ABSTRACT
The design of software for learners must be guided by educational theory. We present a framework for learner-centered design (LCD) that is theoretically motivated by sociocultural and constructivist theories of learning. LCD guides the design of software in order to support the unique needs of learners: growth, diversity, and motivation. To address these needs, we incorporate scaffolding into the context, tasks, tools, and interface of software learning environments. We demonstrate the application of our methodology by presenting two case studies of LCD in practice.


1. INTRODUCTION: MOTIVATION AND GOALS
Education—in schools, in homes, in the workplace—is arguably one of the top growth areas for computing technologies. In designing software for education, we are designing for learners. Learners are also users, so principles of user-centered design apply; however, user-centered design guidelines are not sufficient to address the unique needs of learners:

- **Growth.** At the core of education is the growth of the learner; promoting the development of expertise must be the primary goal of educational software. Rather than just support “doing” tasks, software designed for learners must support “learning while doing.”
- **Diversity.** Developmental differences, cultural differences, and gender differences play a major role in the suitability of materials for learners. To be usable by all learners, a range of software tools which address these differences must be available.
- **Motivation.** In contrast to software developed for professionals, the student’s initial interest and continuing engagement cannot be taken for granted.

To address these unique needs of learners, we are extending the established user-centered design (UCD) framework [21] to a learner-centered design (LCD) framework [33]. Our current focus is on K-16 learners; however, given Senge’s [31] compelling arguments that an organization must be a learning organization in order to be productive, LCD should also have validity for the workplace.

The central claim of LCD is that software can embody learning supports—scaffolding—that can address the learner’s growth, diversity, and motivation. Scaffolding is an educational term which refers to providing support to learners while they engage in activities that are normally out of their reach [37, 39]. For example, Carroll’s [7] “Training Wheels” model suggests that one scaffolding strategy would be to initially hide sophisticated commands from novices.

Our selection of particular scaffolding strategies is directly informed by sociocultural and constructivist theories of learning. An expanded discussion of this theoretical framework and the practices that are informed by the framework is given in Section 2.1. A brief example here will convey the basic idea:

- **Theory.** One tenet of constructivist theory is that learning is a process of actively building mental representations.
- **Practice.** An implication of this theory for software-realized scaffolding is to design software which supports learners in visualizing the process of doing the task (e.g., writing a program). For example, Guzdial’s Emile [14], a computer-based environment for learning HyperCard programming, supports students in building an explicit plan before actually writing code.

Other examples of systems informed by this constructivist framework are Resnick’s StarLogo [27], Fischer’s software-based critics [12], and Schank’s active learning environments [30]. In contrast, Anderson’s work on intelligent tutoring systems [2] is informed by an information processing framework.

The push for educational reform in the U.S. is strong. Currently, the dominant educational paradigm is “didactic instruction,” where learning is viewed as an information transmission process: teachers have the information, students don’t, and teachers’ lectures serve to move information into the heads of students. In contrast, national and state education reform movements are advocating that
some combination of sociocultural and constructivist paradigms be implemented, at least in K-12. For example, Project 2061, a national science curriculum developed by the American Association for the Advancement of Science [1] calls for students to engage in long-term, authentic investigations. In such an education model, the student’s role is to actively build mental representations and assimilate professional practices, while the teacher’s role becomes one of mentor and manager as opposed to information deliverer [e.g., 4, 20].

Sociocultural and constructivist theories therefore inform an authentic, project-based learning environment—and in such an environment, the need for software-realized scaffolding is paramount. (This is discussed more fully in Section 2.) For example, the tasks students undertake are more complex, and often more diverse. Software-realized scaffolding can reduce the complexity of these tasks by relating discrete subtasks to their current mental representations.

In this paper, then, our goals are as follows:

- **Articulate the Theoretical Rationale.** Our perspective, LCD, is based on the learning theories of the sociocultural and constructivist genres (described in Section 2.1).

- **Articulate the LCD Framework.** We outline the main principles of LCD (Section 2.2).

- **Illustrate LCD via Case Studies.** In Section 3, we provide several case studies that exemplify how LCD has informed the design of software that is being used by both high school students and college undergraduates.

We conclude by summarizing the key issues in LCD.

2. LEARNING THEORY IN PRACTICE

In this section, we highlight the key notions in the sociocultural and constructivist paradigms that underlie LCD and the key notions in our perspective of LCD. In preparation for that discussion, we need to first provide a characterization of a learning environment. In particular, there are four elements that must be addressed in constructing an effective learning environment:

- **Context:** What is the environment in which the software will be embedded? How will it be used, and by whom?
- **Tasks:** What are the tasks the software will support?
- **Tools:** What tools will perform these tasks?
- **Interface:** What is the interface to those tools?

We propose to develop theoretically-motivated scaffolding strategies that focus on each of these elements.

### 2.1 Theoretical Rationale for LCD

One goal of formulating and applying LCD principles is to realize current theories of learning in software that can shape and augment curricula in ways that have been shown to be beneficial for learning. Therefore, to describe the parameters by which learner-centered software could be designed, this section illustrates what we mean by learning and how learning can be supported. Drawing upon two theoretical perspectives, constructivism and socioculturalism, we define a framework for the discussion of context, tasks, tools, and interface to support the needs of learners. We chose these two perspectives because together they provide a wealth of principles and evidence for effective learning never before realized in the history of educational theory.

From the constructivist paradigm, we draw upon research on cognition that reveals a richness to the capabilities of the human brain that previous theories of learning did not reveal. From the sociocultural paradigm, we draw upon research on activity and practice that shows that knowledge, learning, and understanding are contextualized, situated, and culturally-based. These two perspectives have informed recent developments in curricula and pedagogy; for example, project-based science [4], Community of Learners [5], and cognitive apprenticeship [9]. These paradigms have also informed recent work in the use of technology in educational settings, for example, CSILE [29], the anchored instruction work at Vanderbilt [8], the work at Northwestern (formerly at Bank Street) [15, 23], the research at Apple’s Learning Technologies Group [17], and Papert’s work on constructionism [22]. Since these approaches have been shown to be quite effective, the principles for LCD software are derived from the same body of evidence. Furthermore, we claim that the use of LCD software in these settings will prove to be quite advantageous (one goal of this paper is to illustrate this claim).

Table 1 shows how the constructivist and sociocultural perspectives define the learning process, and provides a few examples of how this process might be realized through scaffolding.

<table>
<thead>
<tr>
<th>Constructivism</th>
<th>Socioculturalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning is facilitated in a multitude of ways through the scaffolding process [37, 39]. Scaffolding is support that enables a learner to accomplish a goal that would not be attainable without the support. However, what counts as successful and appropriate scaffolding depends on how one defines learning. Table 1 shows that in the constructivist perspective, the goal is to help learners build knowledge representations and mental skills like those of experts. The table also shows that the goal in the sociocultural perspective is to help learners socially construct knowledge through the use of language, tools, and practice in a community. These two ideas lead to very different, yet useful conceptions of how scaffolding (also shown in the Table 1) can be provided through the context, tasks, tools, and interface.</td>
<td></td>
</tr>
<tr>
<td><strong>Constructivism:</strong> According to the constructivist perspective, learning is assimilation, augmentation, and self-reorganization of the incomplete mental structures held by learners [25, 26, 36, 37]. The goal is to enable knowledge organization, depth, and structure that corresponds to that of an expert in the domain under study.</td>
<td><strong>Socioculturalism:</strong> From the sociocultural paradigm, we draw upon research on activity and practice that shows that knowledge, learning, and understanding are contextualized, situated, and culturally-based. These two perspectives have informed recent developments in curricula and pedagogy; for example, project-based science [4], Community of Learners [5], and cognitive apprenticeship [9]. These paradigms have also informed recent work in the use of technology in educational settings, for example, CSILE [29], the anchored instruction work at Vanderbilt [8], the work at Northwestern (formerly at Bank Street) [15, 23], the research at Apple’s Learning Technologies Group [17], and Papert’s work on constructionism [22]. Since these approaches have been shown to be quite effective, the principles for LCD software are derived from the same body of evidence. Furthermore, we claim that the use of LCD software in these settings will prove to be quite advantageous (one goal of this paper is to illustrate this claim).</td>
</tr>
</tbody>
</table>
tasks and practices require the use of tools, such as information management, organization, and communication tools, that break the content and task into sub-parts that relate to currently held mental structures of the student. (This concept is listed in the Examples of Scaffolding row of Table 1.) The tools must support the student in moving toward expert-like knowledge organizations, depth, and structure. Finally, use of these tools (and thus the move toward expert-like knowledge) can be scaffolded by providing visual or conceptual representations in the interface that correspond to those held by the student. Furthermore, the representations help relate the student’s knowledge to more expert-like representations.

Furthermore, in the constructivist perspective, learning is growth in the abilities to self-monitor acquisition, management, and use of information and knowledge. Therefore, scaffolding can be provided in the construction task by providing processes that the student is able to undertake, and by also explicitly encouraging other, more expert-like mental processes used in the task. This explicitness, noted in Table 1, enables the student to assimilate the expert-like mental skills and learn how to consciously use them.

**Socioculturalism:** According to the sociocultural perspective, learning is enculturation, the process by which learners become collaborative meaning-makers among a group defined by common practices, language, use of tools, values, beliefs, and so on [6, 19, 28, 38]. The goal is to enable practices and meaning making that are appropriate in the professional culture of the domain under study.

Enculturation is enabled by creating an authentic context through which students participate in the tasks and practices of the professional culture. Participation in these tasks and practices require the use of tools, language, social interaction, communication, and so on which, according to Table 1, contain constraints and affordances that scaffold the execution of the tasks and practices and help move the learner toward culturally appropriate, and thus meaningful professional practices. Finally, use of the tools (and the move to professional practices), can be scaffolded, for example, by providing symbol systems in the interface that have meaning in the learner’s culture, but also help move the learner toward the use of symbol systems used in the professional culture.

Through LCD software, we create and sustain a learning context in which authentic, personally relevant, project-based, constructivist activities are carried out. This encourages understanding and growth in the constructivist sense that students build knowledge structures and mental

<table>
<thead>
<tr>
<th>What is learning?</th>
<th>Constructivism</th>
<th>Socioculturalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning is assimilation, augmentation, and self-reorganization of the incomplete mental structures held by learners.</td>
<td>Learning is enculturation, the process by which learners become collaborative meaning-makers among a group defined by common practices, language, use of tools, values, beliefs, and so on.</td>
<td></td>
</tr>
<tr>
<td>Learning is growth in the abilities to self-monitor acquisition, management, and use of information and knowledge.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How is scaffolding provided?</th>
<th>Constructivism</th>
<th>Socioculturalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaffolding is provided by teacher-student, student-student, and student-tool/material interactions.</td>
<td>Scaffolding is provided by the contextual constraints and affordances that support understanding, action, and meaning making through language use, practices, tools, social interactions, and so on of a culture.</td>
<td></td>
</tr>
<tr>
<td>Reduce task complexity by structuring the task into discrete steps (that relate to the student’s mental representations) to reduce cognitive overhead, which will enable assimilation and accommodation of the new information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relate (positively or negatively) new information to currently held knowledge structures through examples, analogies, and multiple representations that help move the learner to more expert-like representations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motivate the student to become actively engaged by making the information or task relevant, personal, meaningful, and approachable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model metacognitive practices and encourage articulation and reflection, being explicit about the cognitive benefits of these activities.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples of Scaffolding</th>
<th>Constructivism</th>
<th>Socioculturalism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce task complexity by enabling authentic activities which provide tools and contexts that constrain the process of accomplishing the task.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support different learners by enabling participation in co-construction of meaning and meaningful artifacts within the culture.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide translation from use of the learner’s cultural symbol system to that of the professional culture.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enable the socially constructed development of value systems that support activity, engagement, and meaning making in the culture.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Learning as Defined by Constructivism and Socioculturalism
skills as they engage in activities, and also in the sociocultural sense that students learn and partake in the understandings developed during professional, authentic activities rich with scaffolding cues. The use of technology is not paramount to curricula that draw upon the constructivist and sociocultural paradigms; nonetheless, we argue that properly designed software—LCD software—can help foster the success of teaching and learning in contexts that substantiate these theories.

2.2 Learner-Centered Design: Connecting Theory to Needs to Practice
At the outset of this paper, we identified three needs of learners: growth, diversity, and motivation. In a fully-articulated LCD framework, each of these three needs would be addressed by theoretically-motivated scaffolding strategies in each element of a learning environment. At this point in our work, we can only partially complete this picture. Thus, Table 2 describes our current rationale for the links between theoretical justifications, learning environment element, learner need, scaffolding strategy, and implementation.

Our research strategy has been to focus on what we feel are the most salient linkages; we indicate that focus in Table 2 by emphasizing the salient learner’s need. In the Task element, the scaffolding strategies that we have identified, rationalized, and implemented, focus on supporting growth of understanding in the learner. In the Tools element, they focus on the diversity of learning styles and levels of expertise. And in the Interface element, they focus on motivation.

To ground these rather abstract notions, in the next section we present two case studies of how we implemented the scaffolding strategies in software, and what impact that software has had on learners. Table 2 summarizes the connections between theory, needs, and practice.

3. LEARNER-CENTERED DESIGN: TWO CASE STUDIES
Model-It and NoRIS are learner-centered software tools which we have designed for two different contexts:

- **Model-It: High school, project-based science classroom:** We are working with science teachers at Community High School in Ann Arbor to develop a new high school science curriculum in which computing technologies are routinely used, and in which the subject matter of earth science, chemistry, and biology is combined and taught in the context of meaningful, long term projects. Model-It, software for building and testing computational models, is one of the tools we are developing for use in this environment.

- **NoRIS: University nuclear engineering classroom:** The University of Michigan Nuclear Engineering department encourages the use of computational science in the upper-level undergraduate curriculum. NoRIS is a problem-solving environment we have developed for use in these classrooms.

While on the surface these two contexts are different, at their core they both require the same sorts of scaffolding; the only real difference is one of emphasis. In the high-school context, motivation is a big issue, while it is less so in the undergraduate context. However, in the undergraduate context, structuring the complex tasks that make up a computational science-style argument is the real challenge.

In our discussion of each example, we first present the software design, and how it incorporates scaffolding to address learner’s needs regarding software context, tasks, tools, and interface, as summarized in Table 2. Then, we present examples from the user testing data which illustrate the impact of specific software features designed to provide scaffolding.

3.1 Case Study One: Model-It
Model-It is designed to support learners in building and testing models of dynamic systems. Scientists build models to test theories and to develop a better understanding of complex systems [18]. Similarly, we want to support students in the building of models, as sociocultural learning theory says that learners should be involved in professional practices. Constructivist learning theory predicts that by constructing external representations of scientific phenomena, learners are building an internal, mental model of the phenomena. We believe that by building models, students will support, refine, and develop their understanding of a scientific system by constructing models to represent their understanding of the phenomenon and its complex interrelationships.

The modeling tools that have typically been designed for students fall into two categories: pre-defined simulations, and modeling environments. Pre-defined simulations, such as Maxis’ SimEarth and Wings for Learning’s Explorer, are not constructivist; although user-friendly and informative within their pre-programmed domains, they do not provide access to underlying functions and representations which drive the simulation, nor the ability to add or change functionality. On the other hand, modeling environments, such as High Performance System’s Stella or Knowledge Revolution’s Working Model, allow unlimited flexibility in building models. However, they are difficult to learn because they don’t support the novice’s knowledge representation of the domain; for these tools, building complex models requires mastery of a complex authoring language [35]. Thus, current modeling tools inadequately address the needs of learners.

3.1.1 Design and LCD Scaffolding
**Context:** Model-It, with its emphasis on building and testing models, is designed to be used in an authentic, project-based science curriculum, grounded in constructivist and sociocultural educational paradigms. The 9th and 10th grade students in our pilot studies have been engaged in a
long-term project investigating the question “How safe is our water?” Specifically, they are studying a tributary of the Huron River which flows near the school, collecting a variety of data to determine the quality of the water. Since this water eventually ends up in their drinking fountains, the question is motivating and personally meaningful to the students.

Using Model-It, the students constructed models of the stream ecosystem, and were assigned open-ended projects in which they were asked to build models to represent their choice of particular stream phenomena, e.g., land use practices: the impact of man-made structures such a golf course or parking lot on stream quality. Creating models is motivating to students because the students are engaged and challenged to create an original artifact. Furthermore, as students have more input into the choice and control of their environments, their motivation for pursuing cognitively challenging problems increases [4]. Allowing students to decide how to plan, design, and work on their models can engage them in the learning process.

**Tasks:** Model-It scaffolds the complexity of the modeling task by providing a set of pre-defined high-level objects (e.g., stream, bugs, golf course). These physical objects match the learner’s knowledge representation of the domain, in contrast to an expert’s knowledge representation which might consist of domain-independent primitives of inputs, outputs, functions and states.

Students select from this set of objects, define factors of the objects, and relationships between the factors. Model-It redefines the task of defining relationships by supporting a qualitative representation of relationships, rather than requiring formal mathematical expressions. This scaffolding is important for learners because their knowledge structures don’t initially include a quantitative command of the concepts involved.

**Tools:** Learners need tools appropriate for their learning styles and levels of expertise; therefore Model-It provides tools for both qualitative and quantitative definition of relationships. Initially, relationships can be defined qualitatively by selecting descriptors in a sentence, e.g., “As stream phosphate increases, stream quality decreases by less and less” (Figure 1). As students’ knowledge representations of the domain become more expert-like, they have the option of defining the relationship more quantitatively, e.g., by entering data points into a table (Figure 2). Model-It also supports a similar qualitative definition of rate relationships which define how one factor sets the rate of change of another factor over time.

To support different learning styles, and to facilitate the learner’s shift to more abstract mental representations, these tools provide both textual and graphical representations of relationships. Given a qualitative definition, the software translates the text into a quantitative visual representation; e.g., “decreases by less and less” is interpreted as shown by the graph in Figure 1.

**Interface:** Learners often need extra motivation to sustain interest in a task, and the interactivity and engaging personal graphics of Model-It can help provide that motivation. To make the task more concrete and authentic, objects are represented with actual digitized photographs and user-defined graphics. Students can create their own objects and paste in their own pictures. In Figure 3, the background graphic is a photograph of the actual stream the students studied. According to sociocultural perspectives of learning, this personalized representation creates a context through which the activity has meaning.

The Factor Map (Figure 4) provides an interactive overview of the model. It helps students structure the task by providing a means of visualizing the network of factors and relationships, rearranging the nodes in a meaningful way, and making changes (e.g., drawing an arrow to create a new relationship).

---

1 Model-It can be used to build a wide range of process flow models; for our preliminary classroom study we chose the domain of stream ecosystems. In our description of the program, we use examples from this domain.
were asked to create their own models to represent some chosen ecological phenomenon. In a recent paper [16], we present a detailed analysis of the data. The following discussion focuses on a representative pair of students, Paul and Jim, two 9th graders from our first classroom study, and how Model-It scaffolded them in creating a complex model in just one 45-minute period.

**Context:** The open-ended modeling task assigned to the students gave them the flexibility to branch off and explore different topics, and to express their own understanding. For example, to demonstrate land use impacts, Paul and Jim chose to put the golf course object into their model, and show how factors of the golf course might affect the stream and the organisms living in it:

J: Let's use that one.
P: The golf course?
J: Yeah, we haven't used that one yet.
P: How the golf course affects what, though?
J: How the golf course affects, um, bacteria.
P: Too hard.
J: It's easy. Because the golf course, a lot of geese are on the golf course, and the geese feces go in the water.
P: Oh, and it affects fecal coliform
J: Which in turn affects the bacteria, and the fecal coliform grows on bacteria.
P: Okay, where do you want the golf course?
J: Right there.

This opportunity to build their own models was extremely motivating for Paul and Jim; they displayed excitement and enthusiasm for the project throughout the class period. Once they had completed their initial goal of representing the golf course impact, they branched out on their own to create more relationships, from the stream quality to the mayfly population. They expressed pride in their model, and called the teacher over to show it off to her. This reaction was typical of the entire class; as one student said in post-interviews, “It makes you think more about a real-life situation, where there's no real answer—you set it up and everything.”

**Tasks:** Students were comfortable expressing themselves qualitatively, and using the qualitative definition of relationships, they were able to build complex relationships very quickly:

P: As geese increases fecal coliform increases at about the same. And then if we want, it won't take long to put in nitrates.
J: Okay.
P: We can add that in.
J: Cause that's part of fertilizer...
P: Cause that's part of fertilizer, yeah. So we go to stream...okay...to nitrates N I T nitrates.
J: Lesser and lesser.

Paul and Jim created four accurate, interrelated relationships in four minutes, and in the next four minutes, tested and verified their model, and found another relationship to add (they realized that the size of the golf course should affect...
the number of geese on it). Figure 4, above, shows the factor map of their final model. In class discussion, they proudly described how their model worked:

P: The size of the golf course affected the geese, the number of geese...
J: The more land there is the more geese... And the more geese the more fecal coliform.
P: The golf course size affected nitrates and phosphates...because the bigger golf course has more fertilizer and fertilizer has nitrates and phosphates in it.
Teacher: Do you have any [relationships] going to quality?
P: Well I’m getting there, okay? This is complicated! Okay, fecal coliform goes to quality, phosphate goes to quality, nitrate goes to quality... And then the quality went to rate of growth.
Teacher: Why?
P: Because the better quality...
J: There is the more mayflies can grow. And then the growth went to count and the decay went to the count.

Tools: Providing a variety of modeling and visualization tools proved very useful for learning, as students could choose the tool which made the most sense to them. For example, we provide both qualitative and quantitative means of defining relationships to support students at different levels of expertise. While Paul and Jim exclusively used the qualitative “text view” tool, another classmate preferred the precision afforded by the quantitative “table view.” Often, students transitioned from one to the other during our longer studies, switching to the "table view" when they realized a need to make their models more accurate.

Interface: Meters and graphs provided visualization of simulations as they ran, and were used for model testing and verification. For instance, during their testing, Paul and Jim used the meters to try different values of golf course size, and realized that it should affect the number of geese on the golf course, so they went back to put that relationship in: “So, golf course size affects golf course geese. Yeah, we can do it. As golf course size increases, geese increases by about the same.”

3.1.3 Summary
Our Model-It testing showed that the software design scaffolded the learners' growth, diversity, and motivation. Within the context of this project-based classroom, working on an authentic problem, students were able to build and test computational models, a task which is usually inaccessible to learners in high school science classrooms. Students used modeling tools provided by the software in ways reflective of their learning styles; their engagement with the modeling task was evident in their interaction with the interface as they built and tested their models.

3.2 Case Study Two: NoRIS
NoRIS is designed to provide an environment that will enable students to use professional computational science tools to carry out a scientific investigation. More and more researchers are turning to computational science when they investigate problems because increased computing power allows them to model physical phenomena, giving more explanatory power to their arguments. Therefore, it is important for students to use authentic tools as they learn to conduct investigations and construct scientific arguments [10]. However, learning to use computational science tools and techniques is a complex process that poses difficulties to the learner.

First, there are many different individual computational tools available to scientists, but few tools that provide comprehensive support for the entire investigative process. For example, visualization packages are very powerful, but very specific for a certain subtask of an investigation. Others, such as Mathematica, Maple, etc., are attempting to integrate more functionality within a single package, but the packages are still complex and do not support all investigative tasks [34], nor do they provide support for learners.

Second, computational science results in artifacts of different media types, but there is no support for the construction of the scientific argument, or for the management of the artifacts necessary to support the investigatory process [13]. For example, in a given situation, a student may need to refer to a source code file, data file, and graph, all of which may reside in different directories. The responsibility for organization and access of these artifacts is with the student.

Finally, students are confronted with a variety of different interfaces and tools, which adds an additional level of complexity to the investigation.

In order to address these shortcomings and provide computer-based support to help students learn the investigative process, we have developed NoRIS (Notebook-based Research and Investigative process Support system).

3.2.1 Design and LCD Scaffolding
Context: NoRIS provides a platform that enables students to use computational science so that they can carry out a scientific investigation. NoRIS is being used in a senior-level nuclear engineering class where students investigate numerical methods. NoRIS assumes more of a sociocultural perspective: by giving these students an environment that reduces many of the complexities inherent in computational science, NoRIS aims to support students as they begin learning the tools and practices of the professional researcher.

Tasks: Students with little expertise can be hindered by having to remember the variety of disjointed, lower-level tasks that make up an investigation. NoRIS therefore restructures an investigation in terms of high-level tasks:

- Notekeeping: Students continually record important observations, data, etc. throughout an investigation.
• Building cases: A case encompasses the major tasks that use computational tools, such as writing numerical-method programs, visualizing data, etc.

By providing support and structure for these high-level tasks, NoRIS allows the student to begin constructing an understanding of the investigative activities that researchers perform.

As well as restructuring the investigative process, NoRIS also reduces complexity by handling the student’s managerial tasks, such as artifact management. Artifact management is important because throughout an investigation, the student may have to re-use, modify, or refer to artifacts such as notes, source code, data files, etc. However, it becomes tedious and distracting for a student to coherently organize their artifacts. By supporting artifact management, the student can focus more on their investigation less on mundane, bookkeeping tasks.

**Tools:** In order to provide an environment that students can use for scientific inquiry, NoRIS provides the variety of tools needed by beginning students to complete their tasks. As we have seen, there are many computer tools that can be used in a scientific investigation: computational tools (such as compilers and algebraic/mathematical software), visualization tools, etc. NoRIS provides this functionality by integrating existing software packages.

However, for tasks such as artifact management, there are no existing tools that the student can use. NoRIS is designed as a computer notebook, a metaphor that corresponds to the student’s current mental representations—they know what it is and how to use it. The notebook metaphor provides an organizational structure to help students manage the different artifacts that they have created during the argument. For example, NoRIS includes the Notebook Summary window (Figure 5) that summarizes the different numerical-method programs that have been written by the student.

NoRIS has a simple, consistent interface to all of the different tools, by providing button and menu-based access to each tool. This corresponds to the constructivist principle that complexity is reduced by offering similar processes that match current mental structures for a range of tasks. Examples include:

• The Workspace Tool (Figure 6) provides button and menu-based access to the tools needed by the student to build and analyze numerical method cases.

![Figure 6: The Workspace Tool](image)

• The Multiplot Tool (Figure 7) allows the researcher to easily plot data files in the same graph window for analysis. The student can simply “check off” all of the files that they want to plot and then press the Plot button.

![Figure 7: The Multiplot Tool](image)

Once students can access the necessary functionality, they need further support to help them identify and complete their investigative tasks. To further sustain the students as they proceed through their investigation, NoRIS provides visual cues in the interface so that the students can see the different steps in the process, and the different types of information that they must record. Examples include:

• The Workspace Tool (Figure 6) contains a task diagram (a constructivist concept) of the process used to construct a case for the numerical method that they are investigating. Each button represents a different stage in the case-building process. Pressing a button presents the user with a menu identifying their options for that particular stage.

• Notepad windows (Figure 8) contain button palettes that identify to the student the different types of
information that they should be thinking about and recording throughout the investigation, such as problem objectives, descriptions of the numerical methods they are investigating, etc. This structure encourages students to reflect and keep important notes throughout their investigation.

We also saw that it was useful for NoRIS to handle some of the student’s tasks, such as artifact management. One student noted the advantages of this:

“I am usually disorganized and after a while, I spend a lot of time organizing things—setting up directories, putting codes and things in the right places. NoRIS takes care of this—this really helps because it lets me concentrate on the problem...”

We also saw that automatic artifact management helped students manage the complexity of re-using and modifying existing artifacts to build new cases. Since it was easy for students to find and re-use old artifacts, students could build new cases from old cases quickly; this encouraged students to experiment with numerical methods by continually modifying a base case for the different experiments.

**Tools:** Students liked the fact that they had all of the necessary tools available to them in one application. Since students are not experts in using computational science, they may not know what tools to use in given situations:

“[NoRIS] really provides an integrated package that beginning students can really use...having all of the needed information ‘at my fingertips’ is an advantage so that I do not have to bounce around different programs...this is good for students who are inexperienced with [computers]”

By having all of the functionality available to them in one application, the students did not have to be moving from application to application, which can be a problem for students who have little expertise with computational science.

**Interface:** Students liked using the interface because it was easy to invoke their different tools. The fact that they could use menus and buttons to access many of their tools made it easy for the student to quickly get started with their investigation.

Students also appreciated the different visual cues provided by the interface. For example, one student commented on the notepad button palette that identifies important pieces of information that should be recorded:

“[The button palette] helps lay out the thought process I should be following when I start working on my problem...Seeing [the buttons] makes me pause and think about the problem rather than just jumping in and starting to write programs, which is what I might normally do.”

The visual cues of the interface therefore structure the task and articulate their thoughts.

**3.3.3 Summary**

Our user testing has shown that students are able to use NoRIS to complete authentic scientific investigations, and that they find the structure provided by the program helpful. Furthermore, by providing an accessible interface to the array of computational tools used by professional
researchers, NoRIS supports learners in their enculturation into professional practices.

4. CONCLUDING REMARKS
Model-It and NoRIS are two components of the ScienceWare suite of tools, a “computational workbench” that we are developing to scaffold learners engaged in the full range of scientific investigatory activities. As we apply our LCD strategies to the design of the ScienceWare tools, and study the use of these tools in classroom settings, our goal is to develop a fully-articulated LCD framework in which the needs of learners are specifically addressed by theoretically-motivated scaffolding for each element of the learning environment.

In putting forth the notion of UCD, Norman, Draper, and their book contributors [21] sought to focus attention on the needs of users at a time when there was growing interest in developing usable and productive interfaces and interaction paradigms. Similarly, in putting forth the notion of LCD, our intent is to focus discussion on work that is expressly intended for learners at a time when, as Business Week [3] declared, there is a “revolution” going on in educational software. UCD has proven itself to be a useful notion; time will tell whether LCD is similarly so.

ACKNOWLEDGMENTS
We would like to extend our great appreciation to the other members of the Highly Interactive Computing (HI-C) research group, the teachers and students at Community High School in Ann Arbor, and Dr. William Martin and the University of Michigan Nuclear Engineering Department, for their feedback and support.

This research has been supported by the National Science Foundation (RED 9353481 and IRI 9117084), the National Physical Science Consortium, and the University of Michigan.

REFERENCES


