Submitted to the ACM Conference on Human-Computer Interaction (CHI)
March 22-27, 1997, Atlanta GA USA

Designing for Multiple User Groups: Lessons Learned from Developing Research Software for the Classroom

Susanne Jul, Jonathan Klein[*]*, Shari L. Jackson, Elliot Soloway*
University of Michigan
1101 Beal Av.
Ann Arbor MI 48109 USA
E-mail: {sjul, phaedra, sjackson, soloway}@umich.edu

ABSTRACT
In designing educational research software for the classroom, we discover three distinct user groups whose needs must be considered: students, teachers and researchers. Our experience is that, despite employing user-centered design practices, our software tends to fail in its overall purpose if all three groups are not considered. To illustrate this conclusion we examine three projects, *PlanIt Out, RiverBank* and *Model-It,* as case studies of user-centered design processes. All had competent staffing and followed standard design practices yet two failed in their overall goals as a consequence of failing to consider the existence of multiple user groups.

This analysis leads us to conclude that traditional design practices tend to fall short in domains that involve multiple user groups, and that special methodologies for analysis of user groups need to be developed. We find that the relevant groups in multiple-user-group design can be identified and characterized in terms of distinct needs and relationships between groups. We develop a small taxonomy for identifying multiple-user-group situations, and suggest that user group analysis must include steps for identifying the relevant groups, determining relationships between groups, and identifying and resolving potential conflicts in the goals for each group.

KEYWORDS
Multiple User Groups, User-Centered Design, Educational Software, Participatory Design, Contextual Design.

INTRODUCTION
Designing user-centered [6] software for a group of users is not easy, even when designing for a single, well-defined user group. However, this process may seem easy in contrast to designing software to be used by more than one group. Not only are there more demands on the design, there is also a lack of methodological support for handling the special problems that arise when designing for multiple user groups. Consideration of multiple user groups is common in marketing and organizational contexts, but the practicalities of designing software to be used by multiple groups simultaneously have never been fully addressed. The literature on user-centered design occasionally admonishes designers to consider the possibility of having to address the needs of multiple user groups [2, 9], but existing methodologies tend to imply a single group of users. They offer neither an explanation for how multiple user groups might be identified, nor for how the potentially conflicting needs of such groups might be integrated.

Our definition of "multiple user groups" differs from a common notion of this term. This term is often used to refer to a situation in which, for instance, a product is designed to impress a group that has the authority to make purchasing decisions for an organization, but whose work does not actually depend on the software. For example, school superintendents may be charged with deciding what software will be purchased anywhere in a school district, yet they themselves will never use the software. In our definition of multiple user groups the needs of each group must largely be distinct, and the immediate work of all groups must in some way depend on the software. This does not imply that all user groups must be hands-on users of the software, as shall be explained later.

We represent a group of researchers in human-computer interaction at the University of Michigan. A part of our research centers on the special demands of educational technology, in particular, software design for the pre-college science classroom. Our software is used simultaneously for research in human-computer interaction and in education. In our work, we have found that our designs must consider the needs of three user groups—students, teachers and researchers—as well as the relationships between these groups. Our designs consistently fall short in achieving their goals when we fail to adequately consider the needs of all three groups, despite following standard user-centered design practices. We believe that this indicates a shortcoming of existing user-centered design methodologies.

We reach this conclusion by conducting a retrospective

[*]Current address: Phaedra@media.mit.edu, MIT Media Lab, Cambridge MA.

analysis of the design processes of three of our projects, *PlanIt Out, RiverBank* and *Model-It*. We examine each project to understand how it succeeded or failed, then examine the design processes to see how the failures occurred. We find that a significant portion of the failures can be traced to the implicit emphasis, in user-centered design methodologies, on a single user group. In comparing the processes of the three projects, we find that a unique combination of circumstances led Model-It, which achieved its overall goals, to design for multiple user groups unintentionally. In Model-It's success we find the seeds of a methodological framework for multiple-user-group design, in which participatory design and contextual design both play pivotal roles. This framework begins with a series of steps that constitute an analysis of user groups.

MULTIPLE USER GROUPS

The critical difference between designing for one user group and designing for more than one is, in our experience, the need to take relationships between groups into consideration. These relationships may be classified by the kinds of dependencies that exist between groups: independent, result-dependent, or process-dependent. Two or more groups are:

- *Independent* when their use of the same software is entirely coincidental. The work that is affected by the software used by one group in no way depends on the work of the other. For example, lawyers and newspaper reporters may use the same word processing software and may, indeed, have different needs of the software. In general, though, these groups and their needs are independent of each other.

- *Result-dependent* if one group is dependent on the *outcome* of the work of the other group. This occurs when one group produces something—including intermediate or partial results—that is then used by another. For example, an order department may use database software to store customer orders that the shipping department later fills.

- *Process-dependent* if the work of one group depends, in some way, on the *way* in which another group performs its work. For example, if teachers want students to think in specific ways, the educational software their students use should encourage those ways of thinking.

Note that process-dependent groups may also be result-dependent. For example, teachers may want to examine students' work to determine how students are thinking.

These dependencies are critical because they may affect detailed decisions of the software design, including minute user interface decisions. For instance, in the project-management software PlanIt Out, one of the case-studies described later, students are required to enter a "driving question" for their project at a certain point in their task. To help students to realize the significance of this formulation, the software was designed to accept only input that was at least eleven characters long and terminated by a question mark. Now, had the teachers not found it necessary for their students to grasp the *motivation* for a driving question—to help the students see that their task was to provide an answer to this question—the question mark could have been omitted, or automatically supplied by the software. In this case, the conflict between the teachers' need for student understanding and the students' need for ease of use was resolved in favor of the teachers. However, other scenarios can be imagined where a resolution is not so easily achieved.

The three user groups with which we are working are process-dependent. Teachers and researchers depend on the students to think in certain ways, and researchers depend on both teachers and students to perform in certain ways. The groups are also result-dependent in that students, teachers and researchers all depend on artifacts produced by each other such as completed assignments, curricular materials and software. Although our circumstance of three process-dependent groups may be somewhat unique, any training situation involves at least two groups as does such a familiar situation as usability testing.

CONTEXT

The three projects we examine are part of *ScienceWare*, an NSF-funded project investigating the design, construction, and integration of educational software into the pre-college science classroom. The ScienceWare project involves a three-year-long collaboration between the students and teachers at Community High School (CHS) in Ann Arbor, and researchers at the University of Michigan. To date, the ScienceWare project has produced ten software applications for the classroom, developed in a collaboration by students, teachers and researchers, tested at Community High School and, in a number of cases, integrated into the science curriculum there.

CHS is a 400-student, alternative public high school. Three of its four science teachers have worked closely with education researchers at the university to change from a traditional, didactic model of instruction to a more progressive, project-based one. The result is a three-year instructional sequence, Foundations of Science (FOS), that integrates traditional science curricula such as biology, chemistry and earth sciences into one unified curriculum. Integral to the FOS initiative, and the focus of efforts by university researchers in educational technology, is a large-scale integration of computers into the science classroom. The FOS sequence is required for all ninth, tenth, and eleventh grade students (ages 13-17).

Students: CHS students are representative of the surrounding community and are comparable to other high school populations in the area in socio-economic, educational and technological backgrounds. The Ann Arbor area as a whole is unusually affluent for this part of the Midwest; students tend to have more access to technology, both at home and at school, than in other parts of the Midwest.

Teachers: The three science teachers who began FOS had from four to nineteen years classroom experience with traditional science curricula and teaching methods prior to starting the FOS program. At the outset, only one of the three teachers was proficient with computer technology. The other two, while open-minded, had minimal prior exposure to computers. The ScienceWare project grant funded leave time for the teachers, reducing their teaching load and enabling them to spend time working on curriculum design and collaborating with researchers.

Researchers: The researchers involved in the ScienceWare project are a diverse group, coming from the University of Michigan School of Education, the College of Engineering, and beyond. These faculty and students conduct research in education, educational technology, computer science and human-computer interaction. Graduate students routinely observe classroom activities at CHS for a variety of research studies, and are in constant contact with both teachers and students at the high school.

University faculty and students are involved in the design of the software to varying degrees, providing project management, design, development and knowledge-domain content expertise as needed. The software is primarily designed by graduate students, and undergraduate students assist with implementation. A number of the students associated with the projects have had professional experience in relevant fields such as software engineering, graphic design or secondary-school education prior to their current academic careers. Experts in other disciplines are recruited for projects as needed.

THE PROJECTS

The three projects we examine are *PlanIt Out*, a tool to support students in project planning, *RiverBank*, a tool that supports students in collecting water quality data to be analyzed and shared with other schools and organizations, and *Model-It*, a tool to make modeling of system dynamics accessible to students. All three projects have been tested in classrooms and have proven successful with students in ease of learning and ease of use.

PlanIt Out

PlanIt Out is a tool designed to be used by middle- and high-school students in planning and implementing collaborative science projects. Describing their experiences after the first year of the FOS program, the teachers found that the students seemed to have particular trouble with planning their projects, managing their time and coordinating their four-person groups. Therefore one of the design goals of PlanIt Out was to help provide a framework for learning about planning, delegation and organization.

The design team was composed of four people: a professional software developer with HCI experience, a secondary developer with educational technology research, teaching and school management experience, a professional graphic designer with interface design experience, and one FOS teacher. The design process approximated a participatory design approach [10], with teachers serving as the design team's participatory design partners. The underlying assumption was that the teachers, as constant observers of student behavior, could serve as advocates on behalf of the students, able to articulate the needs of students. The students, who were also consulted, were assumed to lack the ability to plan their own projects themselves, and were also seen as unable to sufficiently articulate their own learning needs of the software.

An understanding of current classroom practices, a model of the student user and a task analysis were developed based on interviews with five FOS students, extensive consultation with the teachers not already on the design team, and interviews with professors in education and in educational technology. Based on this research, a prototype of the software was iteratively developed, and tested by the three FOS teachers, seven graduate students, and six FOS students, selected by the teachers, on a variety of occasions. Members of the design team observed all testing sessions. Prior to releasing the software for use throughout the FOS program, a one-week field test was conducted in a pilot FOS classroom.

The students seem to accept the software. They have had no apparent major problems learning to use PlanIt Out or using it in their project work. There were no problems reported with the students fulfilling, in a timely fashion, the assignments that involved the software. This impression is corroborated by teachers and classroom researchers. One student volunteered that the software was "exactly what we needed" to help plan and execute their projects, and keep project group members on task.

The teachers also uniformly spoke highly of the software; in interviews they insist on PlanIt Out's ability to help their students think about their projects' implementation more concretely, and that it helps students collaborate more effectively. FOS teachers currently use PlanIt Out in five classrooms, with over 125 students using the software. They plan to expand its usage to include all science classrooms in all grades, and are

making plans to integrate it more fully into their curriculum.

Several faculty members conducting educational research, however, felt that the software failed to reflect their needs in an important way. For these researchers, the lack of support for *conceptual design* of projects would seem to reinforce traditional educational methods, at the expense of the more novel methods that both the researchers and the teachers are trying to advance in the classroom. As a result, key members of the research community felt that PlanIt Out, in supporting only the task of planning the *execution* phase of a project, may actually undermine the very educational approach it was intended to support.

During the design and development process, these researchers had expressed their concerns in interviews with members of the design team, and had tried to advance a vision of the software that provided at least some support for conceptual, and not just implementation design. However, this vision of the software's task differed significantly from the teachers' needs of the software, of a relatively simple tool that would support the current process in the classroom.

Unable to reconcile the two disparate views and with limited time available, the design team elected to implement the teachers' vision, at the expense of the researchers' vision. The team felt that they stood on firm methodological ground, based on user-centered design, for such a decision: The students were the end-users, and both they and their task could be adequately and accurately modeled using input from the teachers and students, despite the researchers' differing viewpoint. In light of user-centered design, the researchers' input was treated as unimplementable advice that could justifiably be ignored.

RiverBank

RiverBank is a database tool designed for middle- and high-school students to collect, store and visualize data on water quality. The end users of this database software were considered to be the students who would be performing the actual data collection in the field, as well as data analysis in the classroom. The goal of RiverBank was to store the data, share it with groups outside the school, and allow for easy entry, access, and visualization by students.

Although developed as part of the ScienceWare project in collaboration with CHS, RiverBank was also part of another project to create a vast database on water quality. This project was under the supervision of GREEN, a Michigan-based, non-profit group that organizes and coordinates groups to collect data on water quality internationally. GREEN compiles, maintains, and examines this information and lends assistance to people who want to do such monitoring [7]. To this end, GREEN distributes a catalogue of monitoring and data-collection products, some of which they produce themselves.

The development of RiverBank followed an iterative design approach, in which the design team worked in close collaboration with the GREEN researchers, with additional consultations with FOS teachers. RiverBank's development cannot, however, be described as participatory design, since neither students, nor teachers, nor researchers from GREEN or elsewhere participated regularly in design meetings.

RiverBank's design team was composed of four software developers who were all undergraduate, computer-science-major seniors. Two had at least a year of professional software development experience, and the other two had a year or less such experience. The composition of RiverBank's design team is the most homogeneous in our study, and their design practices reflected the least amount of traditional user-centered design techniques. However we include this example for a number of reasons, chief of which is the fact that the FOS students who used it in the classroom seemed to like the software and find it usable in completing their assignments, as specified by their teachers. Further, as developers working in a human-computer interaction research laboratory, the design team eventually became aware of issues in user-centered design, and tried to incorporate some of its practices, albeit sometimes late in the process.

To this end, the design team conducted a series of meetings with representatives from GREEN whose researchers reviewed and tested the software. RiverBank's developers conducted a single session of user testing using one FOS student toward the end of the development process. Additionally, members of the development team regularly did usability testing on parts of the software on which others had worked. The designers met with the teachers on at least four occasions during the design and development process to solicit their input on the design.

When released to the FOS classroom, the students apparently had no problem either learning to use the software, entering the data or using the interface in general; neither teachers nor classroom researchers who witnessed the students' usage of RiverBank reported any problems of substance with the students use of, or interaction with, the software. The researchers at GREEN similarly appeared satisfied with the final product as demonstrated by its acceptance into the GREEN product catalog.

However the teachers found several flaws with RiverBank. For instance, the software was designed with neither graphing capability nor direct compatibility with any commercially available software that offered even simple visualization capability, such as Microsoft Ex-

	Goals	Criteria
Students	• Get good/passing grades • Do their assignments with minimal hassle • Learn skills that will help them in the future • To have fun	• Usability • Learnability/relearnability • Engagement/interest
Teachers	• Teach students • Develop and refine the curriculum • Create a manageable, controlled process in the classroom	• Supports learning • Fits classroom dynamics • Fits teacher's pedagogical goals and plans
Researchers	• Advance and/or refine educational models • Test educational theories • Observe classroom culture	• Correctly implements educational theory • Obtain valid, meaningful data • Elicits certain user behavior

Table 1 Goals for each group and attendant criteria for software success

cel. For the FOS teachers, the value of a classroom database tool lay not merely in the ability to enter data, but also to give it meaning through visualization. The teachers had voiced these priorities early on, but GREEN convinced the design team that a more generic way of exporting the data, into text files, would extend the abilities of the software, and make it more appealing for a wider audience. The teachers found this solution slow and impractical in the classroom.

Model-It

Model-It [4] was designed to support high school students in building and testing models of dynamic systems. As in the prior cases, the end-users were considered to be the students. The project was primarily motivated by researchers in education and technology, who wanted to make scientific modeling accessible at the high school level [5].

FOS teachers had never seen or done computational modeling in the classroom, but were very open to the project from the beginning. They had no preconceptions about the design or functionality of the software, which proved to be a tremendous advantage in the software's design. As a consequence, a curriculum for Model-It, including tutorial materials, study guides, and activity plans, were developed as part of the project.

Model-It's design team consisted of a core of two and an extended team of six. The core was comprised of the project lead, a computer science doctoral student with a background in educational technology and human-computer interaction, and a masters student in computer science. The extended team included three education researchers, two teachers, and an expert in stream ecology (the domain for which the software was initially developed). The core did the bulk of the design and implementation, and the extended team met occasionally to develop a task analysis and student user-model for the program, for design reviews, and to develop the supporting curriculum.

Model-It was developed iteratively: Based on several months of participant-observer studies conducted in the classroom and on field trips, a prototype was developed and tested. One-hour test sessions were conducted with each of two domain experts, one teacher, and four high school students. None of the test subjects were members of the extended design team. The project lead was present at each session to observe and conduct contextual inquiries [3]. The student sessions were also videotaped. A pilot class of ninth graders, led by the researchers, used the software prior to its release for regular use in the school.

Model-It appears to have been successful for the students, teachers, and researchers. Students found the tool both learnable and interesting: "It's neat, ..." "and it makes you think more about a real-life situation, where's there's no real answer, you set it up and everything." They frequently spend extra time outside of class on their model-building activities, investing more time and energy than required to simply complete the assignments. Judging by teachers' perceptions as well as grades of students' models and reports, students are learning valuable skills and knowledge. Teachers are pleased with the software, because it fits well with the Foundations curriculum and the teachers' educational goals. Finally, the researchers felt that they were able to effectively test their theories of the design of educational technology.

Two special circumstances appear to have contributed to the success of the Model-It project. First, the project introduced an entirely new activity into the high-school classroom, so the teachers' expectations were not pre-existing, but were defined by the project. Second, the primary developer was both an HCI designer and an educational technology researcher, and therefore represented the researchers. The first condition made the need to consult with students, teachers and researchers in the design process inescapable, and the second supplied both the necessary design skills and domain expertise in a pre-integrated form. Lastly, the mix of disciplines represented on the design team resulted in a naturally occurring participatory design effort.

THE PROJECTS REVISITED

Now that we have seen how the three projects succeeded or failed as single-user-group designs, let us consider

	PlanIt Out	RiverBank	Model-It
Students	None	None	None
Teachers	Project management tool	Data collection and visualization tool	Modeling tool and introduction to modeling
Researchers	Project planning tool	Tool to help schools collect and share water quality data	Tool for students to build models of complex systems

Table 2 Expectations of teachers and researchers

what differences might have come about had they been considered multiple-user-group designs. First, we suggest the addition of *analysis of user groups* to traditional user-centered design. Such an analysis is composed of at least seven steps:

1. Identify relevant groups,
2. Identify goals for each group
3. Establish the attendant criteria for success for each group,
4. Determine the relationships between groups,
5. Identify potential conflicts between goals and criteria of groups,
6. Resolve conflicts between groups, and
7. Establish criteria for success as a multiple-user-group design.

Iteration of these steps will, of course, be necessary, with some steps taking place simultaneously and not necessarily in the order indicated.

1. Identify Relevant Groups

The multiple-user-group design situation is heralded by the presence of users who have distinct, differing needs or who exhibit result- or process-dependencies. Merely being alert to the possible existence of multiple user groups is a significant step in identifying the relevant groups. Traditional methods for user and task analysis may all provide the necessary information, with contextual design [3] seeming particularly promising.

In the case of the three projects the three user groups—students, teachers and researchers—were readily apparent. In the case of RiverBank, a fourth group, the GREEN researchers, should also have been considered as a potential distinct user group.

2. Identify Goals For Each Group

The goals for each group can be identified using traditional user-centered design techniques. For the purposes of this retrospective analysis, we have articulated the goals for each of the three user groups for each of the three projects. These goals are summarized in Table 1.

Students: Students generally want at least to be able to complete their assignments with a minimum of hassle; at best, they want to enjoy doing it. In most cases, the longer-term goal is to get good grades; Further, some students are aware of a need to learn skills that will help them later in life, whether in school or in the job market.

Teachers: Teachers generally want to teach their students effectively, as best they can. At the very least, they want their students to learn concepts and information that will help them pass state proficiency examinations. Again, at best, the experience should be enjoyable. To help them achieve their goals, teachers need to develop and refine curricula and associated course materials. In the FOS program, both are intimately linked to the software design.

Researchers: Researchers want to be able to conduct research studies that range from ethnographic studies of a classroom culture to testing specific hypotheses regarding technology and education. The software is often only one component in a larger research plan, so the researchers need their educational theories to be implemented correctly and accurately.

3. Establish the Attendant Criteria

The criteria for success for each group can likewise be established using traditional user-centered design methodologies. We have articulated the criteria for each of the three user groups for each project in the present analysis. These criteria are summarized in Table 1.

Students: In order to accomplish the goals we have identified for students, the software needs to engage their interest, be usable and easily learnable. Students are often exposed to classroom software in sporadic bursts, so re-learnability is also important. These general goals match traditional objectives of user-centered design.

Teachers: The criteria for success for teachers are that the software must support student learning, fit classroom dynamics, and fit the teacher's pedagogy and goals. It must also support the ability of the teachers to effectively manage the process of project-based instruction.

Researchers: The researchers' goals of conducting scientific pseudo-experiments, and studying educational reform requires that the software not only elicit certain types of user behavior and allow for collection of appropriate and valid data, but that the educational ideas thought to be studied are indeed those manifested in the software.

	PlanIt Out	RiverBank	Model-It
Analysis of Existing Task Participant-Observation Long Interview			
Participatory Design Prototyping	*		*
Contextual Interview Usability Inspection Usability Testing			**

* Using teachers as student surrogates.
** Only employed during testing, not during task analysis.

Table 3 User-centered design techniques employed by each project

4. Determine the Relationships Between Groups

As in step one, simply being alert to the existence of relationships between groups, along with some understanding of the types of such relationships, is a powerful tool. Some relationships may be immediately obvious while others may be more subtle. In the latter cases, the designer can begin by first trying to determine the existence of result-dependencies, since process-dependent groups seem to rely on results to monitor the process taking place, and so tend to also be result-dependent. Organizational and work-flow structures are good indicators of result-dependencies, and techniques to understand organizations, such as those employed by computer-supported cooperative work [8], may help uncover such dependencies between groups. Once result-dependent groups are found, these cases may be checked for process-dependence. Process-dependencies are common in contexts where learning is critical, where the software must encourage users to think in certain ways and not simply produce certain results.

We find the relationship between teachers and students in the three projects to be process-dependent, and that evidence for such dependence can be found in teachers' stated goals for the software. PlanIt Out's driving question prompt described above provides an example. The teacher on the design team requested support in the interface for insisting that students not simply enter text into the driving question field, but that the software encourage the student to reflect on the importance and substance of the question itself. Such emphasis on how the student thinks clearly indicates process-dependence. Similarly, in RiverBank, the teachers' need for visualization capability reflects process-dependence. Similarly, the researchers are process-dependent on both students and teachers.

5. Identify Potential Conflicts

Conflicts in goals will likely arise where groups have differing goals and criteria for success, in parts of the design where they are interdependent. Participatory design is well-suited both for identifying and resolving conflicts, and is therefore highly suited for multiple-user-group design. Of course, use of participatory design requires the proper identification of user groups, so that each may be represented appropriately.

Model-It succeeded because its designers intuitively and adequately approximated participatory design. The project lead herself represented the researchers, and included teachers on the design team. Students were not included formally, but were consulted regularly. It seems clear that had the designers of PlanIt Out and RiverBank used such an approach, they would not have felt at liberty to ignore input from researchers and teachers, respectively.

Table 2 summarizes the expectations teachers and researchers had of each of the three projects after the initial stages of design. (These have been articulated for the current analysis.) Clearly differences existed at an early stage in the design process, yet user-centered design failed either to identify or address them. This failure was obscured by the traditional focus on *the* user group, in this case the students, who notably had no expectations of the software. The consonance of Model-It's expectations between the teachers and researchers is probably due to the close collaboration between the two groups.

6. Resolve Potential Conflicts

As mentioned earlier, participatory design seems a likely technique to apply to conflict resolution.

7. Establish Overall Criteria

Finally, the multiple-user-group designer must establish criteria for success for the design as a whole. In order to do this, the criteria for each group must be prioritized, and the criteria for all groups merged. This will obviously be an on-going process, again requiring such conflict resolution strategies as are offered by participatory design.

We have attempted to indicate what the overall design criteria for each of the three projects might be when considered as multiple-user-group designs:

PlanIt Out: Given the limited resources for development, a simple union of teacher and researcher criteria would not have been feasible. Some compromise on a tool offering integrated support for conceptual project design as well as implementation design and manage-

	PlanIt Out	RiverBank	Model-It
Students	*Success* • Simple and usable	*Success* • Usable	*Success* • Usable
Teachers	*Success* • Supports day-day project planning and management (project implementation)	*Failure* • Wrong/incomplete data types (incomplete insect specification) • No graphing ability	*Success* • Defined expectations to match what software provided • Curriculum developed
Researchers	*Failure* • Doesn't support conceptual project design	*Success* • Common data format • Low cost, public domain software (available to schools)	*Success* • Enables students to build models

Table 4 Results of designs for each user group

ment, but with less functionality for each, would have been expected.

RiverBank: Similar to PlanIt Out, it appears that a balance would have been reached between the teachers' need for visualization and the researchers' need for a low cost tool that could easily be distributed.

Model-It: The overall criteria for Model-It would probably not have changed significantly. However, these criteria would probably have been made more explicit earlier in the design process.

CONCLUSION

The preceding description of the design processes of the three projects demonstrates that the composition of the design teams and the user-centered design techniques applied were such that successful design outcomes could reasonably be expected. Yet, in spite of using user-centered design techniques, as summarized in Table 3, two of the three projects fell short in achieving their overall goals, as indicated in Table 4. Not only did traditional user-centered design fail to help resolve conflicts in the design input, it prevented the designers from recognizing the significance of the conflicts. This kept them from identifying the differing design goals for each of the three groups: Facilitating learning for students, supporting personal teaching strategies and curriculum for teachers, and eliciting particular types of user behavior and facilitating the collection of data for researchers.

We suggest that the implicit focus of traditional user-centered design methods is a major source of design failure—one that may be eliminated by intentionally considering multiple user groups. User-centered methods must be extended to identify and address multiple user groups, particularly for domains, such as educational research, that exhibit process-dependencies. Based on our experience, contextual and participatory design appear to be good starting points for formulating a methodology for multiple-user-group design.

REFERENCES

1. Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., Palincsar, A. (1991). Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning. *Educational Psychologist*, 26(3 & 4), 369-398.
2. Butler, K. (1996). Usability Engineering Turns 10. *interactions*, v3, 1 (Jan.), 59-75.
3. Holtzblatt, K. Beyer, H. (1993). Making Customer-Centered Design Work for Teams. *Comm. ACM*, 10 (Oct.), 93-101.
4. Jackson, S. L. (1995). The ScienceWare Modeler: A Learner-Centered Tool for Students Building Models, demonstration, *ACM CHI '95 Human Factors in Computer Systems, Conference Companion*, Denver, CO.
5. Jackson, S. L., Stratford, S. J., Krajcik, J. S., Soloway, E. (to appear). Making System Dynamics Modeling Accessible to Pre-College Science Students. *Interactive Learning Environments*.
6. Norman, D. A., Draper, S. W. (Eds.) (1986). *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates.
7. Mitchell, M. K. and Stapp, W. B. (1993). *Field Manual for Water Quality Monitoring: An Environmental Education Program for Schools*. Dexter, MI: Thomson-Shore Printers.
8. Olson, G. M., Olson, J. S. (1991). User-Centered Design of Collaboration Technology. *Journal of Organizational Computing*, 1, 61-83.
9. Robertson, J. W. (1994). Usability and Children's Software: A User-Centered Design Methodology. *Journal of Computing in Childhood Education*, Vol. 5(3-4), 257-271.
10. Schuler, D., Namioka, A. (Eds.) (1993). *Participatory Design: Principles and Practices.* Lawrence Erlbaum Associates.