# Premature Adoption of a Constructive Educational Technology: A Case Study

Loki Jörgenson* Nathalie Sinclair†‡
Stephen Braham*‡ Ellen Balka§

Simon Fraser University, Burnaby, B.C. CANADA V5A 1S6

**Abstract**

*Is it feasible for middle school students to use component-oriented software tools to construct their own mathematical explorations and experiments? In order to explore the potential of a new constructive educational technology for mathematics, a program of guided collaboration between middle school students on Bowen Island near Vancouver Canada and researchers at Simon Fraser University was established. Novel and recently developed bridging methodologies were introduced to overcome the lack of maturity in the software. The goal was to predict its feasibility as an in-class resource for learning mathematics and to engage the students in the software design process.*

**Keywords** constructionism, mathematics education, JavaBeans(TM), collaboration, telelearning, applets, participatory design

## 1 Introduction

The goal of this project was to test whether a component-oriented environment designed for distance learning would support a "constructionist" approach to learning mathematics in a middle school context. The results were intended to refine the direction of current software development and to suggest how best to use this new technology in the classroom. As the software was not yet fully developed, the challenge was to support a "premature adoption" process in a field trial, in this case, at a small private middle school at a remote location. The students were engaged in a "guided collaboration"; they were encouraged to participate in the assessment and subsequently the development of the technology. They were asked to reflect on their use of the technology to aid the researchers

*Centre for Experimental & Constructive Mathematics
†Island Pacific School
‡PolyMath Development Group
§Assessment of Technology in Context lab

in developing the software and predicting how effective the technology would be when completed.

The central feature of the technology under review is its use for component-based construction of interactive programs; it allows relatively inexperienced non-programmers to build their own tools for exploring mathematical concepts. However, at the time of implementation the construction tools didn't yet support use by students. Consequently, the functionality and flexibility of the planned toolset was assessed by having an experienced teacher/researcher hand-build a set of interfaces focussed on certain mathematical concepts. This defined the scope of applicability the toolset possessed and helped identify where further development was required.

A pressing question facing the developers was whether middle school students would be able to employ such a technology effectively once it reached maturity. The students would have to learn new ways of relating to software; their roles would change from passive users to thinking creators, requiring them to master concepts and tools that are not typically taught at the middle school level. In order to facilitate this process, a "participatory design" scenario was implemented. The main idea of participatory design is to encourage active participation of the users themselves in the design of technology [1]. Through the user of a low-tech design tool, the students thus contributed to the development of the design and functionality of the planned toolset.

Some recent studies of new educational technologies have focussed on engaging children [6, 8], as well as teachers [3], in the design process. This approach affords an opportunity for researchers to gain a fresh perspective on their work and how children perceive and interact with their environment. These studies have typically involved children ages 5 to 10 and have employed approaches such as participatory design, technology immersion, and contextual inquiry. Similar ideas were adapted for use in this project.

## 2 The Participants

The project arose out of a convergence of several groups' interests. Initially proposed as a field testing project for a telelearning development programme, it eventually acquired a multi-facetted character reflecting the goals of each group.

The PolyMath Development Group (PDG)[1] working within the Centre for Experimental and Constructive Mathematics (CECM) is involved with the Tele-Learning - National Centre for Excellence (TL-NCE). Its contribution to TL-NCE is a project known as $M^3$Plexus, aimed at delivering *live* mathematical documents via networks.

The Island Pacific School is a small community-based middle school situated on Bowen Island near Vancouver, Canada. It offers a full curriculum to students in grades 7 through 9 with an emphasis on diverse and enriched experiential learning. Author Sinclair is the mathematics and information technology

[1]At the time, the PDG was composed of Trevor Bradley, Carlton Chan, Jen Chang, Paul Irvine and Terrance Yu as well as authors Jörgenson, Sinclair and Braham

teacher at IPS as well as a researcher at the CECM.

The Assessment of Technology in Context (ATiC) laboratory was established by author Balka at Simon Fraser University, focussing on the uses of participatory design and ethnographic methodologies. It had only recently opened when they were approached to participate in the project.

# 3 Educational technology for mathematics

There has been growing support for component-oriented architectures for educational software. For example, diSessa [4] has described and advocated "Open Toolsets" such as Boxer; these are flexible and malleable collections of components which can be combined to create "microworlds". These microworlds offer students an opportunity to explore the interaction of elements they have constructed themselves. Similarly, the SimCalc project [14] is currently developing educational components using "open" architectures which can be extended, customized, and integrated. See also [9] for a description of component-oriented exploratory software for Mathematics.

There are technical, social and pedagogical motivations for investigating the potential of component-oriented architectures. Their primary benefit is that sets of specific- or general-purpose tools can be built rapidly and relatively cheaply; they then can be easily modified, extended and combined to yield more tools. Moreover, tools built by other development groups can be integrated and customized to meet the needs of a wide community of learners.

So, instead of relying on professional software developers, teachers and learners are given a *toolbox* with which they can create their own interactive, dynamic resources. As diSessa points out, this might encourage a wider range of groups, including teachers and students - as well as programmers - to become involved in the design of tools and activities. It is envisioned that a large and powerful toolset might eventually emerge as a consequence.

Since these tools and resources can be distributed over the World-Wide Web as Java applets and services, they are not platform dependent and do not require the purchase of expensive specialized software or hardware. This ensures that the learning opportunities afforded by these technologies are available to the widest range of learners as possible.

Students working with a component-based toolkit will have an authentic integrated learning environment. They will have access to a broad spectrum of computational, symbolic, and visual tools which they can combine to test relationships and discover patterns. Students will have the opportunity to construct knowledge in response to a problem at hand - one which they will investigate by visualizing, transforming and simulating the mathematical concepts involved. This mode of learning reflects the "constructionist" approach spearheaded by Papert [12] and underlying much of the LOGO based environments, including StarLogo [13].

## 3.1 OpenMath JavaBeans

The technology studied in this project uses Java to construct components, known as OpenMath JavaBeans, which can be linked together to form applets. Recently JavaBeans were introduced by Sun Microsystems as a standard for a component-based architecture. Individual elements offering specific functionalities can be constructed and then subsequently interconnected as desired. Somewhat resembling a software version of Lego blocks, a reliable and easy-to-assemble set of JavaBeans offers a simplified development process that can support use by regular users as well as experienced programmers.

OpenMath is a standard for communicating mathematical objects between computer programs. It is also a general communications standard which supports the exchange of any non-textual, non-image data. It is currently being integrated with MathML, one of the first of the Markup Language extensions to be developed to deal with mathematics presented on-line.

OpenMath Javabeans can also be "shared" over the network. This affords the possibility for collaboration; students can work in real-time with with classmates, teachers and researchers all over the world. Whether they are building games or solving problems together, these collaborative activites will provide students with a rich and flexible social context in which they can build knowledge as groups and as individuals.

## 3.2 Constructing Resources with OpenMath JavaBeans

A JavaBean is to an applet what a disk drive or a monitor might be to a computer: It is a distinct component with a specific function which, when properly assembled with other components, results in a useful tool. The component can be easily re-used in another tool and connected in a variety of ways. Each JavaBean has a variety of possible inputs and outputs which expect or produce certain types of data. 1 shows a typical JavaBean used for translating points; it has three possible inputs including a number for each of the x- and y-axis translation values and the data points to be acted on. When it has received all three, it automatically translates the data by the indicated amount and outputs to whatever is connected to it.

The actual process of building an applet from JavaBeans is a relatively simple one. The ease with which a student might be able to do this is defined by two key factors:

- How well the construction environment has been designed
- How much flexibility and functionality is available in the JavaBeans

Most construction interfaces offer iconic representations of the JavaBeans which allow them to be interconnected in a familiar drag-and-drop fashion. They help identify the nature of the JavaBeans and support the user to connect them properly (outputs going to inputs for example). Some of the better interfaces show how the applet appears as the user adds components (for example, buttons or sliders).

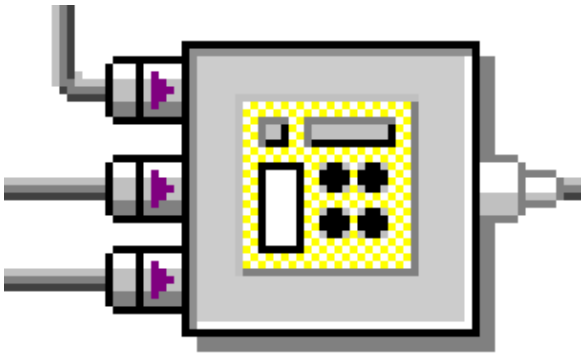

Figure 1: Enlarged view of an icon of a JavaBean for translating points

Typically, the user will begin by selecting and dropping JavaBeans onto the contruction area and then start to interconnect them. It is essential that the user understand how each JavaBean responds to input and what sort of output they produce so that they are effectively assembled together. The design process often takes place on paper ahead of time, with details of the construction being worked out as the applet is built and tested. In many respects, this is very similar to the usual programming process but without a focus on the details of a particular language or syntax. The most important aspect the user needs to understand is how an event, which might be generated by the push of a button JavaBean, is handled by other JavaBeans which receive it and they in turn trigger other events. This requires the ability to imagine a dynamic interconnectedness even while playing with the representation in the construction environment.

Once the user has gathered together all the necessary JavaBeans and subsequently connected them to each other, the resulting applet can be saved permanently. The applet can then be tested and, if satisfactory, can be integrated into an on-line resource or simply run as a separate tool. If necessary, the user can return to the construction tool and continue modifying the design until the desired results is achieved.

# 4 Engaging the Technology in the Classroom

This project focussed on answering the question:

> Can middle school students be reasonably expected to construct their own resources for learning using a component technology approach?

Two possible barriers to success were:

- Limits of the technology - would it be able to meet the requirements for building useful tools and resources?

- Limits of the students - would they be able to learn and employ the necessary skills to assemble their tools?

In the case of the technology, it was not clear that it would offer sufficient depth and breadth of functionality to fulfill the needs of the students. Certainly there were some strict limitations to the technology in its current form; in part this was a consequence of the relative instability of the underlying Java language and the immaturity of the supporting development environment. There was also some speculation that the OpenMath component set was not sufficiently comprehensive and would only be able to provide rudimentary tools. Keeping these issues in mind, the objective would be to identify the projected limits of a mature and fully implemented set of OpenMath JavaBeans.

For the students, it was important to determine if they would be able to employ the technology, mature or otherwise. The construction of resources from components requires an understanding of event-driven processes and an ability to visualize the flow of information. These skills are usually acquired through experience in programming and it was not obvious that the students would be able to adapt to the paradigm in a reasonable period of time. On the other hand, similar skills are necessary for popular board and video games[2], and so it seemed a reasonable premise that needed to be tested.

## 4.1 Testing for the technology barrier: Applets

The first stage was to support the creation of a set of focussed resources by a model teacher. Under a teacher-driven construction paradigm, we hoped to show that the technology could be applied by a knowledgeable educator. The teacher endeavoured to use the toolkit, in its initial incarnation, to create applets which could be used to investigate and explore relations and problems in a particular subdomain of geometry - that of transformations (e.g. translation, rotation, and reflection).

### 4.1.1 Implementation

The model teacher (author Sinclair) was *enhanced* in the sense of having expertise and a technical skill set beyond any to be reasonably expected from most teachers. Consequently, she was able to cope with problems, setbacks and other challenges in the classroom which would have made progress impossible otherwise. In addition, direct technical support was provided to handle critical problems encountered during the resource development. In this fashion, the lack of on-line help, stability in the software, supporting resources and facilitating environment was compensated for.

The specific topic of transformations was chosen in particular to take advantage of the visual and interactive strengths of the technology; tranformations

[2]Such as the well-known board game Mousetrap where children must construct a "mousetrap" composed of many interconnected pieces that fulfill an intended purpose (to catch a mouse)

Figure 2: Coping with group use of single-operator environment

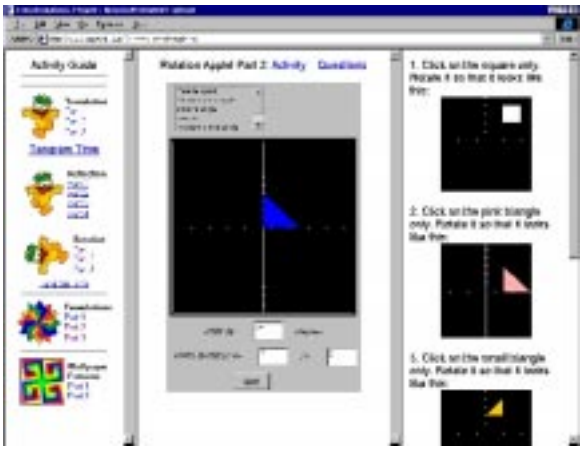

Figure 3: Student learning interface

are typically awkward for students to explore using pencil and paper. Students can often predict the result of a transformation but have more difficulty in identifying the single transformation that connects a shape with its image. Also, transformations is a potentially rich and engaging subject which can be extended to more advanced topics such as tesselations and wallpaper patterns. It was hoped that by having a dynamic environment, students could investigate these areas and gain a deeper appreciation for the fundamental concepts involved, such as isometry and congruence.

The applets themselves were constructed to satisfy a range of pedagogical goals; some were purely exploratory (explore reflecting shapes across the x-axis), some were geared toward applying a specific concept (use reflection to create a given compound shape) and others allowed students to investigate a particular problem (what are the effects of reflection on the coordinates of the shape?).

On average, the students had time to use three applets per session. They were given a set of questions to answer after each session. The purpose of the

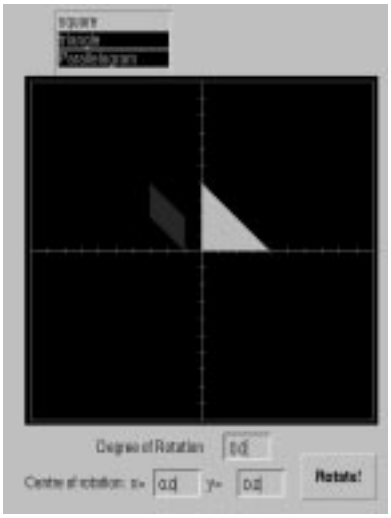


Figure 4: Rotation applet

questions was threefold: to support their on-line learning, to gauge whether they were grasping the concepts explored during the session and to see whether they were able to transfer their on-line learning to the more traditional pencil and paper environment. A student's typical learning interface appears in Figure 3. The applet used appears isolated in Figure 4. Figure 5 shows the underlying JavaBean schematic for the applet as it might be constructed in JavaStudio.

In practice, the twenty four students were assigned to three groups of eight students, rotating between using the four workstations in pairs and two other related non-computational activities. Each pair of students negotiated their roles at the keyboard and mouse, exploring the lesson cooperatively. They followed the provided text with additional guidance and instruction from the teacher. In general, the teacher managed the attention of all twenty four students and thus provided limited support to the computer users.

#### 4.1.2 Student Participation

One of the primary objectives of the project was to have the students actively participating in the design of the applets. Three oral de-briefing sessions were held in order to solicit the ideas and opinions of the students. In addition, each student completed a journal entry following a 40 minute session using the applets. The journal questions were aimed at gathering feedback on three issues:

- students' general feelings about the difficulty level and "fun quotient" of the applets,
- students' opinions about the design and layout of the applets, and

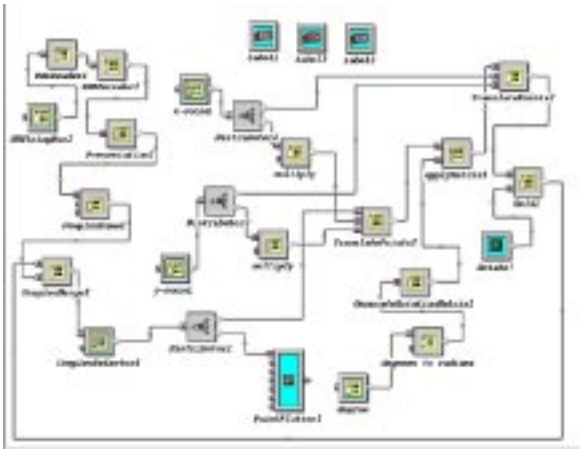

Figure 5: Schema for Rotation Applet

- students' assessments of the effectiveness of the applets and reflections on their own learning.

The oral de-briefing sessions were especially fruitful. A similar phenomenon has been observed in other collaborative research studies with children [8]. The students were very keen on voicing their opinions and discussing with their classmates. They were much more critical and verbose than in their journal entries. They were also much more creative; it was during these sessions that the students suggested imaginative alternatives to the existing applets and even completely new approaches. One such approach was to use collaboration tools to create a game that could be played on different computers.

#### 4.1.3 Observations

The students participated in approximately 12 forty minute sessions each and used a total of sixteen applets. Their feedback over this time as well as the teacher's observations gave strong indications of the strengths and weaknesses of the applets. The strengths lay mostly in the interactive nature of the applets.

Students enjoyed experimenting with different values and shapes; they were able to manipulate the shapes and create patterns with them that would have taken much more time (and frustration) on paper. The colourful and playful aspect afforded by the JavaBeans was also very motivating for the younger students. Another strength was the breadth of concepts that these applets allowed the students to encounter. Whereas transformations are traditionally taught as a self-contained geometry module, these applets bridged across the curriculum from number concepts through geometry to relations and functions.

The major weakness was the lack of interface components specifically designed to provide tutorial-like support. These include:

- GUI features such as labels on the axes and coordinates on the graphs;

- visual aids such as displaying the lines of reflection, the centre of rotation and the translation vectors;
- animation of shapes as they underwent transformations;
- feedback/response mechanisms for reinforcing students' progress.

The limitations and immaturity of the technology prevented the development of such components.

Another concern which arose was the impact of cooperative learning in a single learner context. Although the students were able to adapt very well to this working environment and gleen valuable experiences from exchanges with their partners (and amongst groups), there was no effort to design the applets for multi-user contexts. The students adapted by sharing control of the mouse and the keyboard so that each partner was responsible for certain functionalities. This seemed to work well. However, more effective means of sharing a learning environment are desirable; these issues have been addressed by other studies [7, 15].

## 4.2 Testing for the learning barrier: simCHET

The second line of investigation was aimed at identifying the degree to which the students could adapt to the design methodology required to apply the technology. It would have been desirable to have the students employ a construction environment like JavaStudio[3]. JavaStudio provides a high level of support for the users, offering a relatively intuitive interface with multiple perspectives, drag-and-drop functionality, understandable visual metaphors, and a WYSIWYG representation of the applet as it is being built; Figure 6 shows a typical JavaStudio construction session. However technical problems made it infeasible for them to use it. The tools employed by the model teacher such as BeanBox were inadequate for use by inexperienced users as they were much less intuitive and required an innate understanding of the programming issues.

In its stead, a low tech design tool called simCHET was conceived. It provides a facsimile environment that supports group learning and interaction, hands-on construction and heuristic design of tools. Everyday items such as paper, string, markers and tape are used to represent the essential features of a JavaStudio session. The students worked with the same mathematical concepts they had explored using the applets while learning the principles of the design methodology and then designing applets of their own.

### 4.2.1 Tools and resources

simCHET, simulated Computer Human Engagement Tool, was designed with a view to reproducing the interaction observed in the JavaStudio design process. It was immediately recognized that it would be impossible, and even undesirable, to mimic all aspects of JavaStudio.

[3](TM) Sun Microsystems

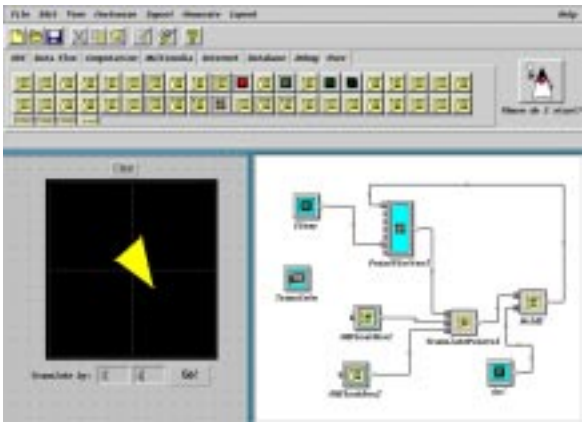

Figure 6: The JavaStudio construction interface: Note the applet being built on the left with its construction schematic on the right.

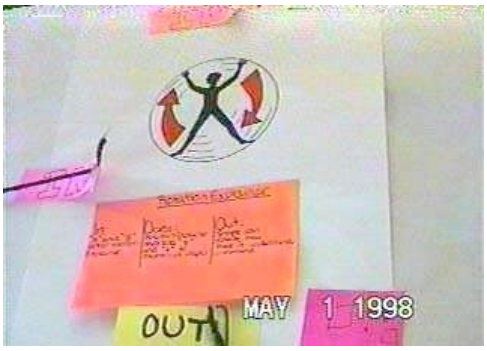


Figure 7: A rotation JavaBean with icon and descriptors

In practical terms, simCHET is a relatively simple tool. It is similar to the CARD/PICTIVE tools employed by Muller et al. [10]. This approach was originally pioneered by Beck and Cunningham [2] in support of teaching object-oriented thinking. It involves the use of index cards or note paper to represent the components or objects within the system. Spatial organization of the cards reflects the relationships between the objects and additional indicators and/or text provides the detailed functionality. Design can then take place with the alteration of either or both the cards themselves and their organization.

In the case of simCHET, the tool is composed of six basic elements:

- page-sized PostIt notes - component icons representing JavaBeans;
- drawable paper surface - a large area for assembling the icons and then adding relational notation and indicators;
- coloured pens - for notation; different colours depicting various processes


Figure 8: A full simCHET-designed applet

and mechanisms;

- smaller PostIt Notes - descriptors indicating the icon functions and dynamics (e.g. what happens when);
- yarn/string of various lengths - the connections between the various components;
- very small PostIt Notes - instantiation[4] tokens used to show specific values or data blocks as they are passed around the system.

Interestingly, one of the most important features seemed to be the descriptors; in initial tests both with adults and students, it was found that they played an essential role in enabling correct design procedure (Figure 7 shows a representation of a Rotation JavaBean with its icon and descriptor). This is not surprising since the descriptors define the functionality of each JavaBean. Similar features are found in most scenario-based tools of this sort [2, 10] as it provides the critical information about the nature of the objects.

Figure 8 shows an example of a full simCHET-designed applet created by students. Both simCHET and JavaStudio can be characterized by what they offer the user in support of the construction of JavaBeans-based tools. Java Studio has an explicit visual interface, dual representations of the function (schematic) and the form (interface appearance) of the design, and a computer-mediated manipulation of components (it won't allow you to hook JavaBeans together that shouldn't be). simCHET on the other hand has a multi-dimensional, multi-modal representation, notional icons and functionalities, unmediated manipulation of the relationships and reliance on user compliance with rules of interaction. Further it can be modified to suit the users' preference for how things should appear or be organized.

[4]Instantiation involved choosing particular values and events to anticipate as a designer how the design would respond.

### 4.2.2 Implementation

An initial testing of the proposed procedure was performed by the members of the project. Authors Jörgenson and Sinclair ran members of ATiC (including author Balka) through a mockup of the initial trial. A number of adjustments were subsequently made to the procedure which later were found to be important; for example, pads of paper were needed to allow students to rough out preliminary designs before attempting their final version on PostIt Note paper.

Three simCHET trials were undertaken within a period of one month. Each trial took about 1.5 to 2 hours. Except for the first trial, each was followed by an oral debriefing (the first trial was followed by journalling).

The first trial introduced simCHET to the students. It required additional time to communicate how the tools were to be used and to demonstrate construction techniques. They were led through the initial steps of the design as a group, identifying the components needed to build a relatively simple rotation applet with which they had worked before. A large support team of five researchers was present to provide constant support for the six working groups. Most of the groups were able to complete a working design within a two hour period.

A followup trial was run the next school day involving a somewhat more complex applet. The students sketched their designs on paper instead of using the full PostIt Note-based process. This was intended to reinforce their experience on the previous day without invoking all the support and materials needed for a full run. The students were able to construct the applets without difficulty in a much shorter time period.

The final trial was aimed at seeing if the students could now employ simCHET to design their own applets using transformations. They were aware that their ideas and designs would be potentially developed by the PDG team. The teachers stressed the fact that their creativity would be particularly welcomed, as would their suggestions for new JavaBeans. The students began by brainstorming ideas both individually and in groups. They were very excited by their ideas and very comfortable working within the framework of the existing technology. As they saw the need for a new JavaBean, they would inquire whether it was necessary and possible to make.

The students then planned the layout of their applets and made a list of JavaBeans that would be needed in order to make the applet work. After having discussed their designs with their teacher and members of the PDG team, the students began the construction of their applets.

### 4.2.3 Student Participation

Through the active participation of the students, it was possible to identify the difficulties they encountered in their design processes, the constraints of the given tools, and the strategies they developed to cope with these problems.

The students found the methods of connecting JavaBeans somewhat confusing; this was mostly due to the fact that they had not fully grasped the

Figure 9: PDG member collaborating with students

function of each of the JavaBeans. In some cases, they decided to group several functions into one JavaBean thus creating a type of meta-JavaBean. Although meta-Beans might be less flexible, they certainly seemed more intuitive to use for the students.

As the students worked on developing their own applets it became clear that the JavaBeans available to them were insufficient. Students repeatedly requested Timer JavaBeans, Collision JavaBeans[5], and Drawing JavaBeans[6]. Although they were not constrained in their design ideas, it was evident that in order to support the imagination of the students, a more sophisticated set of JavaBeans would have to be provided.

Remarkably, the students demonstrated their understanding of the underlying technology by explicitly suggesting new directions for research. They anticipated some of the JavaBeans still under development such as the Collaboration JavaBeans; these would allow students to construct tools which could be simulataneously shared across the network.

#### 4.2.4 Observations

The iconic representation of the JavaBeans played an important role in the students' learning processes. They created unique icons for each of the JavaBeans thereby establishing personal connections to them. While adding descriptors and connectors to each of their icons, they were able to develop an understanding of their functions and further refine their appearances. In addition, by taking their applets through a sample instantiation, they realized that a Hold JavaBean[7] would be needed to properly control the flow of data. These were rapidly integrated into each group's design during the session. It was apparent

[5]The Collision JavaBean would detect when two shapes had collided on the grid.

[6]The Drawing JavaBean would provide more flexible ways for students to create multi-coloured shapes

[7]The Hold JavaBean interrupts the flow of events, pending some action from the user like pressing a button.

that the students had acquired the event-driven mindset necessary for a correct design.

The students were actively involved in the process, obviously enjoying themselves and thoroughly engaged in their learning. One group even presented their applet design as an "interpretive dance", one member adopting the role of each JavaBean and then acting out its function! Each student in the group acted as a JavaBean and performed its task as required by the design. They even invented a new JavaBean whose purpose was to "disintegrate" the applet once the instantiation was completed! Although not technically pertinent, it was apparent that the students felt comfortable enough with the simCHET framework to extend it on their own.

The final trial showed conclusively that the students felt very comfortable with the simCHET framework. Moreover, it was evident that the nature of the technology did not constrain their imaginations. They were able to create unique applets and imagine functionalities that were not currently available. When told that certain functionalities would be more difficult to implement, the students chose to brainstorm other mechanisms to keep their designs interesting. Some of the mechanisms led to strategic collaborative games which were built by the PDG team.

# 5 Conclusions

The primary aim of this project was to determine the viability of component-based technologies in constructionist educational practices, in particular the applicability of OpenMath JavaBeans for use in middle school mathematics education. The two anticipated barriers to its use were tested for by implementing the existing technology in two stages; first, investigating the functionality of the JavaBeans toolset by building applets to teach a specific lesson plan in mathematics; second, using a low-tech participatory design tool to explore the capacity of students to learn and apply the principles of component-based tool construction.

In the first instance it was demonstrated, with some qualifications, that the technology could be applied to teach mathematics, at least in the case of transformations. The emphasis was as an adjunct to traditional teaching, especially in light of the apparent need for a more responsive and refined user interface with significant guidance mechanism. It was apparent that the nature of the learning and the breadth of the knowledge explored was affected by the presence of the technology: This was evident in scope of the material covered (details to be described elsewhere) - the teacher was able to explore aspects of transformations which are usually too advanced for middle school students - as well as the ability of the students to absorb and apply the ideas presented. While it hasn't been quantatively shown that students learned better or faster, it was a fact that they were able to explore ideas that would otherwise have been out of their reach.

In the second, it was clearly observed that students *can* learn and apply

component construction techniques and, perhaps more importantly, they can work creatively within that paradigm. This was the more important result as it was not entirely obvious that inexperienced users could adapt their thinking to the model supported by the technology. While it is most desirable that the computer fit the natural dynamics of the users, it is in the meantime inevitable that the value of such tools will depend on how readily they appeal to students' intuition.

Finally, this project experimentally applied guided collaboration methodologies in an effort to reach the conclusions above. Although the technology was not ready for trial use, it was found that its application could be represented in facsimile to a degree which provided the information needed. This approach relied heavily on the participation of students and the school as collaborators with the researchers and as such was a unique experience for both groups. It required the presence of an unusually skilled teacher/researcher and continuous technical support.

# 6 Future work

The research collaboration was very fruitful both for software designers and for students. While progress was slow developing the actual software, the process clearly affected both groups and their views on technology, both in general and in relation to OpenMath JavaBeans. This partnership could be enhanced by designing new methodologies that enable researchers to guide the collaboration more effectively. The work of Druin *et. al.* [5] suggests a methodology for working with children from ages seven to 10. Middle school kids (ages 11-14) can also be effective design partners; however, existing methodologies should be adapted appropriately.

As the technology continues to be refined and stabilized, attention should be turned to developing "better things to do and more powerful ways to think about what you are doing" [11]. What kinds of things can and should teachers and students be building with this technology? Will teachers build instructional modules for their students? Will students build games to enable them to play with new concepts? Can students and teachers alike build authentic and cross-curricular problem-solving resources? What parts of the curriculum can and should these games and resources cover?

The technology itself continues to be developed; the goal continues to be to make it intuitive and user-friendly for students without undermining its power and flexibility. In its current state, the technology is rudimentary and restrictive. The participation of its potential users will continue to guide its development. There are other examples of technologies applied in the classroom which have been developed in this fashion: The programming language LOGO, developed for use by children, offers a guiding framework for powerful-yet-intuitive interfaces. Similarly, the icon-based programming language for children ICOBotics(TM) emphasizes the importance of visual interfaces by allowing the user to write programs with pictures. The creation of the software

KidPAD from its parent PAD++ is an example of the important role carried out by children [6] in software development.


### Acknowledgements

This project has been made possible by the work of the entire PolyMath Development Group: Trevor Bradley, Carlton Chan, Jen Chang, Paul Irvine, and Terrance Yu as well as authors Jörgenson, Sinclair and Braham. Also essential was the work of the ATiC lab members, author Balka and ethnographer Michael Jones.

We would also like to thank BC Tel Advanced Communications for donation of an ISDN line to Bowen Island, Zentra Computers for donations of PC hardware, and Innovative Computing Solutions and Redesign for their networking support.

This work has been in part supported by research and equipment grants from the TeleLearning - National Centre of Excellence (TL-NCE) and the Pacific Institute for the Mathematical Sciences. We would also like to thank the Centre for Experimental & Constructive Mathematics, the Assessment of Technology in Context lab and the Island Pacific School for their support and participation in this project.


## References


[1] Ellen Balka. Political frameworks for system design: Participatory design in non-profit women's organizations in canada and the united states. In J. Gaertner and I. Wagner, editors, *Workshop Proceedings: Political Frameworks of System Design From a Cross-Cultural Perspective*. 1995.

[2] K. Beck and W. Cunningham. A laboratory for teaching object-oriented thinking. *Proceedings of OOPSLA '89, SIGPLAN Notices*, 24(10):1–6, 1989.

[3] George Chin Jr., Mary Beth Rosson, and John M. Carroll. Participatory analysis: Shared development of requirements from scenarios. *CHI '97 Proceedings: Conference on Human Factors in Computing Systems*, 1997.

[4] A.A. DiSessa. Open toolsets: New ends and new means in learning mathematics and science with computers. In E. Pehkonen, editor, *Proceedings of the 21st Conference of the International Group for the Psychology of Mathematics Education*, volume 1, pages 47–62. Lahti, Finland, 1997.

[5] Allison Druin, Ben Bederson, Angela Boltman, Adrian Miura, Debby Knotts-Callahan, and Mark Platt. Children as our technology design partners. In A. Druin, editor, *The design of Children's Technology: How we design and why?* Morgan Kaufmann, 1998.

[6] Allison Druin, Jason Stewart, David Proft, Ben Bederson, and Jim Hollan. Kidpad: A design collaboration between children, technologists, and

educators. *Proceedings of the Conference on Human Factors in Computing Systems*, pages 463–470, 1997.

[7] Kori Inkpen. The effect of turn-taking protocols on childrens learning in mouse-driven collaborative environments. *Proceedings of Graphics Interface '97*, 1997.

[8] Maria Klawe and Eileen Phillips. Engaging children as collaborative researchers: A classroom study with electronic mathematical games. *unpublished*, 1995.

[9] Chronis Kynigos, Manolis KKoutlis, and Thanasis Hadzilacos. Mathematics with component-oriented exploratory software. *International Journal of Computers for Mathematical Learning*, 2(3):229–250, 1997.

[10] Michael J. Muller, Leslie G. Tudor, Daniel M. Wildman, Ellen A. White, Robert W. Root, Tom Dayton, Rebecca Carr, Barbara Diekmann, and Elizabeth Dykstra-Erickson. Bifocal tools for scenarios and representations in participatory activities with users. In John M. Carroll, editor, *Scenario-based Design*. John Wiley and Sons, 1995.

[11] Seymour Papert. Teaching children to be mathematicians vs. teaching about mathematics. *AI Memo No. 249 and Logo Memo No. 4*, 1971.

[12] Seymour Papert. Situating constructionism. In I. Harel and Seymour Papert, editors, *Constructionism*. Ablex Publishing Corporation, 1991.

[13] M. Resnick. New paradigms for computing, new paradigms for thinking. In A. diSessa, C. Hoyles, and R. Noss, editors, *Computers and Exploratory Learning*, pages 31–43. Berlin: Springer-Verlag, 1995.

[14] J. Roschelle and J. Kaput. Educational software architecture and systemic impact: The promise of component software. *Journal of Educational Computing Research*, 14(3):217–228, 1996.

[15] Jason Stewart, Elaine M. Raybourn, Ben Bederson, and Allison Druin. When two hands are better than one: Enhancing collaboration using single display groupware. In *ACM SIGCHI'98*. ACM, 1998.