Title: Teaching Object Concepts For XML-Based Representations

Author(s): Robert L. Kelsey

Submitted to: 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications
              Nov. 4-8 2002
              Seattle, WA
ABSTRACT
Students learned about object-oriented design concepts and knowledge representation through the use of a set of toy blocks. The blocks represented a limited and focused domain of knowledge and one that was physical and tangible. The blocks helped the students to better visualize, communicate, and understand the domain of knowledge as well as how to perform object decomposition.

The blocks were further abstracted to an engineering design kit for water park design. This helped the students to work on techniques for abstraction and conceptualization. It also led the project from tangible exercises into software and programming exercises. Students employed XML to create object-based knowledge representations and Java to use the represented knowledge. The students developed and implemented software allowing a lay user to design and create their own water slide and then to take a simulated ride on their slide.

Categories and Subject Descriptors
[Educators' Symposium]: Experience paper

Keywords
object-based knowledge representation, XML, Java

1. INTRODUCTION
1.1 Setting and Participants
This teaching and learning experience took place during a Summer student program at Los Alamos National Laboratory. The Summer programs pair students with mentors to provide students with relevant research experience and on-the-job training. The research and training is designed to relate to and complement the student’s chosen field of study. In this particular case, three college students were being mentored by the author. The students were college sophomores majoring in computer science or engineering disciplines. They had both formal programming language instruction and some practical programming experience.

A lab Summer program is similar to other cooperative education programs. It is a work study type of program where students train and learn while contributing to actual programmatic tasks. The level of contribution (as well as training/learning) depends on both the mentor and student. In this environment, a student and mentor are colleagues. There is no classroom and no grading system. The student and mentor work together to complete a task.

1.2 Motivation
The intent of this particular training program, as described in the following sections, was to bring the students up to speed on object-oriented design concepts and applying those concepts to knowledge representation. Along the way the students learned about using XML and Java. These concepts and techniques laid a foundation for direct contribution to programmatic applications.

Although students were being trained and groomed for further contribution, this was only part of the intent. Students who did not return to subsequent Summer work would still have increased their skill set and experience.

1.3 Scenario
The training and contributing work was set up as a project. The students worked as a team, but at times with individual tasks. The choice of projects was narrowly focused because of the time frame, the potential work to be covered, and the unknown of how well the students would respond. The content and schedule had to be kept flexible. This was easier to do when mentoring a small group of students versus teaching to a classroom of students.

Additionally, the project needed to hold the interest of the students and be fun. It had to be more practical than theoretical and this fact had to be obvious to the students. At the same time, the students needed to understand they were contributing to something real and accomplishing something as well as learning.

Typically, mentor and students met twice a day. In the initial stages of the project these meetings were sometimes
quite long. As the students became more comfortable with the project, the mentor, and each other, the length of meetings dropped. Sometimes the mentor met with students individually. Students were free to request additional face-to-face time as necessary. They were also free to communicate by phone and email.

The students were encouraged to seek out and utilize other resources in the course of the project. Usually these resources consisted of books, the library, and the Internet. Sometimes the resources were other colleagues at the lab. Although there were bounds to the project (such as focus), the students had freedom to move about the bounds and test the waters. This helped teach and train the students about the research process as opposed to just the project content.

2. EXPERIENCE AND RESULTS

2.1 OOD, KR, And XML

Object-oriented design (OOD), concepts and techniques (in particular object decomposition and abstraction) can be used to model object-based knowledge. The eXtensible Markup Language (XML) [6] can in turn be used to create an object-based knowledge representation (KR) for an application domain of interest. XML-based representations are easy to create and use. These representations are particularly useful in application domains where the knowledge can be abstracted to objects and the knowledge is interchanged between application uses [3, 4].

An XML-based representation is created by creating a document type definition (DTD). This is the grammar for the markup language application. An instance of the DTD is created to populate the representation. A DTD then describes the set of classes while an instance describes the objects or instances of the classes. XML has constructs similar to the notion of an object and the properties and characteristics associated with an object.

These concepts for and relating OOD, KR, and XML to each other are the underlying principles that were taught to the students. The following sections describe the experience of teaching these concepts for a particular application domain. The students' acceptance to this teaching and learning approach, their mastery of the concepts, and lessons learned are discussed.

2.2 Child's Play

The choice of the application domain came from a building set type of toy. The toy is called Block-N-Roll by the Taurus Toy Corporation [5]. It is a set of plastic blocks of various types. The various blocks can be connected together to form a slide down which a marble can be rolled. This toy was abstracted to be an engineering design kit for designing water parks. Water park design became the application domain of interest for the student project.

The toy was not just the catalyst for the domain idea, but also the means for communicating the domain to the students. As an application domain the toy had a limited set of objects with a limited set of attributes. It provided a limited and focused domain. This was ideal for teaching object oriented design concepts.

The students were given the toy and told that it was an engineering design kit for designing water parks. This was the extent of the information given to them. They were told to play with the blocks, familiarize themselves with the set, and build water slides. The play time lasted about a day and a half. Interestingly, there was never any hesitation or complaints about being told to play with blocks.

Various slide designs appeared from this play time. Some of the slides were simple while others were quite complex. The students attempted to push the limits of the toy by creating designs that used every available block or used the blocks in a probably unintended manner. Some of the slides would never make it past the design stage as they would have been more dangerous than a trip over Niagara Falls.

2.3 Blocks To Concepts

After a couple of days of play time, discussions began on the basics of OOD concepts and object-based knowledge representation. Particular attention was paid to describing the different types of blocks by identifying properties and characteristics of each block. The act of trying to describe the blocks began with questions posed by the mentor. What is this block's shape? What is its color? What are its dimensions? What makes this block different from other blocks? What makes it similar? What makes the block different from other objects? As the students answered these questions they began to gain an understanding of object decomposition.

Care was taken to separate form from function when describing the blocks. Often, students were quick to talk about a particular block's function (in terms of a water park slide) when trying to identify characteristics and properties. The students were instructed that the form of an object refers to its characteristics and properties while the function refers to its methods and operations.

By the time these discussions were completed the students probably felt they knew the deep philosophy of blocks. This was an important point and lesson. When doing object-oriented design, whether for subsequent programming or knowledge representation, time and care must be spent. Time spent to create the correct representation will make life easier when the representation is to be used.

Eventually the students settled on their decomposition of the blocks. The different types of blocks included straights, curves, terminators, crossovers, whirlpools and supports. A general class of blocks evolved. Each of the different types of blocks could be described with an instance of the general class. A number of instances described a complete slide.

2.4 An XML-Based Representation

The students were taught the basics of XML including DTD design and use. The object concepts were mapped to XML constructs and an XML DTD was created for representing water slides. An (XML) element construct corresponds to an object, while an attribute construct corresponds to an attribute associated with an object. The general class of blocks became an element construct and the block properties and characteristics became attribute constructs associated with the element construct. At this point the students were
told to again play with the blocks and create slides, but to also record each slide in the XML-based representation.

The XML-based representation went through several iterations while the students exercised it with their block slides. Eventually, the representation stabilized with three different elements. The elements were slide, slidegroup, and segment. The slidegroup element contains slide elements. The slide element contains segment elements. The slide element has two attributes: number and direction. The number attribute is an identification number for uniquely identifying a slide. The direction attribute refers to the direction of the slide being defined.

The segment element has four attributes: number, type, cover, and support. A segment element represents a single block in the toy. The number attribute is an identification number for uniquely identifying a segment. The type attribute can have one of ten possible values: terminate, up, down, straight, curve.left, curve.right, cross.left, cross.right, whirlpool.left, and whirlpool.right. Type refers to and describes the segment (block). Slides begin and end with terminate type segments. Up and down type segments refer to ramps. The cross.left and cross.right types when put side by side construct a criss-cross intersection of slides. The whirlpool.left and whirlpool.right types are another kind of intersection where up to four slides converge on a whirlpool or drop-through. The straight, curve.left, and curve.right are self explanatory.

The cover attribute can have a value of either true or false. This refers to whether a segment has a cover (creating a tunnel) or is open. The support attribute can have a numeric value of zero or greater. This attribute refers to the number of blocks underneath a segment that structurally support the slide segment.

2.5 Graphic User Interface

An actual XML-based representation is an instance of the DTD. XML instances can be created with any text editor, although this requires an understanding of the DTD and its use. The students created instances with text editors, but were directed to develop an instance creation tool for lay users. In this case the tool would be for creating definitions of water slides. The students were given a crash course in user interface design and the use of Java and open-source Java libraries for manipulating XML. Collectively, the students had some experience with C++ , thus they picked up programming Java quickly.

The function of the graphic user interface (GUI) was to create an XML instance of the water park DTD. From the perspective of a lay user the GUI facilitates the creation of a slide representation. The students iterated on the design and implementation of the GUI with the help of surrogate users. Figure 1 shows the GUI for creating slide instances.

The GUI builds a slide one segment at a time. The radio buttons on the left side allow selection of the type of segment. The image in the middle of the GUI window is an overhead view of the type of segment currently selected. The cover selection menu sets whether or not a segment is covered. The Add Segment button adds the segment to the slide while the Delete Segment button deletes the last segment. When a slide is completed, the Export XML button will dump an XML instance of the slide to a file. The blank area in the GUI window is where an overhead view of the completed slide can be viewed.

2.6 Using The Knowledge

XML-based representations facilitate the use of the knowledge being represented [3]. There are a number of open-source XML parsers and other Java libraries available for manipulating and processing XML. The students made heavy use of these libraries while creating application uses of the represented knowledge of slides for water parks. Such uses included construction supply lists for slide construction, generation of the physical parameters of a slide, and environmental factors associated with a slide.

Many of the application uses could be classified as a format interchange application which is a common use for XML. For example, a high-level construction supply list might begin with a list of slide segments categorized by type. This information can easily be generated by converting the XML instance using XML Stylesheet Language (XSL) [7] and the Apache Xalan processor [1]. A style sheet expresses a format and processing map between an XML instance and a target result (format). Xalan is a processor which accepts the instance and style sheet and generates the result.

Other application uses can be achieved in the same manner. Given the physical dimensions of the different segment types, many physical parameters of a slide can be generated. These include total slide length, vertical drop, and ground area to contain a slide. Give a few more initial conditions (such as amount of water for a segment and weather conditions) it is possible to generate more complex information including the amount of water loss over time.

In another application the students were able to convert the slide instances to Scalable Vector Graphics (SVG) [8] format to make 2-dimensional drawings of slides. These drawings represented maps and blueprints of the slides in the water park. Apache Batik [2], an SVG viewer, was used to render and view the maps and blueprints.
Perhaps the biggest achievement for the students was to convert the slide instances into 3-dimensional representations that were simulated. As a result, not only was the slide viewable in three dimensions, but could also be simulated from the perspective of someone riding the slide. This was achieved by conversion to Virtual Reality Modeling Language (VRML) and then rendered with a VRML viewer. Figure 2 shows a VRML rendering of a simple slide.

![VRML rendering of a simple slide.](image)

**Figure 2:** VRML rendering of a simple slide.

### 3. CONCLUSIONS

Object concepts, such as object decomposition and object abstraction, can be more easily taught with physical and tangible objects. In this project a limited set of construction blocks was used to illustrate the notion of objects and their associated properties. The blocks were found to be an excellent medium for communicating the object concepts. More extensive exploration was achieved by abstracting to a real-world use as an engineering design kit. The students were accepting and at times enthusiastic to this approach to learning these concepts. Their understanding and retention of the concepts was apparent and exceeded expectations.

The project also demonstrated how XML can be used for object-based knowledge representation. The students were introduced to knowledge representation concepts and issues as part of this project. They employed Java and XML to create, populate, and use an object-based knowledge representation. Knowledge representation is not a typical subject area for undergraduate-level students, but this project was a useful vehicle for teaching about knowledge representation topics.

Whether or not this approach would be as successful in the classroom is unknown. The number of students in a classroom setting would obviously be more, but perhaps if the students were broken into small groups and each group was a project, then the approach might work. It would be interesting to identify other toys and domains that would work as well as this one. There may be other examples that would better fit in a classroom setting.

### 4. ACKNOWLEDGMENTS

The author would like to thank the students: Francisco DeMaria, Raven Rotsaert, and Stephen Seibert. The author would also like to thank Jane Riese and Ginger Young for their help.

### 5. REFERENCES


