy is automatically sent to the workstation of that person. Shared copies distributed over a network can still keep sharing the same state.

From Object-Oriented Architectures to Media-Based Architectures

Current computers treat various types of intellectual resources. Among them are multimedia documents, system utilities, application systems, and user environments. Unless they can provide a dedicated functional linkage between any pair of these different types of resources, they fail to integrate these resources (Figure 3 (a)). By a functional linkage is meant not only a data communication linkage but also an application linkage through message exchanges. Different pairs require different types of linkages. If there are n different types, we require $O(n^2)$ different types of linkages. This is the essential difficulties of open integration systems that are open to the future addition of new intellectual resource types.

The IntelligentPad architecture solved this problem by separating media from their contents, and standardizing the logical structure and the interface of the media. Media of a certain type in general play their most important role in providing a uniform access protocol for various types of intellectual resources. Books are the most typical example. They have a long history for the development of their common structure starting with a front cover, followed by contents, then a body with hundreds of pages, indices, and then ending with a rear cover. Books are organized in this way to provide their readers with a uniform access protocol. While media of the same kind share the same organization structure, they can contain different structures of information as their contents. The same idea was adopted by the IntelligentPad architecture. Each primitive pad consists of its shell and its content. Its shell defines its standard media structure and interface. It is up to the developer of each pad how the contents is implemented in the standard shell. While the IntelligentPad is based on an object-oriented architecture, it further restricts its architecture. Therefore, we call such architectures media-based architectures to distinguish them from conventional object-oriented architectures (Figure 3 (b)).

In a media-based architecture, only one type of functional linkages is used to connect any pair of shells (Figure 3 (b)). In the IntelligentPad, each shell has an arbitrary number of jacks called slots. It also has a single pin-plug to connect itself to one of the slots of another shell. The shell architecture and the standard linkage facility are provided by the kernel of the IntelligentPad systems. Neither users nor pad developers have to worry about them.

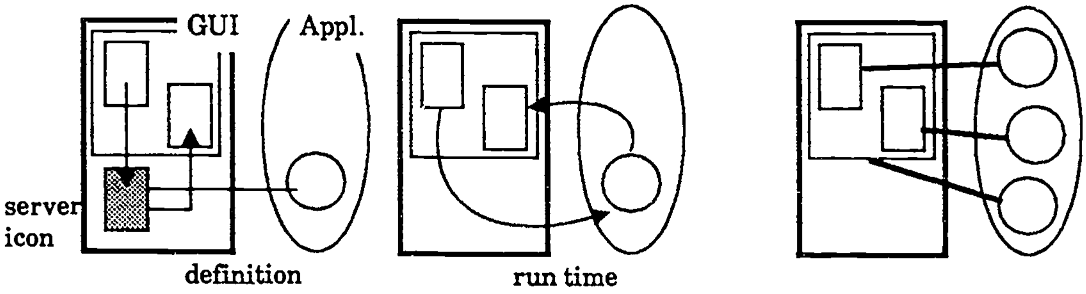
(a) object-oriented architectures (b) media-based architectures

Figure 3 Object-Oriented Architectures and Media-Based Architectures

Pad Synthesis as the Application Linkage through the Embedding

Interapplication communication mechanisms provided by various graphical environments can be classified into the following 4 categories; (1) Cut-and-Paste, (2) Drag-and-Drop, (3) Object Linking, and (4) Embedding. The first two may require no explanations. Object Linking adopts graphical links to interrelate two different application objects. Embedding adopts a document-oriented model. In a document-oriented model, a compound document created by a client application may contain various component objects linked to different server applications.

The embedding mechanisms are further classified into two categories; (1) Widget Embedding, and (2) Media Embedding (Figure 4). In the widget embedding, the link between a component and



(a) Widget Embedding (b) Media Embedding

Figure 4 The widget embedding and the media embedding.

a server application is not fixed, and hence it needs to be defined by the object linking mechanism. Besides, a single server application can be differently linked to more than one components. The NeXT's InterfaceBuilder falls into this category. In media embedding, the link between a component and a server application is fixed and never changes wherever this component is embedded. This fixed pair defines a media object. Besides, in media embedding, the embedding of a media object into another media object defines an application linkage between these two media objects. The IntelligentPad architecture adopts the media embedding mechanism. Suppose you make a copy of the object with some embedded objects. In the widget embedding, this copy can not inherit its original's relations to the server applications and hence loses the original function. In the media embedding, however, this copy inherits its original function.

Only the media-embedding mechanism enables us to edit tools to define new functions as if we edit compound documents.

A Pad Pool as a Meme Pool and Its Search

The IntelligentPad architecture aims at the exchange and the distribution of pads across different platforms, which enables its users society to form an open market of pads. Some pads will be priced, and others will be free. Each user can easily import any pad from an open market into his own environment through networks or via off-line storage media. He may decompose this imported pad, paste its arbitrary subcomponent on top of another owned in his environment, and create a new pad. This new pad can be again published into the market.

No more than 10,000 programmers, each capable of developing only three primitive pads in a year, can add 30,000 new primitive pads every year to a pile of pads in the society. If 50,000,000 people access the pad pool in the society, and if each of them edits pads to create 5 new composite pads every year, we can expect the increase of 250,000,000 pads every year in the pad pool. These pads are genes for the computers and the networks in our society. Pads can be recombined like genes. An addition of a new pad into a composite pad works as mutation. The accumulation of pads in a market forms a gene pool. Useful pads are more widely distributed by the users replicating them for their own use. This is a natural selection mechanism. Pads work as memes. The pad pool shared by our society works as a meme pool, and promises rapid evolution of various types of intellectual resources represented by pads.

Such a society will not face the so-called software crisis. Instead, it will face new kinds of problems. "How can we rapidly distribute among users the intellectual resources they want?" "How can users search the expanding shared pool of intellectual resources for what they need? " These are exactly the same questions we have now in our consumer culture flooded with various consumer products. Users and intellectual resources respectively correspond to consumers and consumer goods. Meme media will bring us consumer societies and consumer cultures of intellectual resources, which requires new services for the distribution, management and retrieval of intellectual resources.

The IntelligentPad architecture provides four kinds of access methods: (1) a visual catalog of pads for browsing, (2) hypermedia networks for navigation, and both (3) form bases and (4) pad bases for quantified retrieval of pads. Any pads can be managed by these four search methods.

This means that we can even store and retrieve CSCW environments constructed with a FieldPad, CAI environments, or computer games, both constructed with a StagePad, as well as documents, charts, tables, and ordinary desk work tools. A stagePad is used to simulate a sequence of user tasks using multiple pads. Form bases cannot meet our needs to manage all types of pads. IntelligentPad provides the PadBasePad to meet such needs. A pad base is a database in which stored records are all pads. In our daily conversation, we specify what we want by its name if we know the name. Otherwise, we partially specify it either by its category, by its substructure, by its context, or by some combination of these. We may describe fish as things in water. This specifies its context. We may describe a calculator as a thing with more than 10 buttons and a digital display. This specifies its substructure. Pad bases allow us to specify the pads we want by any of these different methods.

Remarks on CAI Applications of IntelligentPad

The IntelligentPad architecture works as a meta tool. It provides tool developers with a standard pad framework to program each tool component as a pad, a large library of pads, and an open environment to export and import various pads to and from other IntelligentPad environments. When applied to microworlds (Papert, 1980), it can provide not only tools and objects that their users can easily combine, but also their construction kits, which enable the users to customize or to decompose the given tools and objects, or further to invent new tools and objects. Besides, a user can easily expand his microworld by importing new tools and objects from any other different microworlds.

The IntelligentPad architecture also provides a powerful mechanism called a proxy pad. A proxy pad works as an interface to an external object such as a simulation program running on either the same machine or the different machine connected by a network, a database system, a computer-controlled device like a VCR, a computer controlled measurement tool, or an industrial plant. Proxy pads enable the microworlds to assimilate external objects, especially objects in the users' real world.

The IntelligentPad will surely work as a meta tool for the development of what Ferguson called "exploratory environments" (Ferguson, 1992).

The IntelligentPad architecture is now implemented in several languages. The Smalltalk 80 version provides more than 400 primitive pads. The SmalltalkAgents version allows concurrent operations of pads. It has made the Macintosh windows and the Toolbox resources work as pads. These two versions are already available. A new version developed in Interviews and C++ will also become available this June.

References

Clark, T. N. (1969). *Gabriel Tarde on Communication & Social Influence*, University of Chicago Press.

Dawkins, R. (1976). *The Selfish Gene*, Oxford Univ. Press.

Ferguson, D. L. (1992). Computers in Teaching and Learning: An Interpretation of Current Practices and Suggestions for Future Directions, In *New Directions in Educational Technology*, (ed. by E. Scanlon and T. O'Shea), Nato ASI Series F-96, Springer-Verlag, 33-50.

McLuhan, M. (1964). *Understanding Media, The Extensions of Man*, McGraw-Hill, New York.

Papert, S. (1980). *Mindstorms : Children, Computers and Powerful Ideas*, Basic Books.

Stefik, M. (1986). The Next Knowledge Medium, *The AI Magazine*, 7 (1), 34-46.

Tanaka, Y. (1989). A Toolkit System for the Synthesis and the Management of Active Media Objects,In *Proc. of the 1st Int'l Conf. on Deductive and Object-Oriented Databases*, Kyoto, Dec., 76-94.

Tanaka, Y. (1991). A Synthetic Dynamic-Media System, In *Proc. of Int'l Conf. on Multimedia Information Systems*, Singapore, 299-310.

Tanaka, Y., and Imataki, T. (1989). IntelligentPad: A Hypermedia System allowing Functional Composition of Active Media Objects through Direct Manipulations, *IFIP '89*, San Francisco, Aug., 541-546.

Tanaka, Y., Nagasaki, A., Akaishi,M., and Noguchi, T. (1992). A Synthetic Media Architecture for an Object-Oriented Open Platform, *Personal Computers and Intelligent Systems, IFIP Congress 92*, Vol III, (ed. F. H. Vogt), North Holland, 104-110.