Experiments in Premature Adoption of Constructive Educational Technology

rough draft June 30, 1998

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 for the construction of their own learning resouces? In order to explore the potential of a new technology, a program of guided collaboration between students at Bowen Island near Vancouver Canada and researchers at Simon Fraser University was established. Bridging methodologies were developed to overcome the lack of maturity in the software in order to predict its feasibility as an in-class resource for teaching mathematics.

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

1 Introduction

The goal of this project was to test whether certain educational technologies currently under development would be feasible as constructionist resources for teaching mathematics. The results were intended to define the direction for further research and indicate how best to use these technologies in the classroom. As the software were not yet fully functional, the challenge was to support a "premature adoption" process in a field trial, in this case, at a small private middle school. The students were engaged in a "guided collaboration"; they were encouraged to reflect on their use of the technology and to aid the researchers in predicting how effective the technology would be when completed.

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

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 of the planned toolset was to be assessed by having an experienced teacher/researcher hand-build a set of interfaces focussed on certain mathematical concepts. The students were subsequently challenged to apply these concepts by building interfaces of their own in a facsimile construction environment.

Of equal importance was the question of whether middle school students would be able to employ such a system effectively once it reached maturity; they would have to learn new ways of relating to software, ways that might be more typical for a programmer than a user. Consequently 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, and thus empowering the users [1]. Through the low-tech design tool simCHET, the students contributed to the development of the design and functionality of the planned toolset.

Engaging children in the design of new educational technologies has become the focus of several recent studies [6, 8] as has engaging teachers [4]. It affords an opportunity for researchers to gain a fresh perspective on their work and how children perceive and interact with their environment. Many of these studies have 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 Telelearning, Information Technology and Ethnography

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.

2.1 PolyMath Development 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^3Plexus, aimed at delivering *live* mathematical documents via networks. The underlying mechanisms, all falling under the general umbrella of PolyMath, are being designed for use in both research and education in the mathematical sciences. The PDG is dedicated to exploring the technical potential and the social consequences of emerging network technologies.

Most of the systems are being constructed using Java(TM)[2] and related resources. Some of them employ a modular component-based approach to rapid tool prototyping and tool construction using JavaBeans(TM)[2]and OpenMath-based systems. These systems are expected to be applicable to a wide range of network-aware resources, from large scale digital publishing to tools for researchers to collaborative materials constructed for and by students.

With a view to exploring the constructionist potential of their component-based technology, the PDG launched this project. It was anticipated that a properly focussed set of tools could be utilized by students and teachers like a software version of Lego, supporting them to create their own learning resources and opportunities. Although the PolyMath technologies were still in the early stages of development, it was felt that a number of important issues should be addressed early on. In particular, the PDG needed to know:

- How easy would it be for non-programmers (like many teachers and students) to use the JavaBeans component approach to constructing resources?
- Would such systems be flexible enough to support the evolving and varying demands of teachers or would they instead constrain them?
- Are the OpenMath JavaBeans too specialized? How will they be used in contexts other than the ones for which they were designed?
- What new teaching methodologies are engendered and/or enforced as a result of using this technology? Likewise what kind of potential might this technology have for delivering a more integrated curriculum?

2.2 Island Pacific School

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

IPS was interested in working with the PDG for several reasons. Foremost, it offered a unique opportunity for the school to get involved in academic research; as a private community school, it has always made an effort to bring learning opportunities of all sorts to the students. As well, an information technology component had recently been added to the province of British Columbia's curriculum and the school was ready to set up a networked lab and develop a strong IT program. This collaboration would give students the opportunity to work with advanced network technologies and to reflect and voice opinions on the perceived effects that they have on their education.

Island Pacific School was also committed to integrating technology into other classes such as mathematics. Working with a mathematics research centre would enable IPS students to collaborate with mathematicians and access powerful on-line tools and resources.

2.3 Assessment of Technology in Context lab

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. The scope and requirements of the proposed study fit the lab's capacities well and so it was viewed as a good choice for a first

[1] Trevor Bradley, Carlton Chan, Jen Chang, Paul Irvine and Terrance Yu as well as authors Jörgenson, Sinclair and Braham

[2] Trademark Sun Microsystems

project for ATiC. In addition, it presented a unique opportunity to work across disciplines and to implement participatory design practices in a type of workplace (a middle school) that had not been a focal point of most other PD projects.

Subsequent discussions between members of the PDG, IPS and ATiC indicated that there was a good compatibility between the groups. A productive working relationship was quickly realized which fueled the project's development and led to a negotiated agreement between ATiC and the CECM. ATiC constructed an appropriate program for ethnographic study and provided the necessary support (ethics clearance, recording hardware like video cameras, expertise, etc.); CECM provided funding for the on-site ethnographer and managed the project itself. The interaction between ATiC, PDG and IPS was carefully delineated in a memorandum of understanding [2].

3 OpenMath JavaBeans: Educational technology for mathematics

The technology being investigated uses Java to construct components, known as JavaBeans, which can be linked together to form applets. This would allow non-programmers with limited familiarity with Java to build their own tools and resources with minimum effort. Each JavaBean has a particular functionality and has specific ways of being connected to other JavaBeans. The use of OpenMath supports functionalities which are specific for mathematics. These OpenMath JavaBeans form an object set which can be reused to build a wide variety of tools.

3.1 Constructionism

Distributed object technologies are being investigated by the PDG to see how they can support a constructionist educational framework. Constructionism [11] is a theory of learning which asserts the constructivist theories of Jean Piaget that learners *make* their ideas by constructing their own knowledge structures. To this, it adds the notion that this kind of knowledge building is particularly effective in a context where learners are actively engaged in constructing something (such as sandcastles or computer programs) in a social context to which they can attach personal meaning [12].

Such a learning environment could be supported by moving away from the existing rigid and drill-driven educational software to a more flexible component-based toolkit. Students and teachers might then be able to construct their own mathematical software for learning. Java can provide resources which integrate a broad spectrum of experimental and visual explorative tools. Students can use these tools to make connections between different mathematical representations, to test relationships and discover patterns, and to answer self-motivated questions. As students are exploring and hypothesizing, teachers can concentrate on the more subtle process of engendering experience and intuition.

The networking capabilities of these distributed object technologies also allow for synchronous learning and building so that students can share their tools, in real-time with classmates, teachers and researchers all over the world. Whether they are building games or solving problems together, it is believed that collaborative activities amongst students will engender the deepest and most fulfilling learning; in this rich and flexible playground students can exercise their doing/thinking/learning within a supportive social context.

3.2 JavaBean components

Java currently holds the popular focus in the arena of programming for network applications. It promises a number of highly attractive features which include being platform independent, network aware, potentially lightweight, robust, and increasingly Web-friendly. Due to the intense interest the industry is showing in this language, developments related to Java are evolving at a fantastic pace.

Recently JavaBeans were introduced as a standard for a component-based architecture. Individual elements offering specific functionalities can be constructed and then subsequently interconnected as desired. Resembling a software version of Lego blocks, a reliable and easy-to-assemble set of JavaBeans offers a simplified development process that can support its use by non-programmers as well as experienced programmers.

The technology being studied in the project was primarily a special set of JavaBeans, constructed to support mathematics. The version of Java used was v1.1 although at the time most publically available Java-capable systems did not fully support v1.1 functionality.

3.3 OpenMath standard

OpenMath is a standard for communicating mathematical objects between computer programs. These OpenMath objects are recursive data structures describing mathematical objects, from functions to data sets to theorems. OpenMath is 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.

Original work by the PDG includes the development of Java libraries implementing OpenMath version 1.0. The current release of the libraries is version 0.5 *early access release* and was employed to create the set of JavaBeans used in this project. Approximately forty generic OpenMath JavaBeans were developed.

3.4 Development Tools

Most stages of the Java development in Phase II employed Java Development Kit (JDK) v1.1.5 and v1.1.6. This required the use of less commonly available of browsers and viewers. Applet construction was primarily done using Beans Development Kit (BDK) v1.0 [3]. There were a number of distinct releases from November'97 to March'98 which were used. BDK offers a relatively simple Beans building environment called BeanBox; this tool was crude and occasionally behaved unpredictably. However it was one of the few tools available which reliably produced applets that worked satisfactorily.

JavaStudio v1.0 [3]offered the preferred component construction environment. Its interface was most intuitive and provided features that could be adapted for use in the classroom. However it proved to have several problems which could not be overcome until late in Phase II including not producing working applets. Regardless, it is JavaStudio's graphic user interface (GUI) which provided the model for part of this project's assessment process (see section 5.2).

3.5 Browsers and Applet viewers

In order to deliver the OpenMath technology to the classroom with supporting resources, browsers were needed. Unfortunately not all of the popular browsers were capable of handling JavaBeans or Java v1.1 applets. Microsoft's Internet Explorer v4.0 provided the most reliable support. However it failed to work with some of the later applets developed in Phase II, at which point the appletviewer from JDK v1.1.6 was used. Using appletviewer discounted the use of supporting HTML text and images in the lessons presented to the students.

[3]Trademark of Sun Microsystems

4 Phase I: Foundations

The initial phase of the project was primarily focussed on laying the foundations for the second phase. This included setting up the necessary software and hardware at both the CECM and IPS, developing the OpenMath JavaBean components, introducing the students to the computing environment and various educational resources, and normalizing the relationship between IPS, the CECM and ATiC. This phase was essential as it identified a number of issues which needed to be addressed before the PolyMath technology could be introduced.

4.1 The hardware/software environment

As the school was relatively poorly equipped to begin with, it purchased and installed an entirely new group of PCs. This included four 166 MHz Pentium MMX workstations with 17" monitors/32M RAM and a 180 MHz Pentium Pro server with 15" monitor and 64M RAM. A 10bT Ethernet LAN was set up using Windows NT for the workstation group. An ISDN line[4] to the mainland was set up with some difficulty[5], providing up to a 128 kbaud link. Considerable work was required to regularize the networking (routing, e-mail, Web servers, etc.).

One of the serious challenges to the project was the installation and maintenance of a stable PC environment at the school. The CECM is primarily a UNiX-based research group. While some significant PC expertise was available during the project, it rapidly became clear that the PC network software presented unique challenges. After myriad reinstallations, workarounds and trials, a highly simplified implementation of Windows NT was settled on, foregoing many of the features anticipated for use in the project.

4.2 Familiarization

Having dealt with the hardware and software issues (many of which were recurring throughout the

[4]Made possible by a generous contribution from BC Tel Advanced Communication

[5]It is long distance from the mainland to Bowen Island but a free local call in the other direction. This necessitated that the ISDN channel always be opened from the Bowen Island side.

project), a process of acclimatization took place. Students were excited by the presence of new resources. Some time needed to pass before they would treat the computers as everyday aspects of their learning environment. As part of the introduction, they were exposed to a wide variety of materials available on computer for mathematics education. These included CD-ROMs, Web pages, Windows-based programs, and non-OpenMath applets.

4.2.1 Other available software

The students experimented with several notable resources:

- MathProbe - a CD-ROM based mathematics dictionary with interactive activities illustrating the concepts being defined
- Logo - a well-known programming environment which emphasizes basic logic command control
- Sophisticated on-line applets from university research groups such as the Geometry Center[6]

With exposure to a broad spectrum of the available resources, the students developed a mature point of reference for their later exposure to the OpenMath applets.

4.2.2 Limited computing resources

Perhaps more challenging than the software was the difficulty adapting an environment designed for a single person to work in groups. There were only four functional workstations available to be shared by twenty four students. Due to time constraints, this meant that groups of at least two, and often up to four, students would work together with a single computer (Figure 1). This issue has been addressed in recent studies [7, 13] using specially designed systems which support multiple user control of a single pointer. In the IPS context, users were left to negotiate use of the resources and develop their own solutions.

4.2.3 On-line learning and estrangement

The students needed to adapt to the new techniques being employed for teaching with the OpenMath JavaBeans. While it seems self-evident that traditional and technological methodologies for teaching mathematics will share some aspects and be widely differing in others, the students themselves needed to adjust their approach to the materials. They gradually

[6] See www.geom.umn.edu

Figure 1: Several students working together at a single computer

learned how to use selected software, the Internet and CD-ROMs to learn and explore mathematics. They investigated different ways in which they could express their knowledge on a computer; from using word processing programs to creating diagrams to making Web pages in HTML. Most importantly, they became comfortable using the computer and confident in their own abilities to navigate through and handle its different aspects.

4.3 Ethnographic study

The presence of the ethnographic team in the classroom was yet another factor. The students were distracted by new faces, new behaviours, still and video cameras, the unusual questions and the behavior of the research team. This was accepted as part of the ethnographic process. Once again, phase I provided an opportunity for the students and ethnographers to develop a normalized relationship.

5 Phase II: Transformations

Phase II was 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. Despite these issues, the objective would be to identify the projected limits of a mature and fully implemented set of OpenMath JavaBeans.

With regard to the students, it was important to ascertain that students 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.

5.1 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 lessons to be constructed would be relatively shallow with regard to the information to be delivered. In these simulated learning contexts, the students would encounter familiar information and then compare how and what they learned with their prior experiences.

5.1.1 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. Initially, PDG members Trevor Bradley and Terrance Yu developed the OpenMath JavaBeans and worked directly with the teacher; Paul Irvine assisted with the PC network and software installation. In later stages, Carlton Chan provided the JavaBeans support with additional PC support from Angus McIntosh of Redesign. In this fashion, the lack of on-line help, stability in the software, supporting resources and facilitating environment was compensated for.

Figure 2: Student Working with Applets

A series of lesson plans were developed to teach a specific set of mathematics concepts, in this case transformations (e.g. translation, rotation, and reflection). These geometric concepts were chosen in particular to take advantage of the visual and interactive strengths of the technology. The lesson plans were centered around a collection of applets developed by the model teacher. Each applet was developed as needed, as if by a teacher who was doing weekly planning for each subsequent set of lessons.

The lessons were constructed around the progression of exploration, application and problem-solving. This progression was intended first to give students an intuitive understanding of each transformation, then to guide them into abstracting their properties, and finally to allow them to use their skills in a problem-solving situation. For example:

- Reflection
  - *Explore* across x-axis only
  - *Explore* across x- and y- axes
  - *Apply* to create a given compound shape
- Rotation
  - *Explore* rotation around origin
  - *Explore* changing degree and centre of rotation
  - *Explore* with self-made shapes
  - *Apply* to create a given compound shape

Solve the following:

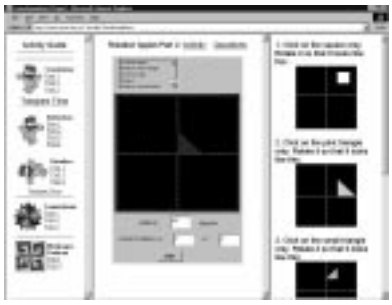

Figure 3: Student Learning Interface

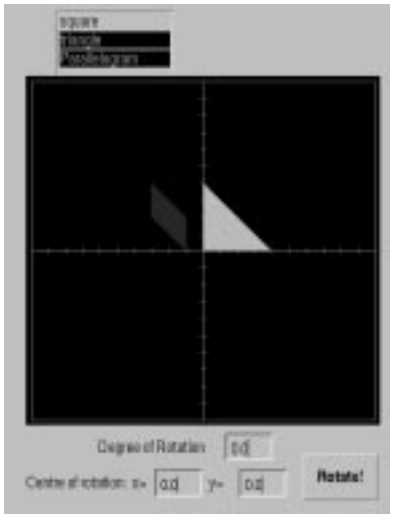

Figure 4: Rotation Applet

- What are the effects of reflection on the coordinates of the shape?
- Are reflections and rotations associative and-or commutative?

For a typical stage in the plan, one or two applets were needed in addition to supporting content. This included both directions on how to use the applets and a sequence of activites to be completed using the given applet. The students also had the chance to look at a set of questions that they would be answering off-line following their sessions. For the lesson above, the student's learning interface appears in Figure 3. The applet used appears in Figure 4. Figure 5 shows the underlying JavaBean schematic for the applet as it was constructed by the teacher.

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

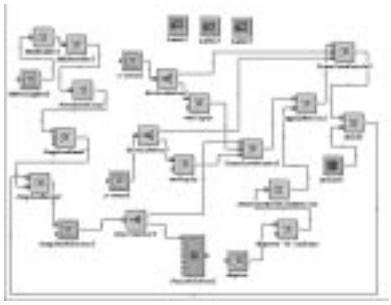

Figure 5: Schema for Rotation Applet

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.

5.1.2 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
- 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.

The students also completed a worksheet following each session. The questions were divided into two categories: (1) questions directly related to concepts pre-

sented through the use of the applets; these would reinforce their understanding and provide a means of assessment, and (2) questions which extended and supplemented these concepts; these were used primarily to introduce concepts that were more difficult to cover with the applets. It was decided to have the students work on these questions off-line both to free up computer time for the other groups and to observe the way in which they would translate their on-line learning to a more traditional setting.

5.1.3 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 beans 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 constructive means of sharing a learning environment are desirable; these issues have been addressed by other groups [7, 13].

Striking learning style differences emerged as the students used the applets. While half the students enjoyed the feeling of exploring and experimenting, the other half expressed strong discontent with the lack of explicit instructions. It is feasible to cater to both these learning styles by providing more guidance to those who need it. Due to time constraints, this issue was not addressed.

5.2 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 Java Studio. However technical problems made it infeasible for them to use it. The tools employed by the model teacher such as BeanBox were clearly inadequate for use by inexperienced users.

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.

5.2.1 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.

In practical terms, simCHET is a relatively simple tool. It is similar to the CARD/PICTIVE tools employed by Muller et al. [9]. This approach was

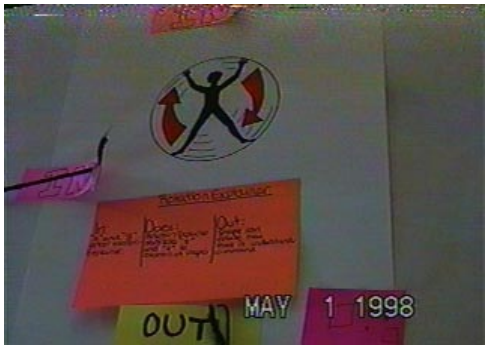

Figure 6: A rotation bean with icon and descriptors

Figure 7: A full simCHET-designed applet

originally pioneered by Beck and Cunningham [3] 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 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[7] tokens used to show specific values or data blocks as they are passed around the system.

Interestingly, the essential feature seemed to be the descriptors; in initial tests both with adults and students, it was found that it played an essential role in enabling correct design procedure (Figure 6 shows an example of a "rotation" bean with its icon and descriptor). This is not surprising since the descriptors peg the functionality of each JavaBean. Similar features are found in most scenario-based tools of this sort [3, 9] as it provides the critical information about the nature of the objects.

In order to understand how simCHET is both similar to and distinct from Java Studio, both can be characterized by what they offer the user in support of the construction of Beans-based tools. Java Studio can be described as having an explicit visual interface, dual representations of the function and the form of the design and computer-mediated manipulation of components.

simCHET on the other hand has a single multi-dimensional, multi-modal representation, notional icons and functionalities, unmediated manipulation of the relationships and reliance on user compliance with rules of interaction. Figure 7 shows an example of a full simCHET-designed applet. Note the similarities to (and differences with) the JavaStudio design for the same applet shown in Figure 5.

5.2.2 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 shown to be important; for example, pads of paper were needed to allow students to rough out preliminary Bean designs before attempting their final good version on PostIts.

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.

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

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 following 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 told 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.

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

Figure 8: PDG member collaborating with 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 Beans, collision Beans[8], and drawing Beans[9]. 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 Beans; these would allow for students to construct tools which could be shared across the network.

5.2.4 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" Bean[10] would be needed to properly control the flow of data. These were rapidly integrated into each group's design. The students gradually acquired the event-driven mindset necessary for a correct design.

[8] The Collision Bean would detect when two shapes had collided on the grid.

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

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

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 Bean and then acting out its function! Each student in the group acted as a JavaBean and performed the task required as an instantiation tool place. They were even able to invent a new JavaBean whose purpose was to disintegrate the applet once the instantiation was finished. Although this JavaBean performs a questionnably important function, 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.

6 Conclusions

The primary aim of this project was to affirm (or deny) 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.

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.

7 Future work

Research collaboration was very fruitful both for software designers and for students. 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" [10]. 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 has to be intuitive and supportive to use for the students without undermining its power and flexibility. In its current state, the technology is quite rudimentary and restricted. The programming language LOGO, which was developed for use by children provides 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 al-

lowing the user to write programs with pictures. The students themselves can play a crucial role in the refinement of PolyMath technology as it matures from an unstable professional tool to a stable child-friendly one. 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, without which this project could not have worked. We would also like to thank Zentra Computers for donations of PC hardware, Innovative Computing Solutions and Redesign for their networking support.

Suggestions and comments from Stefan Sinclair and Peter Taylor on the initial drafts of this article were greatly appreciated.

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] Ellen Balka, Nathalie Sinclair, Michael Jones, Loki Jorgenson, and Stephen Braham. Participatory design with island pacific school: Lessons from the field. In *submitted to The Fifth Biennial Participatory Design Conference*. 1998.

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

[4] 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.

[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] 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.

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

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

[12] Mitchel Resnick. Distributed constructionism. *Proceedings of the International Conference on the Learning Sciences*, July 1996.

[13] 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.