A System Design Framework-Driven Implementation of a Learning Collaboratory

Michael C. Dorneich

***Abstract*—This paper describes a design process to support the development of a learning collaboratory, a distributed, computer-based, virtual space for learning and work. A learning collaboratory: as a distributed distance learning environment, offers tremendous opportunities to expand the way people teach and learn and to broaden educational opportunities to an ever increasing range of learners. The challenge is to design distance learning technologies that engender meaningful learning experiences that take full advantage of the power of computer-mediated communication to support innovative learner-centered and collaborative interactions between students, teachers, subject experts, and resources. The first half of this paper describes the Learning Collaboratory Design Framework (LUCIDIFY), a design process that integrates methods and concepts from cognitive systems engineering, theories of learning and instruction, distributed computing, and computer-supported collaborative learning (CSCL) to guide the principled design of learning collaboratories. The second half of the paper describes how LUCIDIFY was used in the design and implementation of the Collaborative Learning Environment for Operational Systems (CLEOS), a learning collaboratory for teachers, students, and practitioners in the physical sciences, in particular for NMR spectroscopy and X-ray diffraction experiments. CLEOS features two virtual instrument tutorials, an asynchronous messaging system, a project-based design and management application, and a collaborative multi-user domain infrastructure.**

***Index Terms*—Abstraction hierarchy, collaborative learning environment, collaboratory, distributed computing.**

I. INTRODUCTION

THE emergence of computer technology has had profound implications for instruction. Early efforts consisted of practical instructional software applications designed around classroom needs and focused on the transmission or delivery of knowledge to the student. Subsequent work in intelligent tutoring attempted to emulate the behavior of skilled human tutors in software. More recent work has focused on collaborative learning processes, and the idea of a *collaboratory* (i.e., collaborative laboratory) as a virtual space for learning and work. This paper describes a conceptual framework to guide the design of a learning collaboratory.

The design of a collaborative learning environment should promote mutually beneficial interactions between students, teachers, and practitioners. It should foster a community of participants who are bound by a common interest. If the users of the environment find software applications and people that are beneficial, then the community will grow. The ability to nurture and sustain a community of learners, teachers, and practitioners facilitates the long-term usefulness and ongoing evolution of the learning environment.

The developer of a learning collaboratory has many challenges in the process of creating a virtual learning environment and nurturing a user community. The challenge to the developer of such a learning collaboratory is to understand the subject domain, the user community, how to best support learning in this environment, how to best support user interactions, and how to provide enough value for all users.

The developer of a learning collaboratory needs to consider types of knowledge that need to be captured and modeled: the work domain (i.e., "the [goals, resources, and constraints] within which the work takes place" [[43, p. 28])] of the particular discipline (i.e., "what to teach" in NMR spectroscopy, for example), the work domain of instruction (i.e., "how to support effective learning" where high level goals involve teaching and instruction), and the more traditional work domain of the collaborative systems developer (i.e., "what to design").

The primary mission of a learning collaboratory is to teach learners to master subject material and techniques and to develop learning and teamwork skills. Different levels of competence may require different pedagogical support. The development of the novice participant into an expert is a process that goes through many stages and so the learning collaboratory must scaffold the development process by providing flexible instructional support at every level of the process of learning.

A learning collaboratory, as a distributed distance learning environment, offers tremendous opportunities to expand the way people teach and learn, and to broaden educational opportunities to an ever increasing range of learners. Additionally, a learning collaboratory can serve as an environment to explore opportunities for novel instructional techniques, collaboration among both students and teachers, and the delivery potential of student tutorial applications and educational content. Thus the developer of a learning collaboratory can explore a wide range of pedagogical strategies of learning, with collaborative pedagogical strategies being especially germane.

Collaboration, especially in support of learning, is a defining aspect of a learning collaboratory. Technology to facilitate communication is a vital part of enabling a community of learners to grow within the virtual environment. The developer must consider how to best utilize collaborative technologies to create an environment that support collaboration between community members.

Manuscript received July 7, 2000; revised March 31, 2002. This paper was recommended by Associate Editors P. M. Jones and C. W. Mitchell.

M. C. Dorneich was with the Department of Mechanical and Industrial Engineering, University of Illinois, Urbana, IL 61801, USA. He is now with Honeywell Laboratories, Honeywell International, Minneapolis, MN 55418 USA (e-mail: michael.dorneich@honeywell.com).

Publisher Item Identifier S 1083-4427(02)06004-6.

This paper describes the Learning Collaboratory Design Framework (LUCIDIFY), which explicitly addresses the four needs outlined above: modeling domain knowledge, characterizing the user community, explicit support for collaborative learning pedagogies, and collaboration support for a heterogeneous user community. The next section will delve more deeply into the motivations and background of the design of a learning collaboratory. Section III will describe LUCIDIFY and the human-centered design processes that will facilitate design of a learning collaboratory. LUCIDIFY is a general framework for the design of any learning collaboratory. The remainder of the paper will describe how LUCIDIFY was used in the design and implementation of an actual learning collaboratory: the Collaborative Learning Environment for Operational Systems (CLEOS). More specifically, Section IV will describe and define the scope of CLEOS, identify the design goals, and detail the process by which LUCIDIFY was used to design CLEOS. Section V will describe the architecture of CLEOS, and briefly introduce the software applications that populate the learning collaboratory. Finally, Section VI will describe some implementation details of CLEOS and how CLEOS addresses issues of distributed computing and issues of collaboration.

II. Background

A *collaboratory* is a virtual environment that uses technology to mediate communication of nonco-located colleagues who share common interests, tasks, or research areas. *A collaborative virtual environment* is defined by Jones as "an interactive, computer-generated environment that incorporates some level of semantics of work practice and supports multiple human users both synchronously and asynchronously." [32, p. 1]. Synchronous communication refers to simultaneous, "real-time" interaction (e.g., talking on the phone), while asynchronous communication does not happen at the same time (e.g., communicating via e-mail over the course of a day). Technology and software applications allow participants to collaborate and share access to information, instrumentation, and colleagues [22], [33], [39], [55]. A *learning collaboratory* is a collaborative virtual environment where students, teachers, and experts in field or domain work together in a variety of ways to support student learning. In an educational setting, there are multiple reasons to support distributed instruction. Distributed, collaborative educational software applications can provide a wide range of students with increased instructional facilities and learning opportunities that are significantly different than the traditional teacher-centered lecture format. A learning collaboratory, with an emphasis on collaboration among learners engaged in realistic experiences rather than passive listening, shifts the emphasis of learning from teacher-centered (activity centered on the teacher, where students passively listen) to learner-centered (students actively engaged in learning activities, where teacher is facilitator of student activity). The challenge is to design distance learning technologies that engender effective learning experiences that do not just mimic existing teacher-centered learning practices, but take full advantage of the power of computer-mediated communication to support innovative learner-centered and collaborative interactions between students, teachers, subject experts, and resources. Thus, a learning collaboratory can be a useful augment to traditional teaching methods. As financial resources become increasingly scarce, providing affordable and direct access to educational resources becomes important in maintaining a high level of education. Furthermore, if a learning community is defined not only as students and teachers, but as practitioners and researchers as well, then supporting interaction among these parties is likewise critical. In settings where collaboration is an important and natural mechanism for learning and instruction, learners can acquire valuable team skills, something that is often cited as a particular deficiency with graduating college students [63].

The learning collaboratory design framework to be introduced in this paper starts with a vision of the future with a more learner-centered approach to instruction. Rather than students learning in isolation, group-based learning is emphasized. Students who previously were disenfranchised now have multiple avenues to learn, with flexible instruction methods that can be more closely tailored to individual learning styles. Rather than spending the majority of their time reading books and working through paper examples, students are exposed to and engage in a broader range of learning activities. The learning activities take the form of realistic practice of the phenomenon (if not in a real setting, then in a realistic simulation of the setting). For example, virtual instruments (e.g., computer-based tutoring systems) provide a meaningful context in which to learn operational procedures. Computers are used to actively engage students, rather than being just another passive information resource. Technologies enable a wider community to develop, providing learners with more opportunities to collaborate and interact with peers, students, and experts. Providing mechanisms to support communication between students engenders an environment where students have opportunities for peer-to-peer learning as well as apprenticeship learning with a "more capable peer" [59]. This increases a student's access to different types of instructional resources. Finally, teachers can use this same technology to collaborate with other teachers, learning from each other and even collaboratively developing curricula.

A wide range of issues must be addressed in building a learning collaboratory, including cognitive systems engineering issues in analyzing a work domain (i.e., "the [goals, resources, and constraints] within which the work takes place" [43, p. 28]), pedagogical issues in supporting effective distance learning through realistic experiences, distributed computing issues in managing data and processes in a heterogeneous computing environment, and methodological issues in designing and evaluating a learning collaboratory. This paper addresses these issues in a proposed conceptual framework, LUCIDIFY. This conceptual framework is used to design a software system that is a learning collaboratory for the physical sciences, the collaborative learning for operational systems (CLEOS). CLEOS is a distributed collaborative software application for teaching the procedures and theory of physical science experiments. As such, it provides a virtual environment where nonco-located learners can gather to learn about spectroscopy and X-Ray diffraction via two collaborative computer-based tutorial systems (the collaborative virtual spectrometer (CVS) and the virtual X-ray diffractometer (VXRD). Project-based and collaborative learning is supported via the collaborative tutorials and a learning project management application (the tool for organizing and supervising projects (TOSP), where groups track progress of their projects and teachers collaborate to create projects. Students working individually with a tutoring system may interact with experts via an asynchronous messaging system (the Question Board [QB]). Students working together in groups synchronously use the tutoring systems, real-time discussion capabilities, and the QB to communicate.

III. Learning Collaboratory Design Framework

The use of Collaboratories to facilitate learning via heterogeneous, distributed knowledge networks was first proposed at a National Science Foundation workshop in 1989, then later promoted in a policy statement by the National Research Council in 1993 [33]. The term "Collaboratory" (i.e., Collaborative Laboratory) is used to describe a new kind of virtual environment, and is defined as "a tightly coupled knowledge network supported by advanced internet-based, computing and collaboration technologies. These technologies typically include digital libraries, synchronous and asynchronous collaboration tools, and on-line instrumentation including remote sensing, modeling and simulation, and data analysis tools" [33].

Other definitions abound [55], [39], [22]; they have in common the idea that technology can be used to mediate communication of nonco-located colleagues in an environment dedicated to a specific group, topic, or research area. Technology and software applications allow participants to collaborate and share access to information, instrumentation, and colleagues. Early examples of successful collaboratories include the Upper Atmospheric Research Collaboratory (UARC) [55], the Learning Through Collaborative Visualization (CoVis) Project [11], the Environmental Molecular Sciences Collaboratory [47], the International Personality Item Pool [25], TANGO [56], Keck Observatory Collaboratory [36], and the Collaboratory for Microscopic Digital Anatomy (CDMA) [25].

LUCIDIFY is a framework for design that combines methods to structure domain knowledge, represent navigational strategies, characterize expertise, and support collaborative learning and work. The framework, represented in Fig. 1, is a collection of methods and procedures that can be used to guide a principled design.

LUCIDIFY represents a generic set of guidelines to be used when designing a learning collaboratory. The bottommost level, Domain Knowledge, of Fig. 1, refers to a structured representation of "what to teach". This representation provides a basis for making knowledge inspectable and sharable. Learning can be accomplished in a virtual environment by situating the problem solving or learning experience in a realistic simulation of the domain, and thus an articulated domain model is a key component for the proposed design framework. The work domain can be represented via an abstraction hierarchy [42], [43], [57]. Problem-solving strategies can be represented as navigational strategies through the user's conception of the work domain.

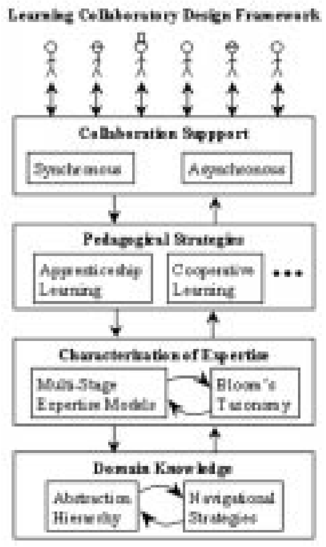


Fig. 1. Learning Collaboratory Design Framework.

The next level of Fig. 1, Characterization of Expertise, refers to characterizing the capabilities of the learner by understanding the expertise levels of the learners in order to support their stage in the learning. Multi-stage expertise models can be used to model the capabilities of users. Bloom's taxonomy is an example of a model of learner behaviors. By modeling both learner capabilities and behaviors, developers of a learning collaboratory can explicitly design applications that support users at multiple levels of expertise with support for multiple types of learning behavior.

The third layer of Fig. 1, Pedagogical Strategies, refers to which pedagogical strategies will be supported in the learning collaboratory design. Since a learning collaboratory is by its nature a distributed collaborative system, consideration of collaborative learning pedagogies play an important role in determining how to teach a wide range of students in a virtual environment. Apprenticeship Learning and Cooperative Learning are but two examples of the host of collaborative learning strategies that that the collaboratory developer may what to consider.

Finally, the topmost layer of Fig. 1, Collaboration Support, refers to the developer's task of supporting collaboration in a distributed learning environment. Since collaboration is so important in how students are taught, multiple forms of collaboration are included in the design of the collaboratory: synchronous collaboration for real-time (simultaneous) interaction and asynchronous collaboration for collaboration that takes place at different times.

Note that the learning collaboratory is explicitly designed for a heterogeneous user population, including students, teachers, and experts (hence the different stick figures atop Fig. 1). The learning collaboratory design should account for all classes of users (not just students) and support teacher activities, encourage experts to interact with the community, as well as support the targeted student population.

LUCIDIFY maps out the high-level issues that must be addressed when designing any learning collaboratory. To summarize, the approach taken in this work rests on four conceptual areas: 1) mapping the domain of practice in a structured way, with explicit consideration of strategies of domain understanding; 2) characterization of capabilities of the learner; 3) pedagogical support for situated practice in an realistic domain; and 4) collaboration as a mechanism for effective learning and work. Situated collaborative learning in an environment of realistic practice is the goal of a design based on this framework.

A. Domain Knowledge

Many factors must be considered when designing a collaborative learning environment. High level goals of the learning environment, general functions the environment is to support, a characterization of the user community, and a pedagogy of learning. Domain knowledge is a critical component of educational technologies; an explicit representation of "what to teach" provides a basis for making knowledge inspectable and sharable in a collaborative learning environment. Thus modeling of the domain at many levels of abstraction is a key component for the proposed design framework. One well-developed framework for modeling domain knowledge at multiple levels of abstraction is the *abstraction hierarchy* [42], [43], [57].

The abstraction hierarchy (AH) is used to represent the means-end (causal) structure of the domain of practice. The abstraction hierarchy provides a rich representation of the work domain at multiple levels of abstractions and is ideally suited to map the domain's goals, values and priorities, functions, activities, and resources in a meaningful way. An AH typically has five "levels" (or rows of elements), where each level represents an entire description of the system at one level of abstraction [45].

1) *Functional purposes* are the goals, across the work domain, of interest to anyone with a stake in the organization.
2) *Value and priority measures* indicate how well the functional purposes are served by the purpose-related functions.
3) *Purpose-related functions* are the "general" functions of the work domain that are carried out in order to achieve the functional purposes.
4) *Object-related functions* describe the activities and processes that use the physical objects to achieve the purpose-related functions.
5) *Physical objects* are the resources or physical objects of the domain.

One critical feature of an abstraction hierarchy is that elements at different levels are linked by causal (means–end) relationships. Properties at one level satisfy the functions at the level above and provide the reasons for the functions at the level immediately below. The links between the elements of different levels represent the relationship of the connected elements. When considering an element at one level, connections upward answer the question "why" and connections downward answer the question "how." The AH itself provides a useful metric of completeness of description in that each component must have connections in both directions.

Typically, the AH has been used to describe simulations of specific engineering systems (i.e., [5], [44], [58]). Rather than describing work domains, the typical use of AH has been to describe "isolated engineered instantiations and their "local" production goals under which these systems are operating within their respective work domains." ([45, p. 1]). The AH is more than a systems analysis tool for literal physical systems; it is a tool to model the work domain in which the system was designed to operate. Rasmussen defines a work domain as "the landscape within which the work takes place." ([43, p. 28])

There are several types of knowledge that need to be captured and modeled: the work domain of the particular discipline (i.e., "what to teach" in NMR spectroscopy, for example), the work domain of instruction (i.e., "how to support effective learning" where high level goals involve teaching and instruction), and the more traditional work domain of the systems developer (i.e., "what to design"). The abstraction hierarchy can be seen as a conceptual representation of a domain of knowledge.

In addition to learning the structure of the domain, it is important for learners to learn how to solve problems as they "navigate" their knowledge of the domain. Traditionally, descriptions of *navigational knowledge* stages are used to explain how people learn to navigate in a geographical domain. In this work, the idea of "navigational knowledge" is used as a metaphor for expertise, i.e., the ability of a student to learn to navigate through a conceptual domain of knowledge. Thus the metaphor employed is stages in which someone may learn to navigate their conceptual representation (map) of the domain. Each stage can be considered a phase of learning [3] and a characterization of current knowledge. Thorndyke [51] identifies three types of navigation knowledge [61].

1) *Landmark Knowledge*: orientation exclusively via highly salient visual landmarks. This provides a skeletal frame of reference around which the following phases of learning are built upon. The analogy here is that a novice or apprentice learner knows only a few facts and enough connections between them to effectively solve problems.
2) *Route Knowledge*: understanding is characterized by the ability to navigate from one location to another. Route knowledge, a highly egocentric frame of reference, is based on recognition of visual features, and categorical statements of action. The analogy here is that moderately skilled learners are able to follow certain routes or strategies when solving problems, but quickly "lose their way" when the problem changes in unexpected ways.
3) *Survey Knowledge*: knowledge resides in an internalized "cognitive map" [52], as an analog to a true physical map. Survey knowledge is a world-based frame of reference. The analogy here is that the learner has progressed to the point where a conceptual map of the environment begins to form, and further skill acquisition expands and deepens the conceptual representation of the domain.

Learning is a process that follows a path from present knowledge to the incorporation of new concepts within the existing structure of their knowledge representation. If the goal of instruction is to expand the conceptual representation (e.g., in scope, interconnections of concepts, and richness) of the learner, then the instruction pedagogy must consider the capabilities and strategies of the learner. In addition, the teaching of new strategies to the learner may be as important as the teaching of the concepts themselves. This interaction of knowledge of the structure of the domain knowledge with the learner's strategies of action form the basis of a characterization of the user's capabilities within the domain of practice.

B. Characterization of Expertise

Modeling of the domain at many levels of abstraction and the identification of problem solving strategies are key components for the proposed design framework, considering the hypothesized ways in which expertise develops, i.e., from low level details to greater levels of abstractions. A property of expertise is the use of abstract representations of the domain, rather than detailed or low-level representations [62].

When studying how a learner migrates from novice to expert, an effective representation of expertise is necessary. Many researchers have created models of expertise [38], [19], [21] that describe the characteristics of experts [48], [62], [41] and novices [38] as well as their problem-solving strategies [48], [4]. Early models of expertise focused on identifying differences between novices and experts, without considering the stages in between [21]. The maturation of from novice to expert involves shifts in domain knowledge, problem-solving strategies, and levels of abstraction.

Thus the evolution of expertise can be framed in terms of a description of the person's conceptual representation of the domain and their strategies for action. Different levels of competence may require different pedagogical support. The development of the novice participant into an expert is a process that goes through many stages. A guiding principle of the LUCIDIFY approach is to scaffold the development process by providing flexible instructional support at every level of expertise.

C. Pedagogical Strategies

Pedagogical strategies (i.e., strategies of instruction) are based on theories of learning. Examples of learning theories that are collaborative in nature include situated action theory, apprenticeship learning, problem-based learning, and cooperative learning. **Situated action theory** emphasizes the local management of activity as mediated by relevant environmental cues [50], [1]. The implications for learning are that appropriate actions are generated from a recognition of appropriate opportunities given the context. **Apprenticeship learning** is a means through which situated learning can occur, where apprentices are active participants in an activity, usually with an expert. Apprentices' process of learning moves from peripheral to full participation in the activities of a community of practice, as the expert "fades" from engagement of the activity. **Problem-based learning** is an example of a collaborative, learner-directed method of instruction where a small team of students, together with a tutor or coach, learn in the process of working through a problem [37]. **Cooperative learning** is a more general strategy in which students work together toward similar goals. Cooperative learning views learning as a process of active construction of knowledge, and that process can be facilitated by social interaction. Cooperative learning tends to happen naturally when students have face-to-face interactions, such as learning that occurs around a table in a student study group. Social skills themselves should be nurtured and explicitly monitored by the teacher. The teacher's role is to act as a monitor of group interactions and to provide students with suggestions on social skills. The students themselves are also responsible for monitoring and evaluating their own progress in both the academic and social processes within the group [29] (as cited in [30], also see [49]).

A related alternative view of instruction is organized around Bloom's Taxonomy of cognitive learning [6]. Learning is demonstrated by knowledge recall and the intellectual skills of comprehending information, organizing ideas, analyzing and synthesizing data, applying knowledge, choosing among alternatives in problem-solving, and evaluating ideas or actions.

D. Collaboration Support

Collaboration, especially in support of learning, is a defining aspect of the design framework. Technology to facilitate communication is a vital part of enabling a community of learners to grow within the virtual environment. Apprenticeship learning emphasizes the interaction between expert and learner as the fundamental building block of the learning process. Thus collaboration, cooperation, and communication between expert and apprentice is important. Collaborative learning, for instance teams of students working together on a project, is another explicit design goal. Cooperative learning emphasizes the benefits of face-to-face interaction between students. In a virtual environment, face-to-face interaction is usually not possible, and is replaced by computer-mediated communication. Thus it is important to identify what aspects of face-to-face interaction support group learning and design a system to support those aspects.

Collaboration can take many forms, and require various technologies in their support. Learners can collaborate synchronously or asynchronously. They can collaborate by engaging in the same activity, or they can each work on interacting but separate activities in pursuit of a larger goal. They can share equipment, or coordinate their use of separate tools. The approach outlined here makes every attempt to provide an environment where participants can learn while engaging in meaningful activities in an realistically represented domain. Situating learning activities in authentically simulated practice grounds much of the instructional practice.

E. Iterative Design Processes

A wide range of issues must be addressed in building a learning collaboratory, including cognitive systems engineering issues in analyzing a work domain, pedagogical issues in supporting effective distance learning in an realistic manner, distributed computing issues in managing data and processes in a heterogeneous computing environment, and methodological issues in designing and evaluating a learning collaboratory. LUCIDIFY gives structure to the design process by giving explicit consideration to these areas of design, and suggesting modeling tools to address these issues. Associated with LUCIDIFY are a series of human-centered design processes that have proven effective in the process of designing a learning collaboratory.

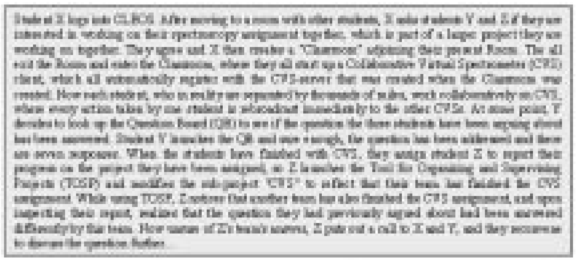
The principle philosophy driving the design approach is the notion that the needs of the community and the tasks they perform are of paramount importance. Thus, the users who are and will be performing the tasks are the greatest source of information and understanding of the tasks themselves. Consequently, in the early stages of development effort, much of the focus is on constructing an understanding of the requirements of the tasks and the community. Developing models of the organization via the abstraction hierarchy is part of this process. The elements of the organization are identified and codified as a set of objects in the object-oriented modeling approach. Activities within the domain of practice are identified, and models of system use are developed. All of this is done with as much input as possible from the potential users of the design, for example learning community members such as teachers, students, and experts. Thus there is a heavy task focus to the participatory design [40] aspects of the design process. Models of user activities (encapsulated use scenarios) and functions (captured by the AH), are created, which in turn helps drive the development of software applications to support practice. Rapid prototyping [40], with formative evaluations [53] as an integral part of the iterative process, helps to further develop robust models of use. The scenario-driven design approach [7] helps operationalize what is learned about the activities of the community in the form of technological software applications to support that activity.

IV. Application of Lucidify to the Design of CLEOS

’The design framework described in Section III has been applied toward the design and implementation of educational technical interventions such as tutorial software, educational collaboratories, and systems that foster organizational learning. This section introduces CLEOS, an implemented learning collaboratory whose design and implementation was based on LUCIDIFY. This section describes the system scope, design goals, and the design process of CLEOS.

A. System Definition

As a testbed for the design framework, CLEOS has been developed. CLEOS is a virtual environment that endeavors to build and nurture a community of learners separated by time and space. As a learning collaboratory, its educational focus is teaching students the theory and practice of experiments in the physical sciences. Specifically, CLEOS supports the teaching of the underlying theory and operational procedures of experiments in NMR Spectroscopy and X-Ray Diffraction. Student learners in the collaboratory will find two tutoring systems: an NMR spectroscopy tutoring system (CVS), and an X-Ray Diffraction tutoring system (VXRD). Instructors in the collaboratory will find a software application to support project-based learning via a project management tool tool to organize and supervise projects (TOSP) that helps instructors create and manage multiple student projects. All collaboratory participants will find an asynchronous communication messaging system (QB) to help experts, instructors, and learners ask and answer questions posed to the community, as well as software applications to support communication and access to information. These software applications are developed with the explicit goal of supporting a wide range of potential users of the system, including practitioners, teachers, students, and researchers. It is a distributed system, so users of CLEOS



Student X logs into CLEOS. After moving to a room with other students, X asks students Y and Z if they are interested in working on their spectroscopy assignment together, which is part of a larger project they are working on together. They agree and X then creates a "Classroom" adjoining their present Room. The all exit the Room and enter the Classroom, where they all start up a Collaborative Virtual Spectrometer (CVS) client, which all automatically register with the CVS-server that was created when the Classroom was created. Now each student, who in reality are separated by thousands of miles, work collaboratively on CVS, where every action taken by one student is broadcast immediately to the other CVSs. At some point, Y decides to look up the Question Board (QB) to see if the question the three students have been arguing about has been answered. Student Y launches the QB and sure enough, the question has been addressed and there are seven responses. When the students have finished with CVS, they assign student Z to report their progress on the project they have been assigned, so Z launches the Tool for Organizing and Supervising Projects (TOSP) and modifies the sub-project "CVS" to reflect that their team has finished the CVS assignment. While using TOSP, Z notices that another team has also finished the CVS assignment, and upon inspecting their report, realizes that the question they had previously argued about had been answered differently by that team. Now unsure of Z's team's answer, Z puts out a call to X and Y, and they reconvene to discuss the question further.

Fig. 2. A use scenario for CLEOS.

are geographically dispersed, making use of local computing resources and the Internet to connect to CLEOS and interact with other nonco-located community members.

B. Design Goals

The overarching goal of the design of CLEOS is to create an effective virtual learning community. Software applications within CLEOS will support mutually beneficial interactions between all members of the learning community, including practitioners, instructors, learners, and researchers. There are three principle types of design goals in the development of CLEOS: educational, methodological, and technical.

From an educational standpoint, the goal of the design is to create a virtual learning community centered around CLEOS. A distributed, collaborative educational learning environment such as CLEOS can provide a wide range of students with increased instructional facilities and learning opportunities that are significantly different than the traditional teacher-centered lecture format. A learning collaboratory, with its emphasis on collaboration among learners engaged in authentically simulated experience, shifts the emphasis of learning from teacher-centered (passive) to learner-centered (active participation by the learner). Fig. 2 describes a scenario that illustrates the scope of CLEOS by describing a typical interaction between community members collaborating on an experiment together.

The challenge is to design distance learning technologies that engender meaningful learning experiences that do not just mimic existing teacher-centered learning practices, but take full advantage of the power of computer-mediated communication to support innovative learner-centered and collaborative interactions between students, teachers, subject experts, and resources. Thus a learning collaboratory can be a useful augment to traditional teaching methods. In addition, the support of learning does not stop with support of student activities. Software applications within CLEOS will also support instructor activities as they develop teaching materials for the learners in the community.

From a methodological viewpoint, the goal of the design of CLEOS, and its constituent components, is to operationalize concepts and techniques within LUCIDIFY, upon which the design is based.

It became clear early in the project that the success of any software developed would depend on the currency of the data within the system. In order to facilitate adoption of the software applications, the developers felt that users efforts in working with the system should be rewarded in kind with benefit to their work [24]. Thus an explicit design goal emerged indicating that

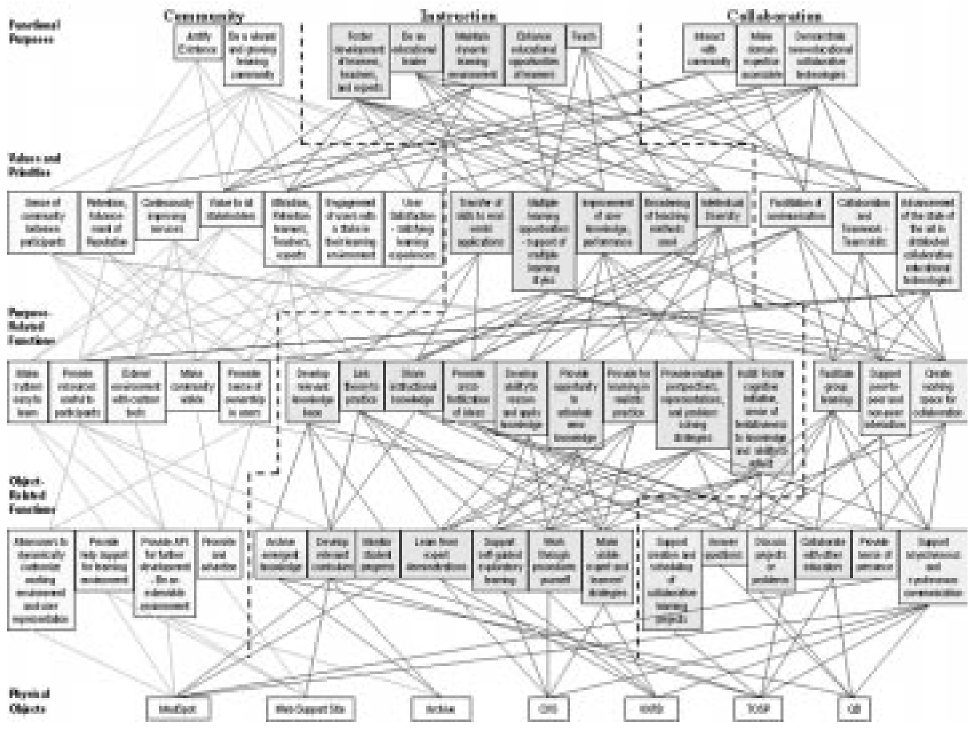


Fig. 3. Initial abstraction hierarchy for CLEOS.

a user who needed to input and maintain information should realize a concomitant benefit in doing so.

Finally, from a technical standpoint, the goal of the design of CLEOS is to build an extensible software system that allows users to adapt their environment in a way that best supports the work on which they are collaborating. In addition, the software components of CLEOS should be extensible from a programmer point of view such that the software infrastructure can grow to meet the needs of a changing user community and expand to provide more functionality to the users as the community matures.

C. Design Process

LUCIDIFY was used to guide the design of the component systems and overall architecture of CLEOS. An initial AH in Fig. 3, describes the pedagogical, collaborative, and social work domain of CLEOS.

There are several stakeholders who goals, values, and activities are represented in the AH. These stakeholders include the some obvious and some not so obvious ones.

- *Instructors*—people who populate the environment with educational material, facilitate learning, and guide a lot of the learner activity,
- *Learners*—users who come to learn about NMR spectroscopy and X-Ray diffraction, and
- *Experts*—people who act as resources to expert knowledge, and who answer questions.
- *Educational Researchers*—CLEOS is after all a testbed for researching learning and collaboration in a distributed computing environment,
- *Software Developers*—who are responsible for designing and implementing the original system, as well as continuously extending the functionality, scope, and capabilities of the system as the community matures, and
- *Hosting Organization*—which must gain enough of sense of success to justify continued hosting of the learning environment.

The functional purposes level of the AH describes the goals shared by all stakeholders of CLEOS. Varied priorities and values of stakeholders are represented at the Value of Priorities level of the AH. It is these values and priorities that the stakeholders use to judge the success or failure of reaching their goals. The Purpose-Related Functions of the AH encapsulate the general functions that will be used to satisfy the Functional Purposes, subject to the values and priorities used as metrics. It is here, in the case of CLEOS, drivers toward specific pedagogical strategies are captured. The Object-Related Functions detail the activities that CLEOS will support in order to realize the Purpose-Related Functions. The bottom-most level, the Physical Objects, of the AH in Fig. 3 is underspecified due to space constraints. The elements to be found in this level should really be the specific features within the applications (objects) that comprise CLEOS. Thus when it is asked how will CLEOS support learning through expert demonstrations (an Object-Related Function), the answer is the "Observe Expert" feature of the tutoring applications of CVS and VXRD.

It is through careful articulation of the AH in Fig. 3 that the features, instructional pedagogies, and collaboration aspects of the elements of CLEOS are designed in a principled way. Gaps in the AH (missing links in either direction) identify gaps in CLEOS's ability to realize the stated functional purposes, and leads the developer to revise and iterate the design. As more is learned, as the community matures, and as CLEOS is expanded, the AH will be modified as well to track the evolution of the work domain.

Abstraction hierarchies are typically characterized by five levels of abstraction. Often, a second dimension, a part–whole decomposition, is modeled. The abstraction hierarchy in Fig. 3 does have a second dimension, but rather than a part-whole decomposition, it is a characterization of three "categories" within the AH. The categories are: community, instruction, and collaboration. In some sense, one could draw three separate AHs, one for each category. However, during the course of the design and development of CLEOS, it became clear just how inter-dependent these three categories were as part of the design. The elements at the abstraction levels within any category are highly linked, yet there are strong links connecting elements of one category to the elements at another abstraction level of another category. Community impacts instruction by providing a learning environment and functions that support learning. Clearly, collaboration has a strong impact on instructional and community functions.

The interweaving of community, instruction, and collaboration directly impacts the design of CLEOS. Each instructional software application (CVS, VXRD, QB, and TOSP) has use models that include stand-alone and collaborative modes. It is important to consider these use-models in the context of a community of learners, and design the interactions between the people, software applications, and environment to support the overall design goals of CLEOS. The next section details the architecture and software components of CLEOS.

V. CLEOS: Description

This section describes the system architecture of CLEOS, based on a multi-user domain (MUD) that facilitates a sense of presence between community members, as well as serving as the communications infrastructure of the learning collaboratory. CLEOS, as a learning collaboratory dedicated to the teaching of NMR spectroscopy and X-Ray diffraction.

A. System Architecture

CLEOS is a virtual learning environment. The user interacts with a virtual "space," navigating rooms and interacting with other users. Within the environment, users will find software tutorial applications that allow for multiple users to operate the same tutorial. Fig. 4 illustrates the system architecture of CLEOS as a set of functional layers.

The topmost layer in Fig. 4 are the users within CLEOS, a diverse collection from students to teachers to experts in the field.

The Application Layer is a set of applications with which the users can interact. These applications are the software tools within CLEOS that support learning. Currently, there are four applications: two virtual instrument tutorials, the CVS and VXRD; an asynchronous messaging system, the QB; and a project-management application, the TOSP.

The User Interface Level is the graphical user interface of CLEOS itself. Users interact directly with the CLEOS Interface to navigate within the virtual environment, to communicate in real-time with other users, and to launch the applications described in the Application Layer.

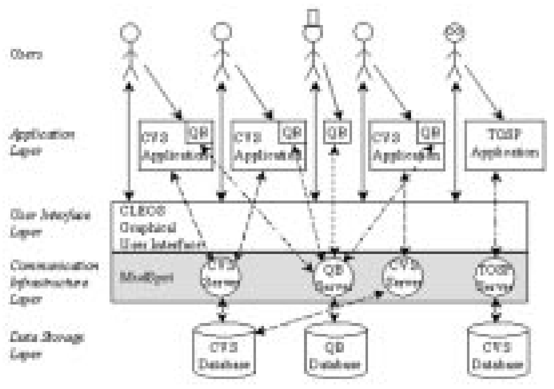


Fig. 4. System architecture of CLEOS.

The Communication Infrastructure Layer is the heart of the CLEOS system. The infrastructure is based on MudSpot, a MUD that is responsible for managing the spatial environment, creating servers to allow multiple users to collaborate via an application, and managing communication between users.

Finally, the Data storage layer is the collection of databases that serve the various applications found in the application layer. The components of CLEOS, and the communication between components (denoted as the arrows in Fig. 4) will be described in more detail in the following sections.

B. Communication Infrastructure Layer

The Communication Infrastructure Layer of CLEOS facilitates the communication between users in a virtual environment where users can collaborate with each other while using synchronously collaborative software applications. The client/server paradigm [20] forms the basis of synchronously collaborative software applications. The heart of the infrastructure for CLEOS is a MUD called MudSpot. This section will briefly introduce MUDs in general, and then discuss MudSpot in particular.

1) Multi-user domains: A MUD program is a shared virtual environment that enables multiple people to interact and communicate synchronously [12], [13]. Usually, such environments are Internet-accessible with text-mediated communication. A user of a MUD controls a character, and he or she defines the characteristics (or description) of that character. To the user, a MUD looks like a series of rooms connected by exits. People and places (rooms) have descriptions, usually in the form of text. It is through these descriptions that the environment takes on a distinctive characteristic, which serves as the backdrop for the interactions between people. Communication is mediated by a series of commands that allow users to speak to each other, or to perform certain actions. These actions range from "saying" something, to "grinning," to navigating between rooms.

An extension of MUDs are MUD object-oriented (MOOs) [12], [54] that allow users to create rooms, exits, things (objects), and verbs. In MOOs, the users are also the programmers of the environment, in a more direct way than with MUDs. MOO users augment and increase the functionality of the virtual space, and thus MOOs are constructed social spaces that undergo a dynamic process of continual evolution [54].

MUDs and MOOs have been created and used for social interaction, play, and increasingly, for education. What is it about MOOs that make them appealing for education? Distance learning situations are characterized by a "class" where teachers and students are isolated and instruction involves simply watching teachers on TV screens. The ability to bring the "class" together where they can interact synchronously allows students to become active rather than passive members of the class. Students can gather in a virtual environment and interact in real time with their teachers and each other. Peer collaboration and an active learning environment can foster learning the subject manner, and also build useful team-building and collaboration skills. Additionally, rather than receiving class material through the mail, the possibility exists to integrate the educational material into the virtual environment. Students can download materials as needed, whenever needed. Finally, virtual environments for learning make it possible for students from a wide array of physical locations to meet with highly qualified teachers [2].

2) MudSpot: MudSpot is an extensible MUD that forms the infrastructure of CLEOS. It is through MudSpot that users communicate and work together on tutorial and other software tools. MudSpot is an extension of Flanagan's JavaMUD [23]. The key enrichment that MudSpot provides its users is the ability to dynamically expand the virtual environment by adding rooms, things, and software applications. It is the ability to run collaborative software applications from within the environment that makes MudSpot a virtual environment for collaborative work. MudSpot provides explicit support for task collaboration through task-defined workrooms and classrooms that mediate the interaction between users [28]. In fact, MudSpot renders a shift of its occupants from *users to collaborators*.

MudSpot, like most MUDs, consists of "people," "places," and "things." Particular to MudSpot is the idea of specific room types. Some of the "things" within MudSpot are collaborative software applications, specifically dedicated servers for collaborative software that allow two or more client applications to synchronize. In Fig. 4, two users each have started running their own versions of the CVS (tutorial) Application. These two applications are both hooked up to a single CVS server within MudSpot (the users are in the same "virtual room" within CLEOS) and so whenever one user modifies something on the CVS Application, the other user's CVS application is immediately updated, via the Server, to reflect the change. Within MudSpot there can be many instantiations of a server, each server dedicated to the group of users collaborating with a software application. In addition to the objects within MudSpot, the user has the ability to perform certain actions. The user can, for instance, speak to others in the room, or can launch client applications (software).

MudSpot is a shared environment that consists of several types of rooms. The type of room defines what sort of activities occur there, and what sort of software applications (generally collaborative) are available for use. In addition to a generic room, there are two specialized rooms currently available in MudSpot. A **workroom** is envisioned as a place where users meet to collaborate on some task. The type of task will most probably depend on the types of software applications that are available in that room. A **classroom** is an area where students work together on some learning activity. Users of MudSpot have the ability to create any of these types of rooms as they wish, as well as creating exits from one room to another. Interestingly, the network of exits between rooms may be such that no two dimensional maps of the virtual space can be drawn.

3) An Extensible Environment: A key feature of MudSpot is its extensibility. There are really two fundamentally different types of extensibility MudSpot has to offer: 1) user extensibility and 2) programmer extensibility.

User extensibility is the type of extensibility afforded by MUD/MOO environments. Users can add rooms at will, naming and describing them as they wish. The richness of the resulting environment is a testament to the personal creativity invested into it by the users as they create their work environment.

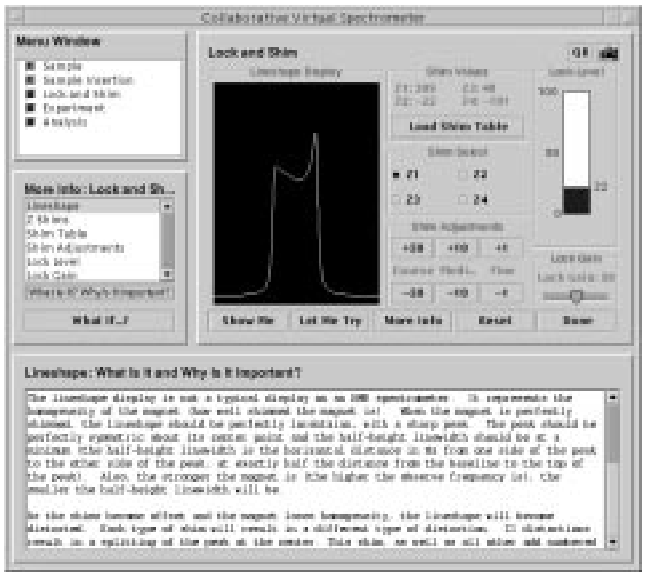
Programmer extensibility is a result of the way MudSpot was designed. It has been programmed in such a way that it is relatively straightforward to customize and extend its capabilities. User commands can be added. Different room types can be added. Most importantly, it is straightforward to add collaborative software applications directly into MudSpot. In addition, there are software hooks to allow external programs to be launched from within MudSpot. CVS, TOSP, VXRD, and the QB are examples of programs written independently from MudSpot, and yet with a minimum of overhead, incorporated into the environment. The applications found in CLEOS is the subject of the next section.

C. Applications Layer

CLEOS is a virtual learning environment containing software applications to enhance learning. Students can find software tutorial applications to teach NMR spectroscopy (CVS) and VXRD. Teachers and students can create and modify project-based learning modules via TOSP. Finally, students can ask and answer questions via the QB.

1) Collaborative Virtual Spectrometer: The CVS [16], illustrated in Fig. 5, represents a computer-based instructional tutor to educate students on the procedures involved in conducting a basic, generalized NMR spectroscopy experiment. As a potential component of a learning collaboratory, CVS was designed with explicit consideration of collaboration. In keeping with the design criteria in LUCIDIFY, several models of synchronous and asynchronous collaboration are supported. The CVS tutorial is designed to be used in three ways: 1) as a stand-alone tutorial system that works one-on-one with a student user; 2) in an asynchronous collaborative effort via the QB;and 3) and as a software application that can be synchronously shared by multiple students working on different machines, where each user's copy of CVS is synchronized with the others in real-time.

The CVS Tutorial was developed under LUCIDIFY and thus its design explicitly support the pedagogical strategies of apprenticeship-style instruction and exploratory learning. Support for active contextualized learning [26] lends itself well for the teaching of procedural knowledge and operational skills [9].

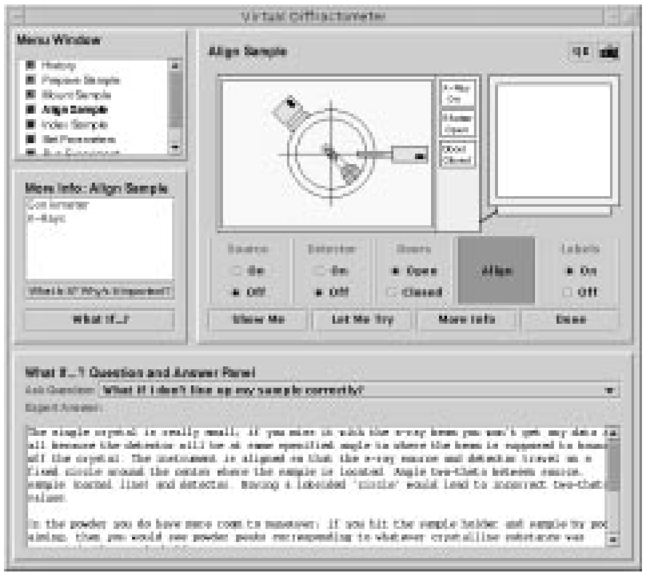


Lineshape: What Is It and Why Is It Important?

The lineshape display is not a typical display on an NMR spectrometer. It represents the homogeneity of the magnet (how well shimmed the magnet is). When the magnet is perfectly shimmed, the lineshape should be perfectly lorentzian, with a sharp peak. The peak should be perfectly symmetric about its center point and the half-height linewidth should be at a minimum (the half-height linewidth is the horizontal distance in Hz from one side of the peak to the other side of the peak, at exactly half the distance from the baseline to the top of the peak). Also, the stronger the magnet is (the higher the observe frequency is), the smaller the half-height linewidth will be.

As the shims become offset and the magnet loses homogeneity, the lineshape will become distorted. Each type of shim will result in a different type of distortion. Z1 distortions result in a splitting of the peak at the center. This shim, as well as all other odd numbered

Fig. 5. Collaborative virtual spectrometer.



What If...? Question and Answer Panel

Ask Question: What if I don't line up my sample correctly?

Expert Answer:

The single crystal is really small. If you miss it with the x-ray beam you won't get any data at all because the detector will be at some specified angle to where the beam is supposed to bounce off the crystal. The instrument is aligned so that the x-ray source and detector travel on a fixed circle around the center where the sample is located. Angle two-theta between source, sample (center) [illegible] and detector. Having a lopsided 'circle' would lead to incorrect two-theta values.

In the powder you do have more room to maneuver. If you hit the sample holder and sample by poor aiming, then you would see powder peaks corresponding to whatever crystalline substance was

Fig. 6. Virtual X-Ray Diffractometer.

Vygotskian theories of learning stress that individuals gain skills by engaging in tasks with an "adult or more capable peer" [59]. In general, four overlapping stages of pedagogy can be identified [10]: (1) Modeling, through the observation of expert performances, (2) Coaching, with expert guidance and help, (3) Fading, where expert assistance is gradually withdrawn, and (4) Reflecting, student self-monitoring and reflecting upon past performances. The role of the expert is to provide appropriate "scaffolding" for the student apprentice.

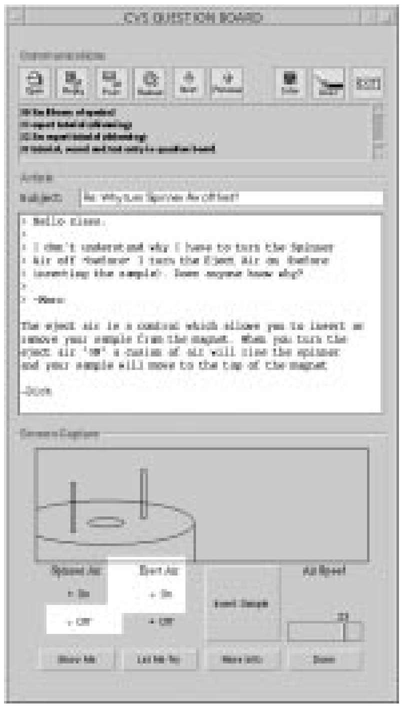
There are two major modes of instruction in CVS: 1) "observe expert" and 2) "act as an apprentice". Additionally, tools for exploration and reflection exist as options to explore related theoretical concepts and ask "what if" questions [31]. The "more info" option present the student with a context-relevant list of theoretical concepts ("What is this and why is it important?") and hypothetical situations ("What if…?"). The "Let Me Try" option provides guidance via a checklist of procedures related to the current context. The "Show Me" option allows students to return to the demonstration mode of "Observe Expert" of the particular lesson. Fig. 5 illustrates the CVS in "Act as Apprentice" mode.

The typical user of the CVS tutorial would be an undergraduate college student with little or no expertise in NMR spectroscopy, but with a grasp of basic science, chemistry and physics. It is the goal of the software to guide the development of expertise in both the theory of NMR spectroscopy and the operational procedures involved in conducting a spectroscopy experiment. The goal of CVS is to raise the user's level of expertise to the point where he or she is competent enough to run an experiment on actual machinery (which is expensive and therefore requires that only trained experimenters are allowed to use it). The design of CVS is based on the Collaborative Apprenticeship Learning Object Toolkit, a library of Java objects for the building of collaborative tutorial software. A detailed discussion of CALOT and the design of CVS is beyond the scope of this paper, and the reader is referred to [18] and [17] respectively.

2) Virtual X-Ray Diffractometer: The VXRD, illustrated in Fig. 6, represents a computer-based instructional tutor to educate students on the theory and procedures involved in conducting a basic generalized X-ray diffraction experiment. VXRD, like CVS, is a CALOT-based system [18] designed under guidance of LUCIDIFY, and so it supports two modes of instruction, "Observe Expert" and "Act as an Apprentice." VXRD can be used in the same ways that CVS is used.

3) The Question Board: One of the key goals of LUCIDIFY is to support asynchronous collaboration to enable collaborative learning. Thus a central goal in the development of the CVS and the VXRD is to use its web-based technology to create mechanisms for collaborative learning software applications. The QB, illustrated in Fig. 7, was developed in order to provide a forum for asynchronous communication between students. The QB is a messaging system that allows students to post messages to a central server, which then makes all messages available for general inspection. Students can read other student's messages, and reply if they wish. Through this messaging system, students can communicate asynchronously as they hold discussions on topics of interest, ask and answer questions, and generally inform and learn from each other. In this way peer-to-peer learning is enabled. The QB d takes this concept one step further by allowing the students to attach to their message a snapshot of the CALOT-based tutorial system under operation. Students can highlight, with transparent colored boxes, portions of the snapshot in order to draw attention to salient features germane to their question or comment. Thus, students have a common object which they can annotate and share in the course of a discussion.

The QB is designed to be used in two ways: 1) in an asynchronous collaborative effort during the operation of a CALOT-based tutorial, and 2) and as a tool to inspect an archive of subject- specific communication within CLEOS. The QB acts as a simple mechanism to help students contact each other and communicate about their common task: working through the tutorial. This form of loose collaboration facilitates peer-to-peer



```
> Hello class,
>
> I don't understand why I have to turn the Spinner
> Air off *before* I turn the Eject Air on *before*
> inserting the sample? Does anyone know why?
>
> -Mers

The eject air is a control which allows you to insert or
remove your sample from the magnet. When you turn the
eject air 'ON' a cushion of air will lift the spinner
and your sample will move to the top of the magnet.

-Dick
```

Fig. 7. Question Board.

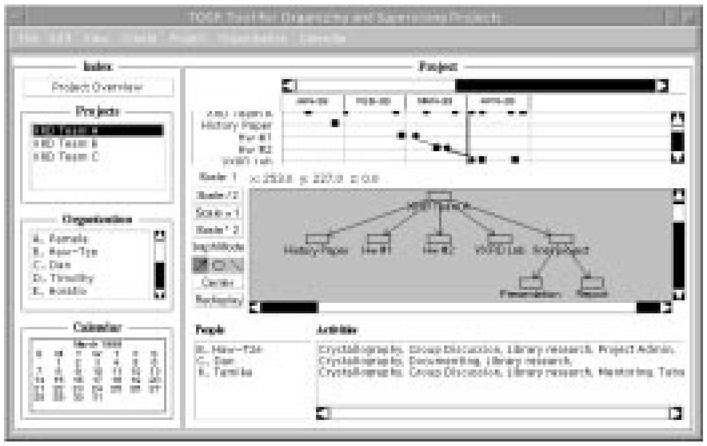


Fig. 8. Tool for Organizing and Supervising Projects.

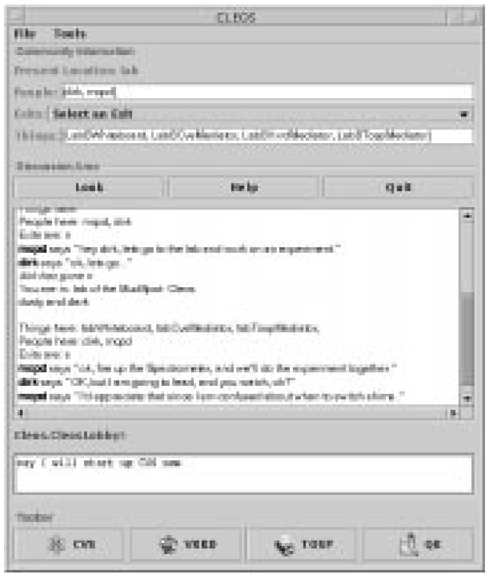


Fig. 9. CLEOS graphical user interface.

learning while still placing most of the responsibility of working through the tutorial on the individual student.

In the context of CLEOS, the QB can also be used as a separate program. All the various forms of the QB (embedded in tutorials like CVS or VXRD, and stand-alone versions) are served through the same database, and so all comments and questions are inspectable. Thus the QB provides some initial simple mechanisms for asynchronous collaborative learning, including an "organizational memory" of comments from previous students, educators, and others, such as bulletin boards and organized annotations.

4) Tool for Organizing and Supervising Projects: TOSP, illustrated in Fig. 8, is a project management and organizational information software application built upon the SPOT software libraries [27]. The SPOT libraries model the components of a research organization as a series of objects, e.g., projects, people, events, skills, activities, roles, deliverables. TOSP builds upon these descriptions an environment where these objects can be visualized, manipulated, and edited. The stand-alone version of TOSP allows the user to work with the data in the database. TOSP can also be used as a distributed, synchronously collaborative tool. Additionally, different sets of users can use TOSP: instructors can collaborate while designing a project, student teams can collaborate as they work through a project, and others can inspect the activities and progress of projects.

D. User Interface Layer

The graphical user interface (GUI), illustrated in Fig. 9, is organized in such a way as to reflect the multiple perspectives and roles a user plays within the CLEOS community, as identified in the abstraction hierarchy in Fig. 3. The GUI presents three windows: 1) Community Information, 2) Discussion Area, and 3) Toolbar.

The **community information** window depicts information about the people, places, and things in the user's present location. The name of the current room, and a list of the people in the room, is displayed. Exits, contained in a pull down menu, are labeled with the direction and destination. Selecting an exit will cause the user to move through it to another room. Finally, a list of the things in the room is given. In Fig. 9, the user is in a classroom where servers for a whiteboard, CVS and TOSP are running. People in that room will most likely be collaborating on one or several of the software applications in the room.

The **discussion area** is where users in a room can communicate to each other. At the bottom of the area, users input their commands. Directly above the input field is the instruction area where, in red, the system prompts the user for specific input by displaying instructions. Finally, the large text area displays the running conversation. System (informational) messages are displayed in italics, user names (when speaking) are displayed in bold.

The **Toolbar** contains buttons for each software application that is available in CLEOS. Which software application can be used collaboratively depends on which servers(s) are running

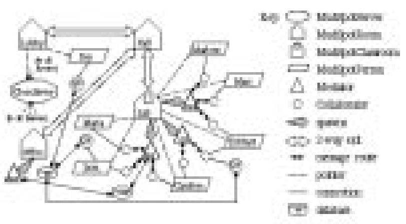


Fig. 10. CLEOS, object-centered perspective.

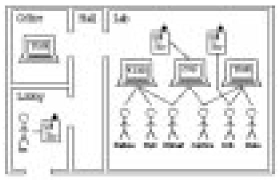


Fig. 11. CLEOS from a user-centered perspective.

in the room in which the user is currently located. The QB is executable anywhere, since it is simply an external window into the message database. The two tutorial software applications, CVS and VXRD, have QBs incorporated within them as well, and when launched from a tutorial, have the ability to take screen captures of the tutorial interface.

The next section discusses the implementation details of CLEOS, discussed from two stakeholder perspectives (user and developer), and how CLEOS is designed to address issues of community, collaboration, and instruction.

VI. CLEOS: Implementation

A learning collaboratory is by definition a collaborative, distributed application. Addressing distributed computing issues such as system partitioning, persistence, multithreading, coordination, communication protocol, and security are vital to the success of a learning collaboratory. The design, implementation and evaluation of a learning collaboratory depends on distributed computing issues such as information sharing, maintaining user identities, communication, and performance. Some of these issues are addressed by the choice of programming language, and in this case Java is well suited for collaborative distributed applications. Java supports many types of network communications, contains many classes that support filtering and preprocessing incoming and outgoing message streams, and supports capabilities such as distributed objects, remote connections to database servers, and directory services. Java's distributed-object scheme called remote method invocation (RMI)], provides the basic elements needed to construct a distributed application.

A. CLEOS Architecture

CLEOS as a software system is the syntheses of all the systems described in Section IV. There are two perspectives from which one can look at the architecture of CLEOS: 1) object-centric perspective and 2) user-perspective.

The **object-centric perspective**, illustrated in Fig. 10, is the perspective of the collaboratory software developer. CLEOS is composed of MudSpot, CVS, VXRD, TOSP, QB, and more.

MudSpot provides four key types of software objects: MudSpotServer, MudSpotRoom, MudSpotClassroom, and MudSpotPerson. The MudSpotServer keeps track of all the rooms and all the people. Each user in CLEOS is represented by a MudSpotPerson. Each place in CLEOS is either a MudSpotRoom or a MudSpotClassroom. A MudSpotClassroom has in it three servers (for applications CVS, TOSP, and VXRD), each of which were spawned when the MudSpotClassroom was created. These servers service any client applications that have been launched by users within that room. Each room keeps pointers to all the MudSpotPerson(s) in it, and exits lead from one room to another. It is by travelling through exits that users can move from one room to another.

The **user-perspective**, illustrated in Fig. 11, is the perspective of the user of the CLEOS system. What the users "see" is a series of interconnected rooms. The user may encounter other people in a room, and can chat with them directly. When in a room with the appropriate server running, the user is able to launch a collaborative software application that will synchronize with others using that same application in this room. Users can access the QB from any room. Additionally, the QB can be launched from within a tutorial software application, and then also has the ability to render a screen capture of the Lesson Window within that tutorial.

B. Addressing the Issues in Distributed Computing

System partitioning is the issue of how to distribute the system among the available computing resources [20]. CLEOS, as a highly distributable system, contains Objects within MudSpot (e.g., MudSpotServer, MudSpotRooms, client applications, servers) that need not run on the same computers. CLEOS does not fit into the traditional server/client paradigm since some objects act as both a server and a client. The system has been successfully operated using a group of computers in Minnesota and Illinois simultaneously.

Persistence is the ability to save changes to data from one session to another [20]. User initiated changes to the CLEOS environment (e.g., the addition of a MudSpotClassroom) can be saved for future sessions. The TOSP and QB databases are updated with each change the user makes.

Multithreading refers to the ability to run multiple threads of control in the same process on the same machine. Each application, each MudSpotRoom, each server application, each client application, runs as its own process, lessening the need for multiple threads within the process. Some software applications, like the CLEOS interface, and the QB when used in conjunction with a tutorial system, run in their own threads.

Multiprocessing is the use of multiple computers (processors) in a distributed application. CLEOS is inherently distributed since multiple, nonco-located community members each interact with the community using their own computing resources.

Coordination is the ability of users and their associated software applications in a system to interact to reach their goals. How much shared state information to invest with each user, versus how much communication between users, are some of the dimensions in coordination. CLEOS allows for synchronous communication between users in the same room as they "talk" to each other, and thus users can work out their own ways of coordinating their activities. It would be inappropriate, given the use-model that drives CLEOS, for the system design to *a priori* enforce certain modes of coordination between users.

Communication protocol is the way various system elements communicate with each other [20]. CLEOS, being an amalgam of several different systems, has many protocols with which to communicate (e.g., HTTP, Java Serialization, Java RMI).

Security is always vital to the health of a collaborative distributed application. The CLEOS application itself, being a prototype, has no security built into it, but has "hooks" where security checks can be implemented.

C. Addressing Issues of Collaboration

Communication needs of users are serviced in both synchronous and asynchronous ways. Direct synchronous communication occurs as users "talk" to each other within a MudSpotRoom. Asynchronous task-focused discussion is supported through the question and answer abilities of the QB. Synchronous collaboration on tutorial software applications is possible via the server/client paradigm that forms the basis of the tutorial systems found in CLEOS.

User identities are maintained via the presence associated with users and things within CLEOS. CLEOS keeps an updated list of all the people, places, and things in a MudSpotRoom. Each user is represented by a MudSpotPerson, complete with name and description. All discussions are tagged with the user's name as they contribute to the conversation. These description and naming features are meant to help users keep oriented with the people, places, and things within their environment.

State information is somewhat dynamic in a system like CLEOS. Conversations and system messages appear in a text area, so users can follow conversations and activities. All users in a room share the state information of that room. In the QB, new questions are immediately posted to the database, so all information is current. In TOSP, with its central database, all information should be concurrent in all the TOSP applications running.

Performance is always an issue in a system with multiple users. If the performance is poor, users will be frustrated. As of the writing of this paper, CLEOS runs with few noticeable delays on a system comprising of a Sun Microsystems Sparc20 workstation (in Minnesota) and two Sun Microsystems Ultra 1 workstations (in Illinois).

VII. Conclusion

LUCIDIFY is a framework for the design of a learning collaboratory. The challenge to the developer of such a learning collaboratory is to understand the domain, the user community, how to best support learning in this environment, how to best support user interactions, and how to provide enough value for all users. The design framework LUCIDIFY suggests modeling techniques and design considerations that guide the learning collaboratory developer address the many challenges in the process of creating a virtual learning environment and nurturing a user community. This framework includes consideration of the domain, the user profile, the learning pedagogies employed, and explicit consideration of the type of collaboration technologies required. In addition, this paper has demonstrated the use of LUCIDIFY in the design and implementation of the CLEOS. The design of CLEOS can serve as an example of how collaborative, distributed computing technology can be used to support learning in a distributed environment.

Acknowledgment

The author would like to thank the following people for their contributions to this work: P. Jones, J. Jacobs, B. O'Keefe, M. Twidale, and P. DeLisle. The author would also like to thank C. Hayes, S. Whitlow, D. V. Reising, and the anonymous reviewers for their thorough and extremely helpful comments on the manuscript.

References

[1] P. E. Agre and D. Chapman, "Pengi: An implementation of a theory of activity," in *Proc. AAAI'87*, Seattle, WA, 1987, pp. 268–272.

[2] (1998) Why Use a MOO for Education?. Athena Univ.. [Online]. Available: [Online] Available: http://www.athena.edu/campus/moo.html

[3] J. R. Anderson, *Cognitive Psychology*. New York: Academic, 1979.

[4] ——, *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum, 1993.

[5] A. M. Bisantz and K. J. Vicente, "Making the abstraction hierarchy concrete," *Int. J. Hum.-Comp. Stud.*, vol. 40, no. 1, pp. 83–117, 1994.

[6] B. Bloom, *Taxonomy of Educational Objectives Book 1 Cognitive Domain*. New York: Longmans, Green, 1956.

[7] J. M. Carroll, Ed., *Scenario-Based Design: Envisioning Work and Technology in System Development*. New York: Wiley, 1995.

[8] G. Chin, "Management of boundary objects in a shared information space for a public works organization," Ph.D. dissertation, Dept. Mech. Ind. Eng., Univ. Illinois, Urbana, 1997.

[9] R. W. Chu, C. M. Mitchell, and P. M. Jones, "Using the operator function model and OFMspert as the basis for an intelligent tutoring system: Toward a tutor/aid paradigm for operators of supervisory control systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 1054–1075, July 1995.

[10] A. Collins, J. S. Brown, and S. E. Newman, "Cognitive apprenticeship: Teaching the craft of reading writing, and arithmetic," in *Cognition and Instruction: Issues and Agendas*, L. B. Resnik, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[11] (1997) CoVis Web Pages. Northwestern Univ., Evanston, IL. [Online]. Available: [Online] Available: http://www.covis.nwu.edu/info/covis-info.html

[12] P. Curtis, "MUDDING: Social phenomena in text-based virtual reality," in *Proc. Directions and Implications of Advanced Computing (DIAC-92) Symp.*, Berkeley, CA, May 1992.

[13] MUD's Grow up: Social Virtual Reality in the Real World, P. Curtis and D. A. Nichols. (1993, May 3). [Online]. Available: [Online] Available: ftp://ftp.lambda.moo.mud.org/pub/MOO/papers/MUDsGrowUp.txt

[14] S. J. Derry and S. P. Lajoie, "A middle camp for (un)intelligent instructional computing: An introduction," in *Computers as Cognitive Tools*, S. P. Lajoie and S. J. Derry, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1993, pp. 1–14.

[15] L. Deter, M.S. thesis, Dept. Mech. Ind. Eng., Univ. Illinois, Urbana, 1999.

[16] M. C. Dorneich and P. M. Jones, "The UIUC virtual spectrometer: A java-based collaborative learning environment," *J. Eng. Educ.*, vol. 90, no. 4, pp. 721–728, Oct. 2001.

[17] M. C. Dorneich, "System design framework for a learning collaboratory," Ph.D. thesis, Dept. Mech. Ind. Eng., Univ. Illinois, Urbana, 1999.

[18] M. C. Dorneich and P. M. Jones, "The systematic application of the apprenticeship learning pedagogy to computer tutorial design," *Proc. 2000 IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 8–11, 2000.

[19] H. L. Dreyfus and S. E. Dreyfus, *Mind Over Machine*. New York: New York Free Press, 1986.

[20] J. Farley, *Java Distributed Computing*. Cambridge, MA: O'Reilly, 1998.

[21] C. Fiebig, "Development of Expertise in Complex Domains," M.S. thesis, Dept. Comp. Sci., Univ. Illinois, Urbana, 1997.