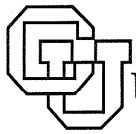
Supporting Personalizable Learning

Gerry Stahl

CU-CS-788-95



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Supporting Personalizable Learning

by Gerry Stahl

CU-CS-788-95
October 1995

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430
(303) 492-7514

For further information or comments, contact the author:

Gerry Stahl
3900 Pebble Beach Drive
Niwot, CO 80503
(303) 444-2792
gerry@cs.colorado.edu

SUPPORTING PERSONALIZABLE LEARNING

BY GERRY STAHL

ABSTRACT

This paper outlines a research agenda for exploring computer-based approaches to rendering educational resources personalizable. Using proposed technologies, learners and their teachers can select exploratory activities as well as curriculum to support or guide these learning activities from digital libraries on the Internet, and adapt the content and display of these materials to personal interests and local needs.

The paper begins by suggesting an initial testbed for personalizable learning software building upon the Agentsheets Remote Exploratorium (ARE) that is currently under construction at the University of Colorado. It then touches upon diverse pedagogical theories to underscore the importance of personalization to learning. Next, it presents a vision of a more comprehensive system of software for personalizing resources from global digital libraries, highlighting the general issues involved. This vision is grounded in two innovative software systems: a Teacher's Curriculum Assistant (TCA) and a Personalizable Learning Medium (PLM). Generalizing from these prototypes, it considers several issues for a theory of personalizable software.

PREFACE: IN THE IDEAL WORLD

In the ideal world of the future you would receive this document in the form of personalizable hypertext. This Preface might query you:

```
[ ]    Which aspects of the following discussion are of most
interest to you?
[ ]    What background do you already have in these matters?
[ ]    How much time do you want to spend going into details?
```

Of course, in the really ideal future, you would already have tailored your computational reading environment to your general preferences and you would just have to tune that embedded knowledge to your interests in this specific material. Then, rather than being a fixed presentation of text, this document would be tailored to your personal interests and it would allow you to explore its ideas in an open-ended format. The tailoring would be automated, using computational hypertext that restructures itself dynamically. You could delegate the personalizing and also make certain decisions yourself on what is presented. You could follow linkages among ideas at your discretion and expand materials to whatever levels of detail you desired.

We are not yet in that long anticipated future. As a default, the author has had to assume that you are too busy to study the details of the following argument and has attempted to present the main points as concisely as possible. Pointers to further motivational discussion and implementation technicalities are given at the end, since paper documents do not allow the active linkages of hypertext.

SECTION I. PERSONALIZING AGENTSHEETS AND THE REMOTE EXPLORATORIUM

Research at CU on human-computer communication and support of life-long learning has long recognized the need to make complex, poorly structured information spaces more personal (e.g., Fischer & Nieper, 1987; Fischer & Stevens, 1991). In the following pages, I propose a series of software systems to explore technologies for *personalizable software* to support *personalizable learning*. This research agenda largely applies approaches and functionality developed at CU—including in my dissertation (Stahl, 1993b)—to the needs of learners and their teachers.

During the past two years, I have designed two systems to support personalizable learning: a Teacher's Curriculum Assistant (TCA) and a Personalizable Learning Medium (PLM). The implementation of TCA and PLM would be a substantial undertaking involving several person-years of design, programming and testing. It involves creating a digital library of educational resources and curriculum, all structured in the correct hypertext format and properly indexed. It requires tools for authors to construct personalizable documents, for teachers to customize lesson plans and for learners to explore resources. The participation of teachers and learners is needed to ensure that the software is designed to fulfill real needs and to meet practical usage requirements.

An incremental approach to implementing the proposed approach to personalizable software is needed. The Agentsheets Remote Exploratorium (ARE) project at CU provides a potential testbed for doing this. ARE is a digital library of Agentsheets simulation titles (Ambach, Perrone & Repenning, 1995). Currently, the ARE library is centralized on a single Web server; there is little supporting information to guide or support the selection and use of the titles; only one version of each title is available; users cannot annotate or otherwise supplement the available information.

There is growing interest among students, teachers and researchers to see the ARE library grow and decentralize. ARE is now at a formative point in its development. The published library is still at a manageable size and its administration is still centralized so that new formats and standards can be introduced without causing problems for an installed base. This is a good time to explore how techniques of personalizable software could enhance the usability of ARE.

An incremental approach to developing personalizable software within the ARE project could proceed in several phases, such as:

1. A personalizable User's Guide to Agentsheets and the Remote Exploratorium.
2. Personalizable hypertext curriculum materials to suggest educational usages of Agentsheets simulations and to provide relevant background materials.
3. A personalizable end-user language for manipulating the computational hypertext.
4. Personalizable versions of Agentsheets titles, with their associated curriculum and commentaries.
5. Personalizable tools for locating Agentsheets titles globally.

The following discussion of these phases is meant to be merely suggestive. The details would have to be worked out with the ARE developers and with typical potential users such as students and teachers.

PHASE 1: A PERSONAL USER'S GUIDE

Perhaps the most urgent need to support the wider use of Agentsheets is an adequate User's Guide. The current manual is too sketchy to meet this need. Without a more complete manual, the anticipated spread of Agentsheets use by researchers building new functionality, by university students developing sophisticated new titles and by public school teachers and students exploring titles in the classroom will continue to place a prohibitive burden on core ARE staff.

An Agentsheets User's Guide should be part of the Agentsheets environment, so that it can be incorporated into context-sensitive help to explain the features of Agentsheets within Agentsheets. This means adding a hypertext facility to Agentsheets. Giving people the ability to author commentary in the User's Guide, will allow developers at every level of Agentsheets (the substrate, the titles and the instances) to document their work in a centralized repository.

A hypertext extension to Agentsheets should include support for personal perspectives and a hypertext navigation language, even if these mechanisms are not fully implemented during this phase. With perspectives functionality, User's Guide contents can be entered within different perspectives, such as `novice user`, `student`, `teacher`, `title developer` and `substrate builder`. Then readers can select the appropriate perspective to display information about using Agentsheets at their personal level. Individual users and groups of developers can define their own perspectives and save annotations that are only relevant to them and that will not be displayed in other perspectives.

The addition of hypertext to Agentsheets will undoubtedly have uses that go far beyond the User's Guide and documentation and that cannot yet be foreseen. The power of computational hypertext will have strong synergies with the computational agency of the simulation environment. The User's Guide simply provides a practical artifact to experiment with in implementing the hypertext.

PHASE 2: PERSONAL CURRICULUM

The next phase is to use the hypertext to develop personalizable educational curriculum to accompany the Agentsheets titles. This can include lesson plans such as those in TCA as well as textual (or multimedia) information for students exploring titles such as that in PLM. These curricular materials will be stored on the Web as part of ARE. All the techniques of PLM can be developed and tested in this context. Also, people using this personal curriculum can annotate it and modify it, creating their own new versions of documents.

PHASE 3: A PERSONAL LANGUAGE IN AGENTALK

Computational hypertext is driven by a navigational language. Queries are expressed in this language; among other things, these queries retrieve sets of linked nodes and display their proper versions for the user. This navigational language can be implemented in the new approach to AgenTalk currently under development.

AgenTalk is the end-user programming language already used to define the behavior of agents in Agentsheets. The new approach provides a drop-and-drag visual programming interface that guides the user in constructing syntactically and semantically valid expressions. This interface would be very helpful and appropriate to the hypertext language.

The kernel of the navigational language is required to implement computational hypertext in Phase 1. However, once in place, it can be extended to serve as an end-user language for various functions, such as defining queries for searching the Web and defining critics for analyzing artifacts, including Agentsheets titles and curriculum lesson plans. Because the vocabulary of the language can be customized with personal extensions, people can define their personal queries, critics and displays by using the language.

PHASE 4: EVERYTHING IS PERSONAL

Ultimately, all aspects of the ARE can be made personalizable. Most of Agentsheets is already modularized or could be made so. This not only means that people can share pieces of their titles (like an agent behavior or appearance), but that titles can have multiple versions at any level. The `Segregation` simulation, for instance, could be posted on an ARE Web server with multiple versions of its parts embedded in it. Then a user (e.g., a teacher or a student) could select which of the available behavior versions a given agent should have. This choice could be made on an *ad hoc* basis in the interface for defining behaviors. However, it could also be made by choosing a perspective; the choice of perspective would make choices of versions throughout the title all at once, automatically and consistently. Associated hypertextual descriptions could guide the user in selecting which perspective to choose. Of course, once a narrow perspective was chosen, all commentary and User's Guide information would be specific (personalized) to the versions associated with that perspective unless one explicitly requested information from a broader perspective.

PHASE 5: PERSONAL RETRIEVAL

The ARE project is primarily concerned with making Agentsheets titles available over the Web. Enhancing the project with techniques of personalizable software would result in a globally distributed but locally personalizable web of Agentsheets versions, curriculum and commentary. That is, versions of title components, documentation and annotations would be distributed across multiple Web servers. The distributed nodes and links of the hypertext system would be located, retrieved, assembled and displayed by the computational hypertext system. In order to maintain acceptable levels of efficiency, all the relevant components would be downloaded to the user's computer and stored locally for a certain period of time. The algorithms for doing this would need to be worked out. The computational hypertext discussed here goes considerably beyond the HTML hypertext of the Web; it provides far more capabilities.

Computational hypermedia has been prototyped in a single-user system; the challenge is to adapt it to work across the Internet, extending the power an order of magnitude beyond HTML.

It might be helpful to use some of the techniques from TCA to support the efficient finding of relevant titles and associated resources on the Web. These involve the indexing of all titles and documents at all sites and storing this meta-info at one or more central sites. Alternatively, it may be possible to automate some of this with Web search engines and other Internet daemons. It may also be useful to download the meta-info to local sites periodically to facilitate filtering and browsing of indexes in personally structured local search spaces.

In addition to posting Agentsheets titles, documentation and curriculum, the expanded system would allow users to post their new vocabulary for the hypertext language—their personal languages—including definitions of search queries and computational critics. They could also post the names of newly structured perspectives, along with documentation on the advantages of using those perspectives. The same language terms, queries and perspectives that are used for personalizing individual documents could then be used in searching the Web. In effect, the global ARE web would become a giant *personalizable* document.

FULFILLING THE PROMISE OF THE WEB

The World Wide Web holds out the promise of providing a decentralized, public medium to meet people's information needs and interests. The proposed research agenda would create a web of information related to Agentsheets titles that could not only serve as a model for how to make information shareable and manageable, but would also show how to make the information personalizable. The Agentsheets Remote Exploratorium could provide an effective testbed in which to develop techniques for managing decentralized evolution of digital libraries with personalizable software.

SECTION II. PERSONALIZABLE LEARNING

TAYLORIZING THE STUDENT *VERSUS* TAILORING BY THE LEARNER

Learning in the future will not consist primarily of training based on Taylorized knowledge. With the term *Taylorizing* I refer to the industrial practice of rationalizing human activities popularized by Frederick Taylor in the early 1900's. This corresponds to the behaviorist movement in psychology, to instructionism in education and, more generally, to rationalism in philosophy. Following this worldview, one analyzes activities into elemental constituents of required skills, physical movements and intellectual efforts. Then one optimizes the process by removing unnecessary steps and often by separating the intellectual supervisory tasks (management, teachers) from the repetitive motions (workers, students). This approach drove the industrial revolution and the public education that schooled its work force.

The term *tailorable* (or personalizable) *learning* refers to an alternative approach needed for the info age. Rationalization provided an historically necessary service by making explicit the elements of work and learning that had traditionally been blended in amorphous, tacit, organic ways. But work and learning in today's world require reorganization and reintegration of those elements under the control of the

individual worker or learner. There is too much innovation and info-overload now to rely on standard operating procedures. Many contemporary work and learning practices cannot be codified; to work in these domains is to negotiate new definitions of the domains with one's colleagues.

Since the beginnings of formal education, theorists have recognized the need to adapt teaching to the *personal situation* of the learner. Ironically, the rationalization process that led to standardization now makes personalizing feasible. By breaking education into instructional elements, it provides the raw materials for the computers of learners to re-synthesize these materials in ways suited to individual needs (e.g., to personal backgrounds or to an immediate task at hand). The computer provides a tool to assist in organizing enormous numbers of elements according to flexibly specified constraints. For instance, if this paper were analyzed into elemental ideas (say, roughly its paragraphs) and their interrelationships (their various types of linkages) in the form of computational hypertext, then the right software could allow you, the reader, to tailor the presentation of that material to your personal desires.

The promise of personal computers will finally be achieved when software makes information personalizable.

Personal computers have always dangled a tempting promise in front of us: to grant us personal control over information. Too many systems make us adapt to the computer instead. The power of today's computers, the sophistication of available software techniques and the medium of the Internet combine to make feasible the fulfillment of that promise.

SUPPORTING AUTHENTIC EXPLORATION WITH PERSONALIZED CONTENT

This paper proposes a research agenda to give learners—individually or in collaborations—control over the information they need for their *own* learning practices. Section II motivates the need for personalizing educational materials: both authentic projects for *constructivist* exploration and *instructional* curricular materials to support learning-on-demand using such projects. The two competing pedagogical approaches are here synthesized by making them both *personalizable* by the learner.

Pedagogical theories have long argued for a level of personalizing that has not been practical within traditional schooling contexts. New technologies promise to facilitate the sharing and personalizing of both constructivist activities and instructionist curriculum to support those activities. The following review of learning theories motivates the need for making educational resources personalizable by learners and their teachers.

PLATO'S CONCEPT OF EDUCATION

Plato presented his view of learning in the *Meno* dialog 2500 years ago. For him, education is a process of drawing knowledge out of the learner. This is accomplished by dialog with the learner, guiding the learner to construct an understanding of the idea being discussed, such as a theorem of geometry. The dialog format is a mechanism for situating teaching within the understanding of the learner and for basing the teaching on the learner's previous understanding. Unfortunately, this

personalized approach to education was overshadowed by the idealistic strains that became dominant in the later Platonist heritage. The subsequent Western tradition—founded on Plato's vision of eternal ideas and culminating in rationalism—gives us little insight for understanding *evolving* knowledge. Plato himself could not account for new knowledge, hence his definition of education as a remembering of something once known but long since forgotten.

ROUSSEAU'S SUBTLE ROLE FOR THE TEACHER OR SELF-LEARNER

In *The Education of Emile*, Rousseau, too, stresses the need to tune educational presentations to the personal interests and abilities of the learner. Rousseau thinks that new material can be learned if one properly prepares, motivates and situates the learner. The ideal is to lead the learner to construct his knowledge stage by stage, advancing over time to where he is prepared for learning more and more. In his labor-extensive economy Rousseau recommends a private tutor who can adapt educational experiences to a learner's personal needs.

VYGOTSKY'S ZONE OF PROXIMAL DEVELOPMENT

Vygotsky (1934) clarifies the notion that learners can be ready to learn something they do not yet understand by defining that readiness as a zone ripe for development. His developmental psychology differs from Piaget's (1929), deriving from social communicative functions rather than primarily from an individualistic logic of development. For both Piaget and Vygotsky, formal education can only succeed when the student is developmentally prepared; for Vygotsky this is a function of the student's social context or community. That is, the ability to learn is dependent upon the social relations and situations in which the learner is active.

LAVE'S COMMUNITY OF PRACTICE

Lave and Wenger (1991) present a theory of situated learning that focuses on an individual's gradually increasing participation in communities of practice. A learner moves from the periphery of a community inwards by learning the knowledge that defines that community. The acceptance of the newcomer into the community is a process of negotiation through which the knowledge base of the community evolves. This view of education applies not only to non-literate apprentice traditions like the midwives of Peru, but to socialization processes in public schools and within professions. Negotiations of knowledge are political processes in which traditions, factions and individuals vie for their rival interpretations of values and definitions from their varying perspectives, continually modifying the definition of domain knowledge in the process. In the theory of situated learning, learning takes place through social activity (practice or *praxis*). That activity constructs new domain knowledge—not just in the sense of constructing personally meaningful representations of the domain in the mind of the learner, but in the sense of the community reinterpreting its own definition of the socially constructed domain. Thus, the activity of learning transforms reality, truth and knowledge (Stahl, 1975).

The idea that communities of practice have fixed bodies of knowledge that can be identified and codified as domain rules provided a useful fiction for early attempts at knowledge-based computer support. However, the limits of this fiction were soon

reached. The first problem is that domain knowledge is overwhelmingly tacit; it is learned through gradual participation in a community. Knowledge acquisition attempts via interviewing of experts confront a multitude of problems: there are no "experts" who know the whole field; it is hard for experts to formulate much of their wisdom outside of situated practice (e.g., when interviewed); terms and rules depend upon further tacit skills and knowledge for their meaning and applicability; different practitioners have wildly different perspectives on the same field; domains evolve over time; creative work reinvents the field continuously (Stahl, 1993b).

LEARNING AS INTERPRETATION

The philosophy of interpretation—based on Heidegger's (1927) thought—explains how a prepared mind learns new domain knowledge, finally answering Plato's quandary of how one can learn what one does not already know. Interpretation is a process of making certain aspects of tacit previous understanding explicit in order to conceptualize and transform (reinterpret) the knowledge. When faced with a phenomenon that cannot be readily understood, one makes explicit one's relevant tacit understandings (from one's zone of proximal development) until one can extend previous knowledge sufficiently to embrace (*interpret*, comprehend) the new phenomenon. Afterwards, the new knowledge can revert to a tacit state for use in future situated practice. According to this philosophy, interpretation is situated, linguistic and perspectival. That is, it is based on previous understanding, current concerns and future goals; relies heavily on domain conceptualizations; and necessarily adopts a personal standpoint. Accordingly, computer support for interpretive processes should provide situational context, offer linguistic tools and be personalizable (Stahl, 1993a).

Hermeneutics (the philosophy of interpretation) provides a framework for relating and synthesizing competing theories of learning. According to hermeneutics, learning is an interpretive process and is therefore situated, linguistic and perspectival. *Situated* refers to the tacit dimension of background skills and knowledge and to the social context in which learning takes place. Constructivist attempts to create authentic projects try to create situations in which the learner has a personal concern for the activity and can bring a background of situated understanding to the learning experience. On the other hand, *linguistic* refers to the process of making knowledge explicit through conceptualization in language. This corresponds to the instructionist approach of providing abstract information about the topic being learned. Hermeneutics recognizes the value in both these approaches and assigns them to stages in the processes of interpretation, that take place within personal *perspectives*. Having a perspective means that the learner understands from within a context of personal concerns, background and goals that ground new knowledge in understood meanings (Stahl, 1992b).

EXTENDING COGNITION TO MEET THE CHALLENGES OF THE FUTURE

The challenges facing professionals today exceed the interpretive ability of unaided individuals. Often, the primary new skills needed are symbolic, representational, terminological. Extended memories (such as electronic media) are needed to keep track of overwhelming volumes of information, external representations are needed to structure it and computational tools are needed to process it. At the same time,

such work cannot be simply automated; the tacit knowledge and the interpretive skills of experienced people are still absolutely central. As new challenges arise, we must extend our skills. Conversely, new skills allow us to project newer challenges that suddenly seem feasible within the proximal zone of our abilities. Through this dialectic, domains of practice evolve—for individuals and communities. The computer-based tools that are increasingly called for in this spiraling escalation of skill requirements must capture domain distinctions and innovations as they are created, must allow for the construction of new tools and must support creative personal perspectives on the domain. These are fundamental requirements for software to support learning throughout life (Stahl, 1992a).

To prepare people for the challenges of the future requires new pedagogical approaches and new supports. The best way to learn new practices is often to practice scaled-down versions of them, to join in a community of practice gradually from the periphery. Since skilled workers carry out collaborative projects involving analysis of data using multiple tools and representations, students should engage in similar but simpler open-ended projects. The skills that need to be learned are not well-defined, atomic facts but the ability to define problems, to evaluate approaches, to communicate issues, to use computational tools, to apply multiple representations, to delegate tasks, to negotiate team efforts, to plan and to report (NCTM, 1989). These skills are often best learned tacitly, through participation in projects. One must be prepared to learn (i.e., have a zone of proximal development already established) by having been involved in similar, if simpler efforts in the past. One must also be motivated and engaged in the new activities by being situated in personally authentic activities.

NEED FOR CURRICULAR CONTEXTS OF PROJECTS

The project-centered learning just described requires curriculum in two senses: guidance and content. Learners need to be guided through individual projects and from project to project to ensure that they are ready for the activities and can get the most out of them. They also need ready access to related information (Stahl, Sumner & Owen, 1995). Guidance might take the form of a teacher like Rousseau or it might be accomplished by social structures that guide newcomers into communities of practice. Often, skilled learners with access to stimulating ideas and engaging info sources can guide themselves by paying attention to their personal interests and abilities.

Written curriculum can suggest sequences of projects and can provide supplementary content to make the projects more effective learning experiences. Such content can include technical information needed to complete a project, explicit rules useful in the specific activities or general background information that makes the project more interesting and meaningful. Curriculum need not mean a detailed blueprint with all learning defined by explicit facts to be memorized and tested. Rather, it can provide material for learning-on-demand, giving flexible access to information in response to a learner's situated needs. The "situation" in this case includes not only the task at hand, the physical work environment and the social community of practice, but also the learner's personal background knowledge and skills. For this reason, information delivery should not only be relevant to the task, timely and culturally sensitive, but it

should be presented in a personally meaningful and effective format for the learner (Stahl, Sumner & Repenning, 1995).

THE PROBLEM AND THE PROMISE

Education is in crisis. The challenges it is called upon to meet by our info society are daunting. Many bold actions are needed to reform education adequately. Once teachers, parents, students and administrators accept the constructionist approach in principle, the question of what is to replace the old textbooks, drill sheets and lesson plans poses an overwhelming practical obstacle. Here is where computer access to digital libraries on the Internet can help: by facilitating the sharing of well thought through educational activities and resources.

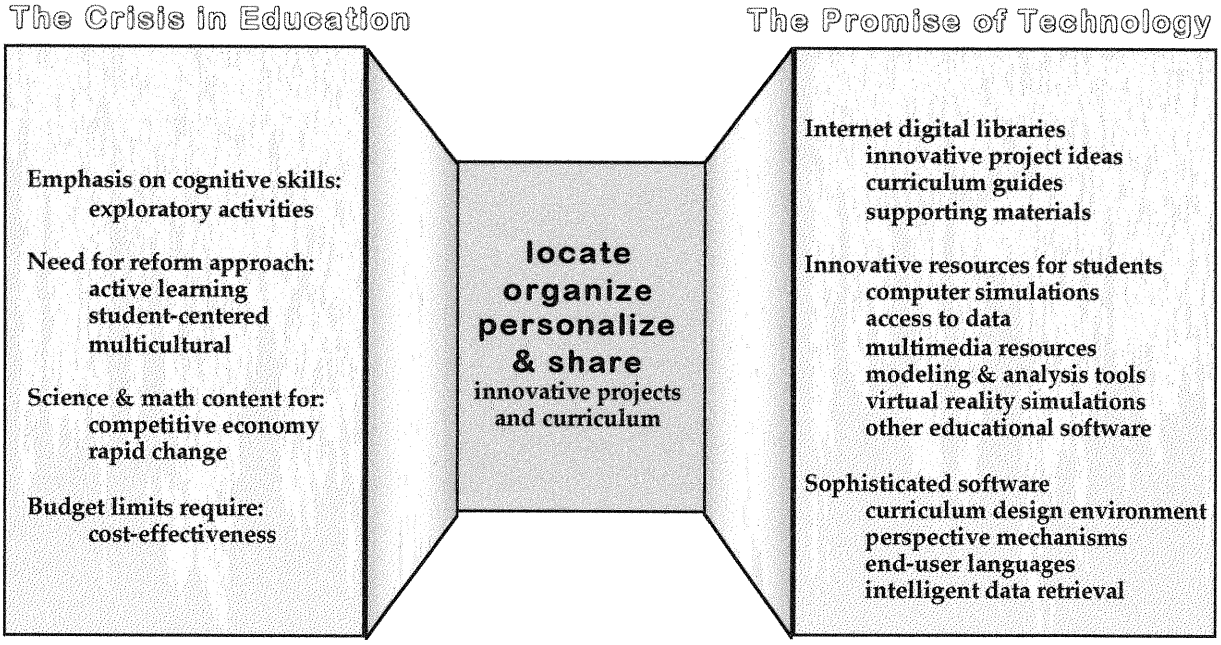


Figure 1. Software can bridge the gap between the crisis of education and the promise of technology by helping learners and their teachers to access project ideas and curriculum and to adapt them to personal needs.

The preceding paragraphs were intended to sketch a context for proposing how computers can support learning by making shared resources personal. Until now, educational software has generally pursued approaches (e.g., "drill and kill" or "edu-tainment") that missed the real potential of educational software: to present project situations and supporting information in formats that learners and their teachers can make personal. Sections III and IV will try to outline what such an approach might entail.

SECTION III. TEACHER'S CURRICULUM ASSISTANT

Two sets of issues have already emerged through the effort to establish the Agentsheets Remote Exploratorium (ARE) on the World Wide Web (Stahl, Sumner & Repenning, 1995):

- *Curriculum support*: Simulations by themselves are not enough to ensure that learning will take place. There needs to be accompanying curriculum that (a) suggests ways for teachers to guide their students in integrating the creation and exploration of the simulations within broader learning contexts and (b) provides insightful and supportive background and related information.

- *Personalized access*: The possibility of Agentsheets users posting their creations on the Web via the ARE points to the need for software utilities to help other potential users to locate ARE sites, search through them for Agentsheets titles of interest, download and run selected titles, adapt the titles to their personal needs and share their creations or adaptations with other people through the Web.

The issues ARE raises are general problems for the use of educational digital libraries. An independent attempt to think through what kind of software is needed to address these issues resulted in two prototypes: a Teacher's Curriculum Assistant (TCA) to help teachers develop curriculum (Stahl, 1995b) and a Personal Learning Medium (PLM) to present textual materials to students (Stahl, 1995a). These two systems will be described in Sections III and IV. Then Section V will draw upon these examples to consider some general issues concerning software support for personalizable learning.

NEED FOR COMPUTER SUPPORT OF CURRICULUM DEVELOPMENT

Educational researchers are calling for constructivist reforms that require significant changes in curriculum (Greeno, 1993). Printed textbooks cannot keep pace with the necessary changes in approach, format or content required by these pedagogical approaches. Nor are textbooks readily adaptable to local conditions and individual learning styles. If schools are going to offer students opportunities to construct their own knowledge actively and interactively, then new educational materials are needed that foster exploration and that are tailored to local students and their situations.

Isolated ideas for classroom activities and individual resources like computer simulations, programming environments, CD-ROMs, or video disks do not by themselves foster conceptual development. They need curricular contexts and related materials that supply motivation, background, goals and opportunities for reflection. They need to fit into developmentally appropriate, balanced structures that allow time on task/time off task, exploration/reflection, individual thought/group discussion, absorption/presentation. Classroom teachers have neither the time nor the resources to design this kind of subtle curriculum from scratch on their own.

Electronic repositories of curriculum are necessary to speed the pace of educational reform and to lower costs of innovative curriculum development. They must combine the latest educational resources with carefully crafted curriculum. Attempts to date to fill this need with scattered Internet postings or World Wide Web pages of fixed resources have proven to be too limited. They do not provide a complete solution to the needs of teachers who want to reform their classroom teaching. They lack supporting materials, suggested lesson plans, or variations for different circumstances. Standard Internet browsers do not adequately support teacher needs: locating relevant resources, searching among them, selecting the best fit, adapting

resources and curriculum to local needs, organizing curriculum into effective learning contexts and sharing results or experiences.

There is a need to: (i) reuse and adapt model curriculum and resources to local classroom situations, (ii) disseminate innovative classroom ideas and experiences globally, (iii) establish digital educational repositories to promote sharing of effective reform curriculum.

THE TCA APPROACH

The Teacher's Curriculum Assistant responds to the needs of teachers interested in implementing educational reform in their classrooms by exploring how best to make available to them model curriculum disseminated via the technology of digital libraries. It provides software tools to teachers for personalizing the curriculum to their students, their teaching styles and their local conditions.

TCA is an Internet-based curriculum development environment. It provides teachers with facilities to make effective use of educational resources available on the Internet. It allows providers of resources—such as publishers of textbooks, educational software, or CD-ROMs—to index and publish their offerings where teachers can locate them. It manages the digital repositories so they can grow in an orderly way.

Acceptance of TCA by teachers, providers and repository managers entails agreement upon a set of standards for descriptive indexing of resources and for structuring of curriculum. For instance, a preliminary TCA prototype uses approximately 30 indexes for resources; it structures curriculum in a hierarchy of semester themes, weekly units, and daily lesson plans composed of resources; and it adopts certain repository management policies. These standards are essential for providing computer support to teachers. With such standards, many techniques of artificial intelligence and information retrieval can be applied to the tasks of locating, searching, selecting, adapting, organizing and sharing resources and curricula. Without them, teachers will remain lost in the Web's immense hyperspace.

In TCA the curriculum repository works as follows (figure 2): Curriculum providers post educational resources to their own servers on the Internet. They publish the addresses of these resources on a central TCA server. Along with the addresses, they also publish descriptions (indexes) of the resources and suggested curriculum for using the resources, adhering to TCA standards. Providers include educational software publishers, textbook publishers, educational research centers, and other organizations interested in the development and dissemination of reform curriculum. Teachers have TCA client software on their desktop computers. This software maintains a database of resource indexes and curriculum that is periodically updated from the TCA Internet server via the school district. Teachers do their curriculum planning with the information on their own computers and then download resources they need from the Internet addresses stored there. This allows the latest educational resources to flow into the teacher's computer, where they are organized into meaningful curriculum to structure classroom activities.

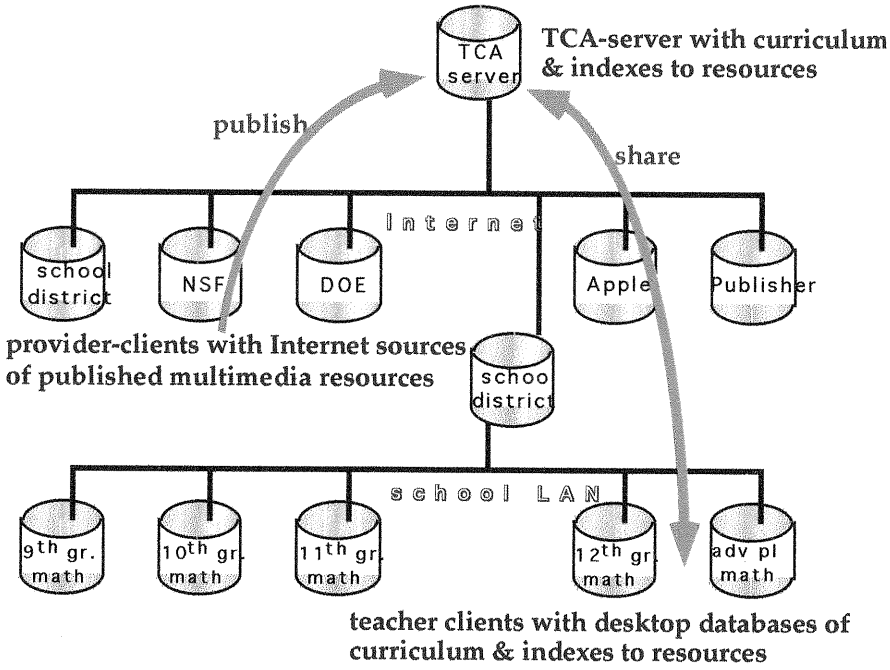


Figure 2. TCA network architecture. Large resources are stored on Internet servers, while summary information about these resources is copied to the desktop computers of teachers.

A TCA PROTOTYPE

TCA includes both client and server software for accessing and maintaining educational digital libraries on the Internet: (i) **teacher-client software** for teachers to make use of model curriculum and multimedia resources, (ii) **provider-client software** for organizations to publish educational resources on the Internet, and (iii) **TCA-server software** to manage the digital repositories of curriculum.

Teacher-client software

The TCA approach begins with teachers and curriculum developers. It works with them to understand teachers' curriculum needs and the traditional curriculum usage process. It explores with teachers ways in which personal computers with access to educational resources and model curriculum can be used to meet their needs and to help them obtain and adopt useful curriculum in their classrooms.

When teachers try to use browsers like Netscape or Mosaic to take advantage of the educational ideas that are beginning to be posted to the Web, they meet with the following problems:

- there are no effective methods for **locating** relevant curriculum sites,
- it is too hard to **search** for items of interest,
- these is no choice of versions to **select** for different situations,
- there are no tools for **adapting** what is found to local needs,
- there is no support for **organizing** scattered ideas into workable curriculum,
- there are no ways for teachers to **share** their experiences.

These problems can be overcome with centralized repositories of carefully structured curriculum and indexed resources. The repositories should support two-way

communication, so that teachers can share their experiences using materials in the repositories and can "grow" the repositories.

Based on preliminary study of these issues, a TCA prototype has been developed. Six interface screens have been designed for teacher support: **Profiler, Explorer, Versions, Editor, Planner, Networker**.

The Profiler, Explorer, and Versions interfaces work together for information retrieval (figure 3). The Profiler helps teachers define classroom profiles and locates curriculum and resources that match the profile. The Explorer displays these items and allows the teacher to search through them to find related items. Versions then helps the teacher select from alternative versions that have been adapted by other teachers. Through these interfaces, teachers can locate the available materials that most closely match their personal needs; this makes it easier to tailor the materials to individual requirements.

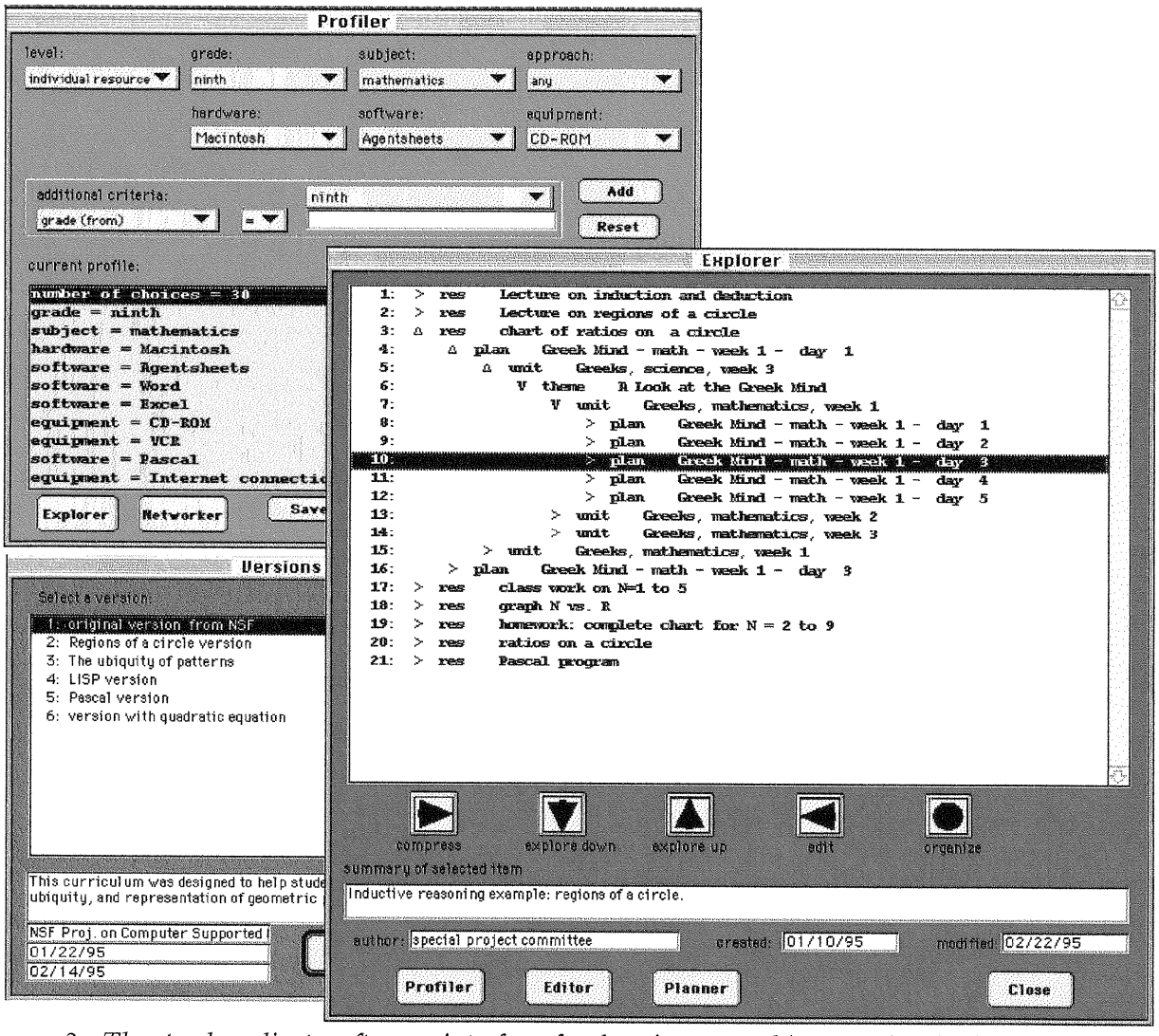


Figure 3. The teacher-client software interface for locating, searching, and selecting resources and curriculum: the Profiler, Explorer, and Versions.

The Planner, Editor and Networker help the teacher to prepare resources and curriculum for use and to share the results of classroom use (figure 4). The Planner is

a design environment for reusing and reorganizing lesson plans. The Editor allows the teacher to modify and adapt resources. This is a primary means for personalizing the curriculum. Finally, the Networker supports interactions with the Internet, providing a two-way medium of communication with a global community of teachers. Using the Networker, a teacher can share personalized versions of standard curriculum with other teachers who might have similar needs.

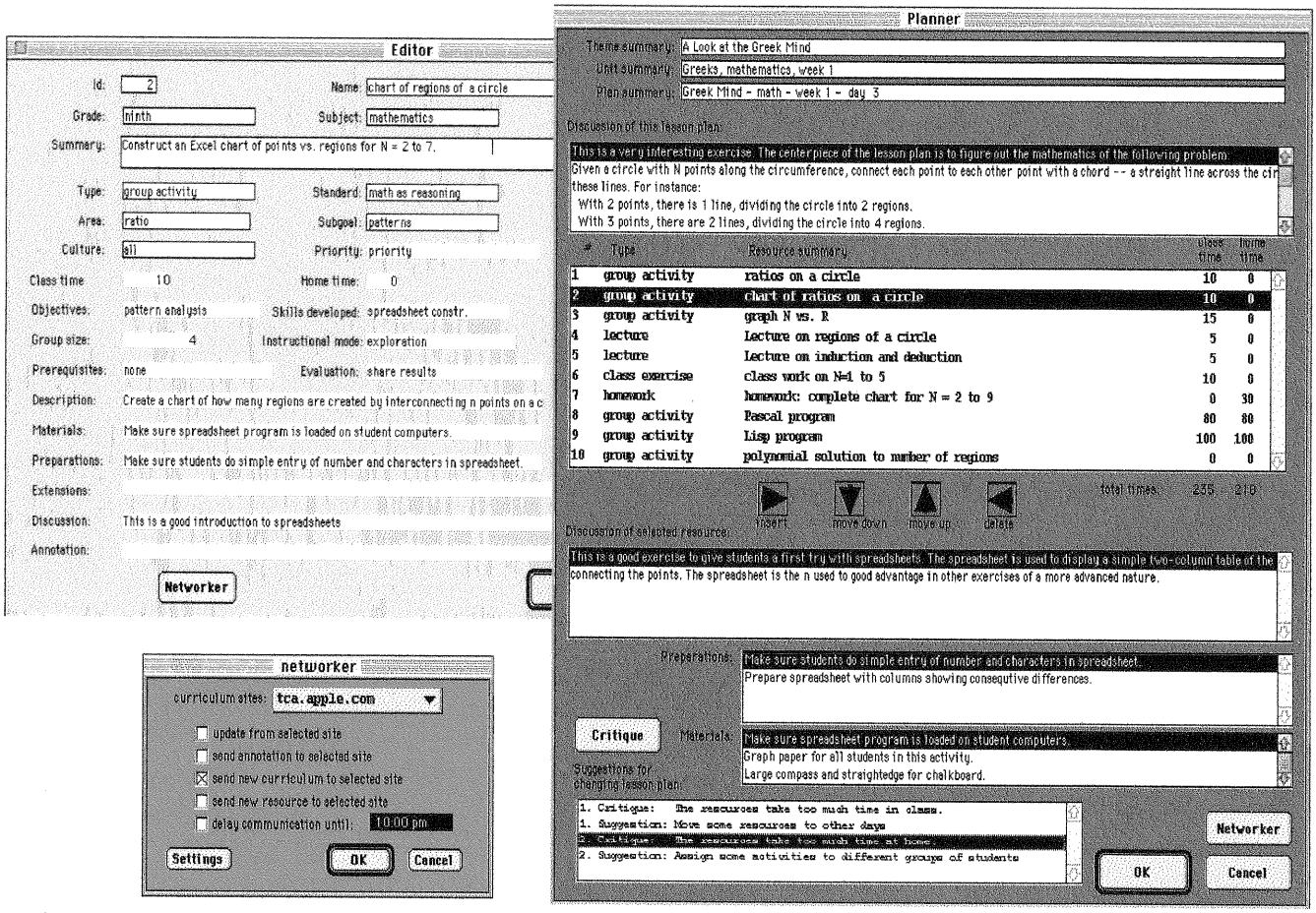


Figure 4. The teacher-client interface for adapting, organizing, and sharing resources and curriculum: the Planner, Editor and Networker.

Provider-client software

For teachers to reuse and build upon model curriculum and innovative resources, they must have easy access to large repositories of materials that have been developed by provider organizations. It requires thoughtful design work, structuring and presentation to develop materials that can be personalized by many teachers with different needs. The TCA software supports this task. Provider-client software gives computer-based support to the providers of curriculum and resources in preparing their materials for the TCA Internet repository.

Digital libraries of model curriculum and related educational resources must be put on the Internet. Both units of curriculum and individual resources must be indexed with descriptors that correspond to teacher interests (e.g., educational standards) and to classroom profiles.

TCA includes software tools to facilitate the construction and inter-relating of curriculum items, the indexing of resources, and the creation of alternative versions of both curriculum and resources. These tools are primarily for providers of curriculum. They can be used by NSF-funded projects submitting content to the repository, by school district staff and by publishers of educational software or textbooks.

Multimedia resources are maintained on servers distributed across the Internet. These are typically servers owned by the provider. The resources can include textual readings, evaluation materials, video clips or software such as simulations. That way, these large files do not take up room on the teachers' computers. This also allows the providers to update the materials whenever necessary.

A central TCA server maintains all the model curriculum associated with the resources as well as the descriptive indexes to the resources. Teachers have a copy of the indexes on their computers and can update their copies from the TCA server at will. The indexes allow teachers to search for interesting items without downloading the large resources. This means that teachers can do their curriculum planning and class preparation on their computer without connecting to the Internet. They only need to connect in order to download selected resources for adaptation and distribution to students. The Internet locations of the resources are included in the indexes. Downloading can be consolidated by a school district for all its teachers and done over night if Internet traffic is heavy during the school day.

Provider software incorporates the TCA index standards: what descriptors are required or optional, what values the descriptors may take and which descriptors may take user-defined values. These indexes are used to structure the curriculum items, the summary information for multimedia resources and the classroom profiles that teachers enter into their copies of TCA. This set of standards is what makes possible extensive software support for curriculum development in TCA.

TCA-server software

Acceptance of the proposed digital repository by a national community of teachers requires the collaboration of a number of organizations as well as the compilation of a critical mass of curriculum content. Management structures have to be agreed upon and standards need to be adopted for the formatting of content.

The organization and quality control of information in digital libraries is critical if they are to be useful and usable. There are two extreme models of how to build an effective library of personalizable materials: (i) provide a few very general materials that have been well thought out and carefully structured for adaptation to diverse needs or (ii) allow practitioners to contribute many versions of materials that have proven successful in classrooms. The need for control over contents must be balanced by the desirability of users being able to comment upon, modify and expand these contents. Digital repositories can be structured variously to adopt different management approaches. One repository could contain model mathematics curriculum developed by NSF-funded projects, be managed by a committee of NSF staff and grantees, and allow no additions by teachers using the curriculum. A second repository could contain mathematics curriculum used by a local school district, be managed by curriculum staff, and allow narrative annotations by teachers discussing

their experiences using resources and curriculum. A third repository could be open for postings by teachers, be self-managed, and allow teachers to add new ideas, innovative resources, adapted versions, and annotations.

SECTION IV. PERSONALIZABLE LEARNING MEDIUM

OVERVIEW OF PLM

While TCA is primarily intended to support teachers, PLM is designed for the individual learner; together, the two systems support personalizing of the whole classroom process of organizing learning activities and engaging in them. Once activities and curriculum are found in a digital library, they must be adapted to the needs and interests of individual learners and presented to them in personally meaningful formats. PLM is a systematic attempt to integrate a number of software techniques into a system to personalize educational materials and to present them in the most effective format for the individual learners. PLM incorporates some of the functionality from TCA within a system designed for learners to use themselves, and then it adds personalizable display functions.

PLM (1) provides access to current educational materials on the Internet, (2) supports hypertext exploration, (3) incorporates multimedia simulations, and (4) provides comprehensive personalizability.

A personalized browser of digital libraries

It is not feasible to manually develop separate versions of curriculum for each type of learner and each set of student interests. However, it *is* technically feasible to store a single corpus of properly structured curriculum on the Internet, keep it up-to-date, and automate its analysis into elemental units that can then be recombined in a large variety of ways to match the needs and desires of different audiences. Furthermore, this process of personalizing can be put under the control of the learner in ways that are not overly intrusive or demanding. PLM takes this approach. It provides a personalized browser of educational resources and related curriculum stored in digital libraries on the Internet. It then displays the material to the learner in a personalized presentation.

Computational hypermedia to customize HTML

Academic research in computer science suggests many promising applications of hypermedia to education, particularly if hypermedia is extended with techniques from artificial intelligence. Promising technologies for this include perspectives mechanisms and task-specific languages. Retrieval of hypermedia materials can use techniques of query reformulation, fuzzy logic and case-based reasoning (Stahl & Owen, 1995) to aid in the difficult task of making relevancy criteria explicit. The user interface to a hypermedia knowledge system can take the form of a design environment to help people construct useful presentations of information.

The technology underlying the PLM system is computational hypermedia. It is a fine-grained hypermedia approach that incorporates personal perspectives and a navigational language. This hypermedia differs from conventional approaches (like

Hypercard) in that text is broken down into sentence or paragraph-length nodes rather than pages and eventually built back up through selection by labeled links. That is, sets of nodes are selected and organized based on choices of perspectives and formulations of queries in a task-specific end-user language. New labels for links can be defined by end-users; they extend the semantics of the navigation language. Alternative versions of nodes can be defined; they are displayed in different perspectives. Displays are defined by statements in the language and take into account the current perspective and the link labels (Stahl, 1991). These techniques facilitate the creation of hypermedia webs of information for learners to explore, where the presentation of displays and available paths are tuned to the learner's needs.

Multimedia incorporating active simulations

The multimedia available with PLM includes text formulated in versions of SGML (the Standard Generalized Mark-up Language for digital text), such as HTML (the hypertext mark-up language used by the Web). It also includes text formulated for Mathematica and other simulations. These alternative description systems are translated and integrated into the PLM hypermedia system. This way, resources posted to the Web or developed in applications like Mathematica can be seamlessly integrated. Someone reading a PLM document can click on a region to download another Web document or execute a Mathematica computation.

A comprehensive sequence of personalizing mechanisms

As currently conceived, information in PLM passes through eight stages in moving from an educational digital library on the Internet to a display on the learner's computer monitor. (The eight stages are detailed below.) During each transition there are mechanisms to tailor the information to the learner's needs. This provides a comprehensive process of customization. Although the mechanisms that control the personalizing at each stage can be accessed by the learner and adjusted as much as desired, most of them can operate automatically behind the scenes. Initial choices in specifying personal preferences can be made by the learner, a teacher, or a software installer and then left alone. The learner can then simply select subject matter and options from menus.

TECHNOLOGIES FOR PERSONALIZING

Perhaps the most important advances in software in the next decade will come in technologies for personalizing information to individual users. A number of recent technological standards for hypertext set the stage for this advance. These standards include: the Dexter Hypertext Reference Model and SGML. They provide standard formats for hypertext that incorporate means for defining custom variations on the internal structure of the hypertext and on the display of hypertext documents. For instance, SGML specifies how to create Document Type Definitions (DTDs) that determine how various textual elements (titles, emphasized terms, embedded links) will appear in displays. World Wide Web documents conform to a specific SGML language, namely HTML.

Similar mark-up languages are used in software applications like Mathematica. Mathematica is a powerful program for the development of documents that include mathematical computations. Sections of the documents are labeled as titles, text,

computation inputs, etc. One can, for instance, write the equations of a 3-D graphical object and then activate the document section to display the computed graphic. The Mathematica language allows a user to specify how the graphic should appear: 3-D perspective, scale, grid lines, or axis labeling. A student reading a Mathematica document can edit the document to explore variations of the problem discussed and then decide how to display the result.

Until recently, most software was designed either for narrowly-defined tasks that required little effort by users but granted them little control (e.g., walk-up-and-use applications like ATM cash machines) or systems for dedicated specialists (like CAD programs for architectural drafting or professional programming environments) that require months or years to master. The high-functionality systems allow for open-ended expression, but at prohibitive cognitive costs. Personalizable software would be adjustable, so users could choose not only levels of power but which functions they want. These decisions could then be modified as user needs and experience evolve.

The next generation of software should empower users to personalize software and information displays without burdening them excessively.

Personalizable software should provide tools and mechanisms for users to structure information the way they want by indicating their desires in natural ways. PLM brings together a set of technologies that can work together to do this for learners accessing digital educational materials.

These technologies for providing user control of information were explored in HERMES[1], a system for NASA designers (Stahl, 1993b). The HERMES design environment was based on a philosophical analysis of human understanding and a theory of how best to provide computer support for designer's efforts to interpret new designs. The goal was to support personalized views of architectural drawings and of design rationale that corresponded to the designers' differing interpretive perspectives.

The system of hypermedia in HERMES incorporates fine-grained nodes and typed links, a perspectives mechanism, and a navigational language. The HERMES project built on research into intelligent hypermedia, including PHIDIAS (Stahl, McCall & Peper, 1992). It also incorporated the design environment approach of systems like JANUS (Fischer, Nakakoji, Ostwald, Stahl & Sumner, 1993b). These technologies were developed for providing high-functionality, knowledge-based computer support to professionals working on complex tasks.

Computational hypermedia, as seen in the "Design Rationale" window of Figure 5, makes many decisions on the presentation of material dynamically (i.e., while the computer system is running), based on information that the user has specified. Here, the displays are assembled dynamically to meet the specified needs of the reader. The reader can then revise the criteria or make explicit decisions on what to see next.

[1] HERMES ver. 2.0 is © 1994 by Gerry Stahl

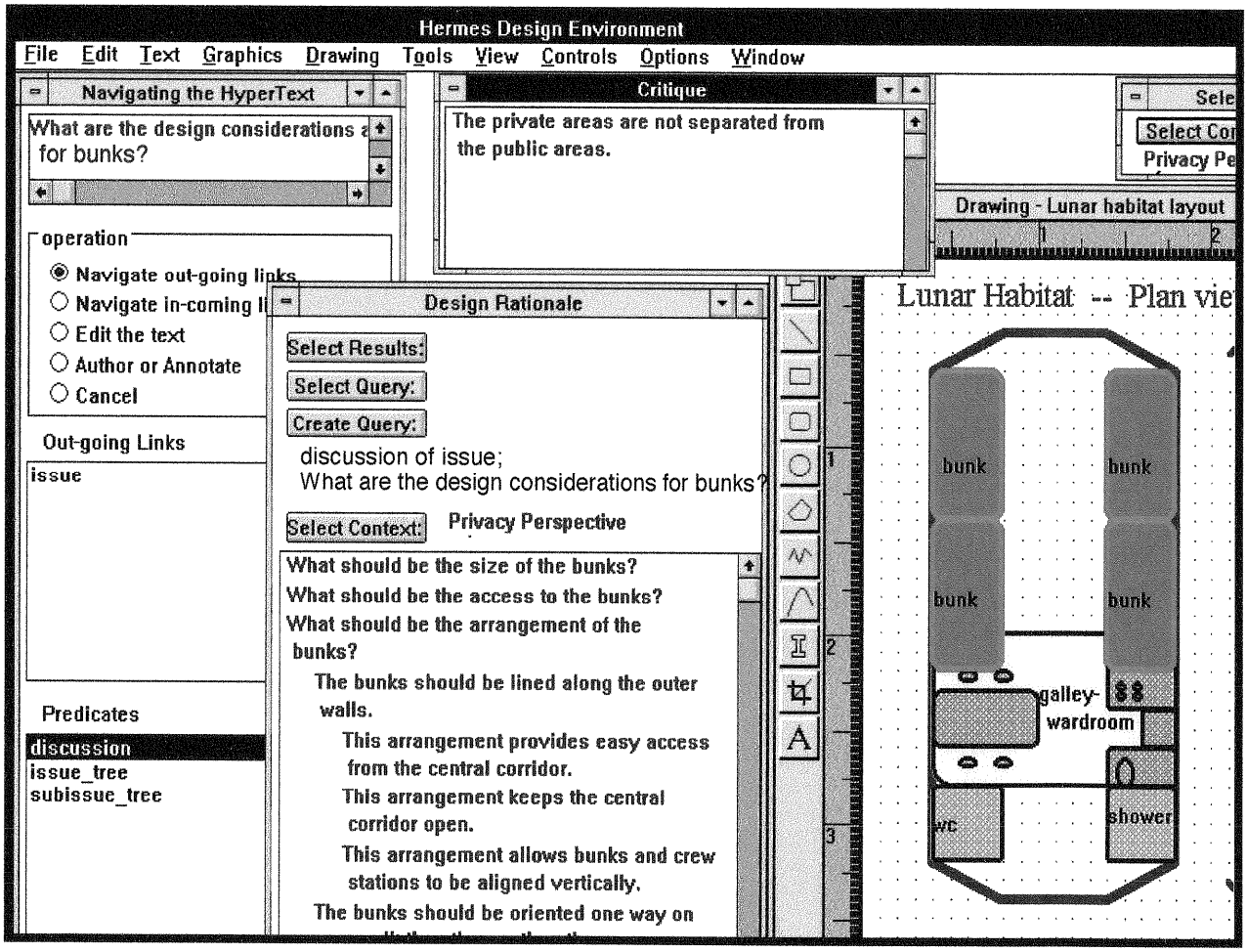


Figure 5. A screen image from the HERMES design environment. Note that the text in the central window has been dynamically assembled based on a statement in the end-user language (`discussion of issue`) and that a specific perspective has been selected (the `privacy perspective`). The text here is design rationale: a hierarchy of issues about the design of bunks, alternative answers to the issues, and arguments for these answers. Each issue, answer and argument is stored as a separate node; labeled links store their interrelationships; they are consolidated into information displays by the execution of language statements within perspectives.

THE SEQUENCE OF PERSONALIZING IN PLM

The technical approach to designing PLM integrates customization ideas, mechanisms, and industry standards from TCA, HERMES, and SGML.

Personalization in PLM takes place in eight sequential stages within the process of selecting materials from the digital library, analyzing them into hypertext nodes and links, and synthesizing selected contents into a personalized display. The stages are represented in Figure 6 and discussed below:

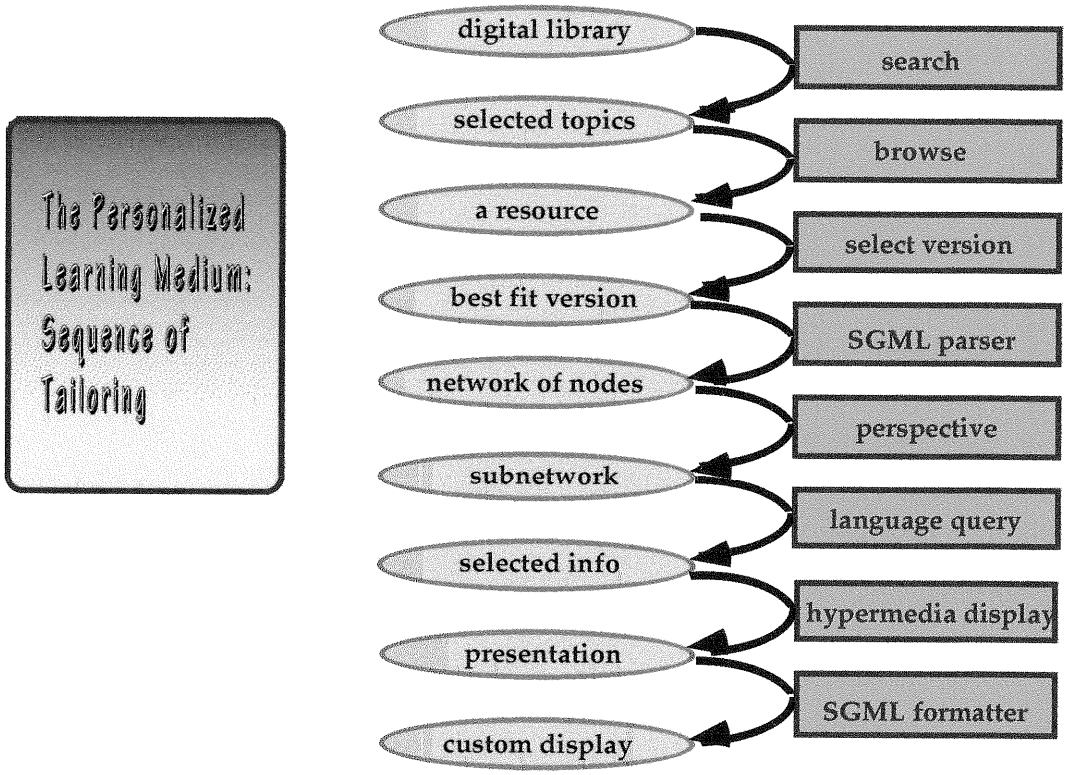


Figure 6. The sequence of personalizing mechanisms in PLM.

Stage 1. Searching for relevant materials.

The learner defines a *Profile* of the materials sought in the educational digital library. This profile includes characteristics of the learner as well; generally, the profile of the learner will not have to be altered frequently. For instance, the learner might request exercises involving the geometry of circles. The learner's profile might specify tenth grade mathematics ability, ninth grade reading level, and a preference for visualizations. It might also indicate availability of specific computer hardware and software. The profile is used by PLM to formulate a query that retrieves a selection of materials from the library. The profile functions as a user model for the software, but one that is under the control of the user.

Stage 2. Browsing among related resources.

Descriptions of selected materials are displayed in an *Explorer* window. The learner uses this interface to browse among related library resources, such as curriculum guides, software tools, historical background, useful mathematical techniques, relevant video clips. By providing for browsing within the confines of the Profile search, PLM gives the learner freedom to explore without the danger of becoming lost. Figure 3 showed an implementation of this synthesis of search and browsing for teachers in TCA. Other browsing tools are possible, including graphical maps of hypertext associations among materials.

Stage 3. Selecting the best fit version.

The digital library may include multiple versions of a given resource. For instance, a geometry problem might be approached using 2-D constructions, equations, or

computer programming. The learner can select the most appealing approach. In Figure 3, a *Versions* selector was integrated with the Profile and the Explorer.

Stage 4. Parsing into nodes and links.

Documents are broken down into their elements (as hypertext nodes), connected by typed hypertext links. The link types are based on the element's SGML markup type (e.g., "title"). This is done automatically by PLM. Custom node and link types may be defined by learners—or by their teachers.

Stage 5. Viewing from a perspective.

The learner's profile defines a *perspective*. The currently active perspective selects which nodes and links can be viewed. This allows multiple, redundant forms of information to be present in resources in the library, of which all but one form will be filtered out. For instance, many people may have annotated a particular resource, but a learner may want to filter out all annotations except her own, her teacher's and her classmates'. Then she would define her own perspective and have it inherit from her teacher's and her class' in order to view what they view in their perspectives.

Stage 6. Querying with the language.

All PLM displays are created dynamically by queries in the hypermedia navigation language. Statements in the language in effect specify starting nodes and types of links to traverse. Execution of a statement takes place within a selected perspective and results in a collection of linked nodes. This collection is the material selected for display. The language can be extended by end-users and terms in the language can have different meanings in different perspectives. By judicious naming of terms, users can construct sets of language statements that read like natural expressions in the task domain of interest.

Stage 7. Synthesizing a display.

PLM constructs a document from the collection of nodes and links. The document is marked up using a version of SGML. At this stage, the information retrieved from the library has been personalized.

Stage 8. Formatting the presentation.

The final stage is to display the information. The display format can be personalized by adjusting the mark-up definitions. For instance, the hierarchical design rationale in Figure 5 was indented by level. Alternatively, different levels could be italicized or text size and color adjusted to individual preferences.

Through these eight stages, standardized materials are selected and displayed in a way that can be tuned extensively by individual learners to their needs. A thoughtfully prepared document in the digital library can be personalized differently by each learner in the world.

SECTION V. PERSONALIZABLE SOFTWARE

Reflection upon the approach to personalizable software in TCA and PLM raises a number of general issues. Section V discusses several of these. It begins with the role of personal perspectives for organizing sets of choices made in personalizing information. Then it discusses the use of meta-info and meta-data for implementing functionality that empowers users with control over documents. It concludes by presenting a typology of approaches to personalizing, distinguishing the methods that have been used in different systems.

PERSONAL PERSPECTIVES ON INFORMATION

According to the philosophy of interpretation, everyone understands things from their own perspective. This perspective is based on their situation in history, in society, in their own life, in their work and in their general concerns: what they know, like and need; how they have been socialized; and how they interact with other people.

To be an effective learner is to be able to control information that one comes across and to personalize it within one's own perspective. Critical thinking—the hallmark of a sophisticated learner—consists in being able to uncover the perspectives that are implicit in information and to critique those perspective from one's own viewpoint. Ideology critique, for instance, lays bare the social mediations that have molded information so that one can evaluate the relevance of that information to one's own position and concerns.

Reality can only be viewed from perspectives; all knowledge is someone's interpretation.

There are several general implications of the perspectival character of information for the task of supporting education with software mechanisms: (i) information should be presented in ways that allow one to select different perspectives; (ii) one should be able to define and adjust one's own perspective; and (iii) one should be able to control definitions of perspectives. Control here means being able to make perspectives visible and changeable. For all perspectives, one should be able to see the definition of the perspective and to understand how it affects the display of information. For one's own perspective, one should also be able to modify its definition.

Such control over information has always been seen as desirable by visionaries of computer supported information exploration. In the inaugural discussion of the idea of hypertext libraries, Bush (1945) proposed that learners should be able to define their own paths through information. In the most recent vision of digital libraries, Negroponte (1995) emphasizes, "Being digital, whatever it means, means having your way." However, to date the need to personalize information sources has not received the attention it deserves in software system implementations.

One reason that systems to personalize information have not been pursued is that such control can be a double-edged sword. Most people do not have the time, skill or interest to organize the information that they use. Ideally, people want to delegate the organizing of most information, yet be in control of organizing very special elements. Delegation can take the form of trusting that some presumably reliable people have

already organized the materials or, given software agent technology, it can take the form of instructing computers in how to automate the process of organizing the information.

In general, one probably wants different levels of control over information, accepting the perspectives in which much information is embodied and personalizing other information to one's own perspective. The implementation of such control may also take place on different levels. For instance, an author may structure some material, a teacher may adapt parts of it and a learner may personalize other sections. Each of these people may do some of this work manually, some of it using pre-programmed software mechanisms and some using mechanisms that they have tailored for specific tasks.

IMPLEMENTING LEVELS OF CONTROL

Two programming principles are especially important for implementing computer support for levels of control:

- Provide information about information (meta-info).
- Migrate modifiable data up to levels of fixed program instructions (meta-data).

To empower users with control over information, provide them with access to meta-info and meta-data.

The following table illustrates these principles:

	role	*product*	*level*
hypertext	author	hyperdocument	information
	commentator	organization	meta-info
	reader	personal path	interpretation
Agentsheets	AS programmer	mechanisms	program
	AS title designer	simulation titles	meta-data
	AS player	instances of titles	data
PLM	PLM builder	build mechanisms	program
	PLM adapter	define structure	meta-content
	PLM author/reader	read & create text	content

Table 1. The use of meta-info and meta-data in systems of hypertext, in Agentsheets and in PLM.

In table 1 three roles have been distinguished for people involved with each of three systems: hypertext, Agentsheets and PLM. Imagine the development and use of a hypertext document. Originally, an author (perhaps a textbook writer) enters information into the hyperdocument. Then someone (perhaps a teacher or curriculum specialist) performs a middle role, organizing the material by defining sections, giving them descriptive labels and linking them together in useful ways. This defines meta-info about the original material, making it more valuable by

facilitating personalized use of the information. In the end, a reader (perhaps a student) blazes a personal path of exploration through the hyperdocument, using the meta-info to make informed choices about what to view.

Meta-info defines the linking that makes hypertext possible. But it also plays a role in prose. Note the extensive presence of meta-info in this paper to guide the reader through a dense argument: abstract, preface, reader's guide, table of contents, several layers of headings, figure text, cross-references, bibliographic references. There are also many implicit structural indicators, from grammatical ways of defining syntactic relationships to the repetition of names and terms that were discussed in previous sections. Many of these literary techniques of communicating meta-info to guide interpretation would be transformed in hypertext media. The danger in non-linear hypertext is that the reader will quickly become lost. Personalizing the hypertext can be used to re-impose structure—but this time under the reader's rather than the author's control.

Agentsheets was also developed with a three-level division of labor in mind. The program was created by a software developer to provide tools for simulation designers and behaviors for simulation viewers. The developer's product is a program. Periodically, he revises the program to meet new needs that title designers have reported. The designers build simulation titles within the Agentsheets environment. Their innovations do not change the Agentsheets program itself, but provide a level of meta-data that defines the look and behavior of components of the new simulation. For instance, the designer of the `Segregation` simulation defined ways of arranging houses and placing different sets of people in the houses with different rules about wanting to have neighbors who belong to their own set. These simulation titles are then shared across the Internet using *ARE*. Finally, simulation players can use the titles to set up specific situations to simulate. For instance, students using `Segregation` can define three racial groups, each of which wants to have 50% of their immediate neighbors be of the same race. The students can arrange icons representing people of these three races in a simulated village and see how they move around. The definitions at this level are simply data (e.g., location coordinates of icons) to Agentsheets. The possibility of programming a general simulation environment like Agentsheets instead of having to program each title from scratch is a consequence of migrating much of the simulation functionality down to the level of meta-data. What would otherwise have been simply part of the program source code has been made into a layer of variable data that controls the end-user's data.

PLM incorporates both these strategies. It structures documents with layers of meta-info, such as labeled links. It also treats decisions about structuring, like choices of perspectives, as meta-data. These are built into the PLM program at the first level. The specific interface facilities enable people to either personalize information directly or else personalize the automated mechanisms that structure the presentations of information. This work can be done by the end users—the learners who read, create paths and annotate the hyperdocuments—or by people in intermediate roles (teachers, curriculum specialists or other super-users) who define the way information will be structured for the learners. Ideally, the work of controlling the presentation of information can be shared among authors, adapters, readers and the

software support systems for each of them. This way the burden should not be too great on anyone, yet the level of control will be substantial and effective.

LEVELS OF PERSONALIZING

There have been some initial efforts in commercial software relevant to the goal of making information personalizable, although most of the work remains to be done.[2] Even the terminology to define this goal is yet to be worked out.

Various levels of personalizing can be distinguished. Different system approaches implement different forms of personalizability. These alternative forms represent specific trade-offs between what the user can do and what the system does automatically. Certain levels of personalizing are appropriate for specific domains, individual users and different use situations. It is useful to identify several categories or major forms that personalizing can take. In the following list the labels are somewhat arbitrary because there has been no consistent distinction of these various forms in the literature. Each of the following levels is discussed below:

- *Specify:* adjust settings to select values of predefined parameters; e.g., a user expertise level may be specified on a scale between `novice` and `expert`.
- *Tune*: adjust display presentations; e.g., set second level headings to print in **bold**.
- *Customize*: extend list of options with new categories or objects; e.g., add a new icon to a palette of representations.
- *Personalize*: make choices that change the program's global set of settings; e.g., choose an overall viewing perspective.
- *Tailor*: change tool functionality; e.g., add a spelling checker or a spreadsheet to an existing application.
- *Program*: define arbitrary system behaviors; e.g., change the source code defining how something is computed.

Specification components and user models

Within the tradition of autonomous artificial intelligence, the common approach to personalizing the presentation of information is to include a user model, i.e., a description for the software of characteristics of the person using the software. For instance, many tutoring programs build up a model of the student being tutored based on the history of the student's behavior using the program (e.g., what sorts of errors that student typically commits). Due to the emphasis on implementing autonomous behavior by the software, these user models were generally not visible or modifiable by the students or their teachers.

The idea of a specification component is to allow users to define the computer's model of what the user wants (Fischer, Nakakoji, Ostwald, Stahl & Sumner, 1993b). For instance, a person using the software could specify a level of difficulty, particular

---

[2] As Lanier (1995) points out, many current attempts to personalize software with agents and front-ends for novices insult the user's intelligence rather than augmenting it, as called for by Engelbart (1963).

topics of interest or other requirements. This is a mechanism for the sharing of meta-info between the computer and the user. Such shared information allows the software to draw more useful inferences. For instance, in TCA, the Profiler acts as a specification component allowing a teacher to define a user model relevant to the selection of curriculum. This is used by the TCA system not only to formulate search queries, but also to define critics for analyzing retrieved lesson plans. The critics are rules that compare meta-info in the indices of resources with meta-info in the user profile to determine whether particular resources are compatible with the needs of the teacher.

Display tuning

The display of information may be tuned to user preferences. The display of hypertext Web pages, for instance, is tuned by browsers like Netscape to the user's operating system: headings and other features look different on a Mac, on an ASCII (text only) terminal or in X-Windows. This is accomplished by encoding display information using the HTML standard, which is variously interpreted by the browsers. This technique is used extensively in PLM, although there the user has additional control over how the display is formatted, whereas Netscape does this tuning autonomously.

End-user customization

The list of features that can be specified and have their display tuned in a system like HTML is pre-defined and fixed. There are just so many levels of headings, etc. allowed. There is no way to define a new feature, like a call-out text.

End-user customization permits the definition of new features. For instance, in HERMES a user can add new names to the list of legal types of nodes and links. These names are also used in the navigation language, making the semantics of the language extensible. Similarly, the list of allowable perspectives can be extended, adding new perspectives (as meta-data) that may build on (inherit) existing perspectives. This kind of customization or extensibility is critical for knowledge-based systems because the definition of domains of knowledge is rarely fixed or independent of users and their innovative tasks (Stahl, 1995c).

Personalizable software

Users should be able to define how they want their software to behave for them. Personalizable software should request the name of the user and then reconfigure itself to do all the things that user specified in the past. If a user has entered a profile of specific interests, has tuned displays in certain ways, has customized lists of terms and has extended palettes to include new items, then all of this should be made automatically available to that user.

The personal perspectives mechanism provides a way of organizing this association of comprehensive changes with individual users. In a program like HERMES that incorporates perspectives, all work by a user is carried out in a perspective. Thus, everything created is associated with some perspective. In the simple case, users always use their own perspectives. Then, their changes are always available to them.

The perspective mechanism also allows for shared perspectives for collaboration. Here, several users can work in one perspective and share specifications, tunings or customizations made in that perspective. They can each also have their own

perspectives with their private personalizations. Their private perspectives can inherit from their shared perspective, so that they can take advantage of group changes as well as their private ones. Inheritance of perspectives allows people to build up sets of personalizations by choosing among existing perspectives to inherit from and reuse. This means that one does not have to personalize a system from scratch, dramatically reducing the overhead otherwise involved in personalizable software.

Tailorable functionality

The trend in the next generation of commercial software is to support personalizing through component-ware. OpenDoc, OLE 2 and development environments like VisualBasic allow software functionality to be developed in independently compiled components. Users can then build their personal system by combining useful components. Everyone can have their own personal toolbelt of components—just like every carpenter meets his or her special needs with a unique toolbelt full of standardized tools. Unfortunately, the market is not yet ready for people to create their own software this way. When they do, the available components will still have to include mechanisms like specification components, display standards, perspectives and task-specific languages in order to support the forms of personalizability discussed above. It is not clear how independently developed components will be able to share meta-info and meta-data adequately to have these personalization components work effectively across them.

Programmability

There are many levels of programming: graphical direct manipulation, visual programming systems, task-specific end-user scripting languages and general purpose programming languages. Each has its advantages in power, disadvantages in cognitive load and appropriateness to specific uses. As a means to personalizing software, programmability probably serves best as a last resort. When specifying, tuning, customizing, personalizing and tailoring cannot accomplish what one wants then programming may be necessary. In many cases, a task-specific language can meet this need without imposing undue requirements on the user. Task-specific languages can use the visual conventions or textual terminology of the domain and can restrict the syntax through direct manipulation or predefined templates in order to minimize the cognitive load on the user.

SOME GENERAL LESSONS CONCERNING PERSONALIZABLE SOFTWARE

This paper has been concerned with computer support for learners and their teachers using resources from the Web. The systems that were described above, TCA and PLM, illustrated a comprehensive approach to global sharing of educational resources combined with local personalizing of relevant materials. Computer support for this takes the form of personalizable software. A number of issues related to the design, implementation and use of personalizable software were discussed in this final Section. It is notable that many of the issues concerning software that personalizes documents apply to personalizing that software itself. Hence the productive ambiguity of the term *personalizable software*.

POINTERS TO FURTHER INFORMATION

Ambach, J, Perrone, C, Repenning, A (1995) "Remote Exploratoriums: Combining Networking Media and Design Environments. *Computers and Education*. **24** (3) pp. 163-176.

Bush, V (1945) "As We May Think." *Atlantic Monthly*. 176 (1) pp. 101-108. Reprinted in (Greif, 1988).

Engelbart, D (1963) "A Conceptual Framework for the Augmentation of Man's Intellect." Reprinted in (Greif, 1988).

Fischer, G, Nakakoji, K, Ostwald, J, Stahl, G, Sumner, T (1993a) "Embedding Computer-Based Critics in the Contexts of Design." *Proceedings of InterCHI '93. Conference on Human Factors in Computing Systems*, Amsterdam, April 1993. pp. 157-164.

Fischer, G, Nakakoji, K, Ostwald, J, Stahl, G, Sumner, T (1993b) "Embedding Critics in Design Environments." *The Knowledge Engineering Review*, **4** (8), Dec. 93. pp. 285-307.

Fischer, G, Stevens, C (1991) "Information Access in Complex, Poorly Structured Information Spaces." *Proceedings of CHI '91. Conference on Human Factors in Computing Systems*, New York, April 1991. pp. 63-70.

Fischer, G, Nieper, H (1987) "Personalized Intelligent Information Systems." *Workshop Report, Breckenridge, CO.* Technical Report 87-9. Institute of Cognitive Science, University of Colorado at Boulder.

Greeno, J (1993) "For Research to Reform Education and Cognitive Science." In *The Challenge in Mathematics and Science Education: Psychology's Response.* Edited by Penner, L, Batsche, G, Knoff, H, Nelson, D. APA Press. 1993.

Greif, I (1988) *Computer-Supported Cooperative Work*. Morgan Kaufmann.

Heidegger, M (1927) *Being and Time*. Harper & Row. 1962.

Lanier, J (1995) "Agents of Alienation." *Interactions*. **2** (3) pp. 66-72.

Lave, J, Wenger, E (1991) *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press.

NCTM (1989) *Curriculum and Evaluation Standards for School Mathematics*. National Council of Teachers of Mathematics.

Negroponte, N (1995) *Being Digital.*

Piaget, J (1927) *The Language and Though of the Child*. Meridian Books. 1955.

Repenning, A, Sumner, T (1995) "Agentsheets: A Medium for Creating Domain-oriented Visual Programming Languages." *IEEE Computer*. March. pp. 17-25.

Stahl, G (1995a) *A Personalized Learning Medium*. Proposal to NSF/SBIR from Owen Research Inc. June 1995.

Stahl, G (1995b) *The Teacher's Curriculum Assistant: A Curriculum Repository and Development Environment to Support SMET Educational Reform*. Proposal to NSF/NIE from Owen Research Inc. April 1995.

Stahl, G (1995c) "Supporting Interpretation in Design." Submitted to *Design Studies*. Special issue on Design Cognition and Computation. 18 pages.

Stahl, G, Owen, R (1995) "Armchair Missions to Mars: Using Case-Based Reasoning and Fuzzy Logic to Simulate a Time Series Model of Astronaut Crews." Submitted to *Knowledge-Based Systems*. 10 pages.

Stahl, G, Sumner, T, Owen, R (1995) "Share Globally, Adapt Locally: Software to Create and Distribute Student-centered Curriculum." *Computers and Education*. Special issue on Education and the Internet. **24** (3) pp. 237-246.

Stahl, G, Sumner, T, Repenning, A (1995) "Internet Repositories for Collaborative Learning: Supporting Both Students and Teachers." Forthcoming in *Proceedings of Computer Support for Collaborative Learning*. October 17-20.

Stahl, G (1993a) "Supporting Situated Interpretation." *Proceedings of the Cognitive Science Society: A Multidisciplinary Conference on Cognition*. Boulder, 1993. pp. 965-970.

Stahl, G (1993b) *Interpretation in Design: The Problem of Tacit and Explicit Understanding in Computer Support of Cooperative Design*. Ph.D. dissertation. Department of Computer Science. University of Colorado at Boulder. Technical report CU-CS-688-93. UMI dissertation services, Ann Arbor, MI, order no. 9423544. 451 + xiv pages.

Stahl, G (1992a) *A Computational Medium for Supporting Interpretation in Design*. Technical Report CU-CS-598-92. Computer Science Department, University of Colorado at Boulder. 39 pages.

Stahl, G (1992b) *Toward a Theory of Hermeneutic Software Design*. Technical Report CU-CS-589-92. Computer Science Department, University of Colorado at Boulder. 16 pages.

Stahl, G, McCall, R, Peper, G (1992) "Extending Hypermedia with an Inference Language: an Alternative to Rule-Based Expert Systems." *Proceedings of the IBM ITL Conference: Expert Systems (October 19-21, 1992)*. pp. 160-167.

Stahl, G (1991) *A Hypermedia Inference Language as an Alternative to Rule-Based Expert Systems*. Technical Report CU-CS-557-91. Computer Science Department, University of Colorado at Boulder. 23 pages.

Stahl, G (1975) *Marxian Hermeneutics and Heideggerian Social Theory: Interpreting and Transforming Our World*. Ph.D. dissertation. Department of Philosophy, Northwestern University. Evanston, IL. *Dissertation Abstracts* **36** (7) order no. 75-29,759. 372 pages.

Vygotsky, L (1936) *Thought and Language*. MIT Press. 1986.

RESEARCH AGENDA CONTENTS

Abstract 1

Preface: In the ideal world 1

Section I. Personalizing Agentsheets and the Remote Exploratorium 2

Phase 1: A personal user's guide 3

Phase 2: Personal curriculum 3

Phase 3: A personal language in AgenTalk 3

Phase 4: Everything is personal 4

Phase 5: Personal retrieval 4

Fulfilling the promise of the Web 5

Section II. Personalizable learning 5

Taylorizing the student *versus* tailoring by the learner 5

Supporting authentic exploration with personalized content 6

Plato's concept of education 6

Rousseau's subtle role for the teacher or self-learner 7

Vygotsky's zone of proximal development 7

Lave's community of practice 7

Learning as interpretation 8

Extending cognition to meet the challenges of the future 8

Need for curricular contexts of projects 9

The problem and the promise 10

Section III. Teacher's Curriculum Assistant 10

Need for computer support of curriculum development 11

The TCA approach 12

A TCA prototype 13

Section IV. Personalizable Learning Medium 17

Overview of PLM 17

Technologies for personalizing 18

The sequence of personalizing in PLM 20

Section V. Personalizable software 23

Personal perspectives on information 23

Implementing levels of control 24

Levels of personalizing 26

Some general lessons concerning personalizable software 28

Pointers to further information 29

Research Agenda Contents 31