ELECTRONIC PUBLISHING, VOL . 3(4), 177–178 (NOVEMBER 1990)

Editorial

This issue of EP-odd is the second of two on hypertext and hypermedia topics. The field addressed in these papers has been in existence for more than 25 years, but the major activity (as measured by popular interest, the number of researchers, and successful commercial ventures) has been within the past decade. In the early years, hypertext was fairly easy to define because there were only two implemented computer systems to consider, namely Engelbart's NLS/Augment and Brown University's HES/FRESS. Both systems shared similar characteristics, incorporating a realization of hypertext that can be summarized as consisting of a graph and a separate "browser" that processed the graph. Consequently, it is the browser that defines a reader's capabilities on the graph. In this view, which we will refer to as the *graph model*, there is a clear distinction between the graph (which we can consider to be a hypertext document) and the browser program.

In the second generation of systems, hypertext evolved in several ways as more researchers began investigating the nature of, and applications for, linked information structures. Systems that contributed to this expansion of the domain include Xanadu, Neptune/Ham, NoteCards, Concordia/Document Examiner, Intermedia, gIBIS, KMS, HyperTIES, and Guide. By the mid-eighties, an evolution of some systems away from the graph model was emphasised significantly by Apple's HyperCard, which illustrates (with a vengeance) what we will call the *program model*. In this way of thinking, a hypertext is not a processed graph, but instead is a self-contained program that directly encodes the behavior a reader should experience when reading the hypertext. The browser program of a graph-model system is replaced in the program model by something closer to a run-time interpreter for a programming language. A document in the program model can best be thought of as a program in a language that has specialized support for graphical input and output.

Since hypertext systems were designed in large part by computer scientists, it is not a surprise that the program model's specification languages resemble traditional general-purpose programming languages in form and in power. We feel that an important trend in the development of the program model, a development that parallels the development of programming languages, is a turning away from large, "kitchen-sink" languages, towards smaller languages based on simpler, more focused, forms of specification. Such deliberate restriction of the language's generality can reduce the complexity of specification and can permit inclusion of additional capabilities in the hypertext system. For example, the use of formal automata (for example, state machines and Petri nets) can allow analyses and transformations that cannot readily be performed on the HyperCard-style documents (which are also, in a formal sense, automata, but are in the inherently complex class of Turing machines).

Another evolutionary trend to come out of work on the graph model has been the development of *augmented graph models*, such as semantic nets and other knowledge representations. The concept of a static document and its companion browser program is retained, but more of the semantics of the browser are abstracted out and placed into some formal amendment of the directed graph that is the document. Again, we see a

0894–3982/90/040177–02$05.00 *Received 18 February 1991*

move away from all capabilities being contained in the browser code and towards formal abstraction of important capabilities within the document representation. Formal abstractions, whether automata or augmented graphs, or some as-yet-unidentified approach, make system-independent documents a real near-term possibility instead of a desirable but far-off goal. This also parallels the developmental history of programming languages, in that abstraction features have become standard fare (e.g., OOP) instead of something programmers were free to implement in each program if they chose to do so.

There is a tremendous amount of commercial interest in current hypertext systems, resulting in a tendency to compare hypertext systems from a "feature-based" perspective. To our mind, the power, generality, expressiveness, and simplicity of the *abstractions* and *models* also are important in characterization of a hypertext system. Paying attention to the abstractions and models helps to identify not only the system's current strengths and weaknesses but also the ease and consistency with which the system (and documents created for the system) will be able to be adapted to future changes in the environment in which the system operates.

In closing we would like to acknowledge the many referees that helped us select these six papers. The papers presented in these two issues went through the normal EP-odd refereeing process, but because of the number of submissions and because of scheduling deadlines, we had to ask the referees to complete their reviews in an especially short amount of time. We would especially like to thank the referees for their willingness to help out under these additional time constraints. We also wish to thank David Brailsford for handling the refereeing of our own paper that appears in this issue.

P. DAVID STOTTS AND RICHARD FURUTA