# CFX: Finding Just the Right Examples for CS1

Dale Reed, Sam John, and Ryan Aviles
Department of Computer Science
851 S. Morgan St. (MC 152)
Chicago, IL 60607
(312) 413-9478
reed@uic.edu, sjohn@cs.uic.edu, ravile3@uic.edu

## ABSTRACT

Finding just the right example to answer a question can be difficult for CS1 students and teachers. For this to work well there must be both an intuitive interface, as well as an appropriate set of focused examples. These examples provide the scaffolding to enable students' discovery. CFX (*S*ee *F*or *E*xample) is such a system, providing an easy-to-use web based interface to a set of hand-crafted examples. A small pilot study showed that students using CFX could find answers to their programming questions in roughly half the time it took using a textbook. Freely distributed as open source software under the GNU General Public License [4], CFX can be used as a database authoring tool to capture a set of examples. Once the content is established, the interface can be automatically exported to a standalone dynamic interface, using static, non-database content. The development of this system has been sponsored in part by NSF's CCLI program, with content developed in conjunction with McGraw-Hill.

## Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features – *Control structures, Input/output, Procedures, functions, and subroutines, Classes and objects, Data types and structures*.

## General Terms

Algorithms, Design, Human Factors, Languages.

## Keywords

CS1, Programming Examples

## 1. INTRODUCTION

Einstein once said: "Example isn't another way to teach, it is the only way to teach." The problem in Computer Science is finding just the right example to answer a question. For this to work well there must be both an intuitive interface, as well as the right content. CFX (*S*ee *F*or *E*xample) is such a system, developed over the past 3 years sponsored in part by NSF's CCLI program, with content developed in conjunction with McGraw-Hill.

Recently a CS textbook publisher hosted a gathering of CS1 teachers to discuss what resources they would like to have available[*]. One of the top requests was to have a good set of examples to supplement their texts. While it is fairly easy to come up with an example, it is difficult to come up with just the right set of examples that pedagogically provide the scaffolding to enable students' discovery. Those examples are out there on the web, however sifting through all the examples to find them is difficult. This is especially true for CS1 students, as they lack the context to know what level of detail is meaningful. For instance, the Google search string "C++ Programming 'for loop'" yields about 21,000 hits. CFX provides a framework into which a good set of examples can be made readily usable.

## 2. THE PROBLEM AND THE SOLUTION

Having just the right set of examples easily available can be a problem from different perspectives. *Teachers* would like to have additional examples to accompany concepts presented in textbooks; *Students* can't easily access electronic versions of textbook sample programs. As part of discovery learning, students often can't find just the right example to answer programming questions; *Textbook authors and publishers*, short of creating a customized web page, it is difficult to place textbook examples into an intuitive web-based interface.

[*] Personal communication 5/13/2003 from Kelly Lowery, Editor, Computer Science and General Engineering, McGraw-Hill Higher Education, 1333 Burr Ridge Parkway, Burr Ridge, IL 60527.

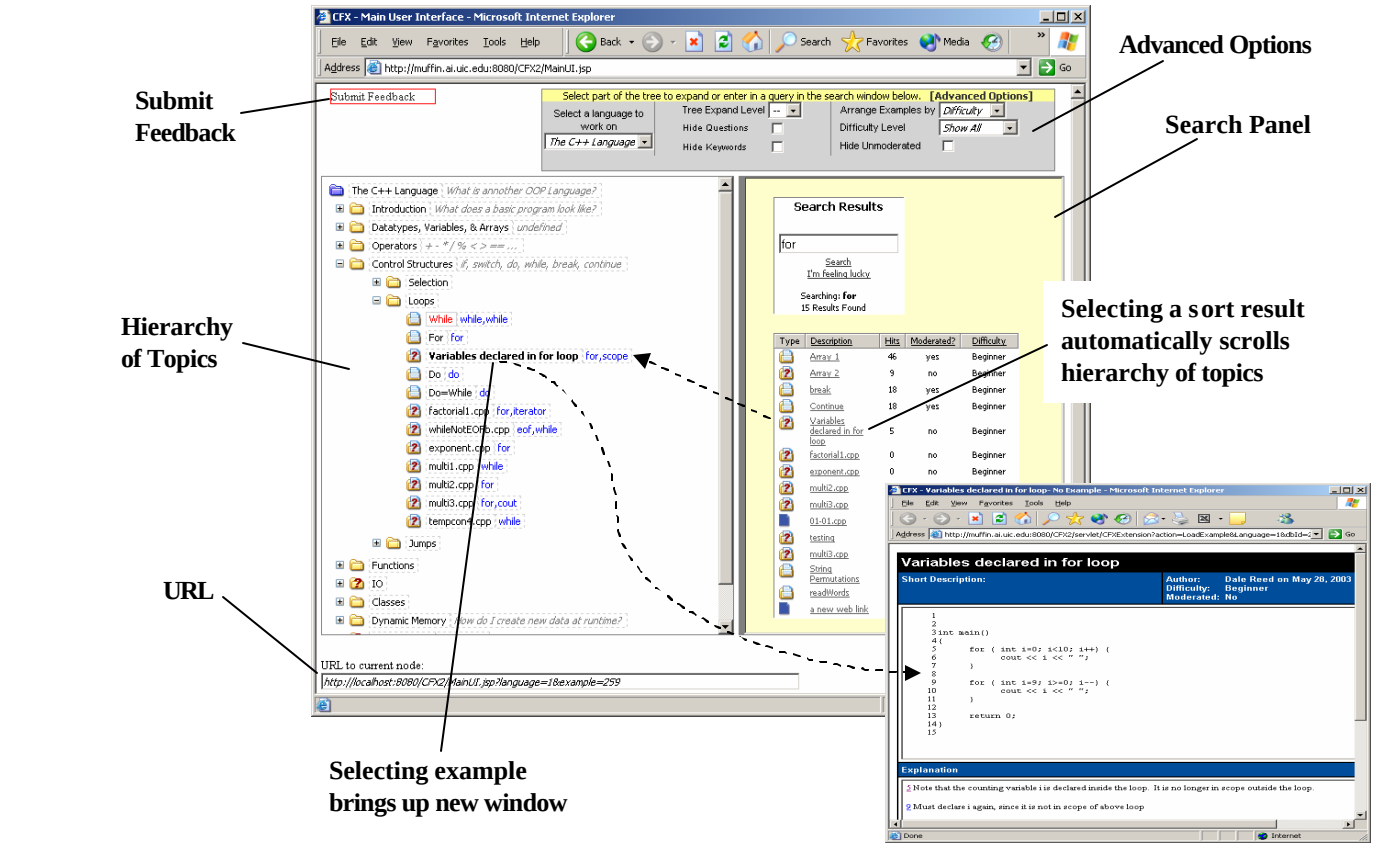


**Figure 1: Main Interface**

Textbook authors and publishers typically currently offer copies of their books' sample programs as zip archives on the books' web sites. The files inside the archive are themselves organized as a hierarchy of directories, each named after the corresponding chapter. Using these examples requires going back and forth between the textbook sections and the directory hierarchy, with no interface for a more intuitive search of these examples. CFX provides an easy to use tool to build the interface, dynamically creating the set of examples. CFX then gives the ability at the click of a button to then generate a dynamic interface (using Javascript and HTML) using static content. This gives a standalone interface with the functionality of a database search.

This can be added (as a single directory of HTML files) to the current textbook examples zip archive, providing students and teachers with an elegant keyword-searchable interface to rich content, while losing nothing from the existing presentation of examples. Students and teachers can then more easily browse through the textbook examples.

Finding just the right example is even more powerful from a discovery learning point of view when an example base of approximately 200 examples[†] is built in addition to those examples presented in a textbook. These additional examples could be incrementally added by the instructor or by the textbook publisher, as has been done with CFX for Java examples by McGraw-Hill. Working with an established textbook publisher gives the CFX example-base a wide audience, as McGraw-Hill hopes to sell 30,000 introductory Java textbooks worldwide this year.

## 3. THE CFX INTERFACE

In the main interface window shown in **Figure 1**, the hierarchy of topics on the left shows the main topics for the C++ Language. Topics are expanded and contracted simply by clicking on them. In the particular case shown in **Figure 1** the "Control Structures" topic has been expanded, and within "Control Structures" The "Loops" subtopic has been further expanded, showing 12 examples. Selecting one of these examples, for instance the 3rd one down titled "Variables declared in for loop" opens up a new window displaying this example. At

[†] The CodeLab system from Turing's Craft (http://www.turingscraft.com/) as of Spring 2002 has about 230 examples per language. Though perhaps coincidental, this number coincides with the author's experience of the number of examples that seems to be appropriate.

bottom left is the URL text box, which gives a URL to the current screen configuration. This allows an instructor to paste this URL as a link into a curriculum web page somewhere else, where selecting this link brings up the page as-is. On the right hand side is the search panel, where users can type in keywords, such as "for," or "output." The use of the search window is discussed later.

At upper left is the Submit Feedback button that allows users to submit questions that they were not able to answer using CFX. In the upper right we have the Advanced Options panel that controls what is displayed and how it is displayed. This is shown again in **Figure 2**.

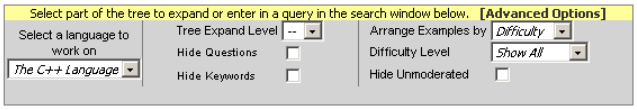


**Figure 2: Advanced Options**

The textbox at left in **Figure 2** allows selecting which language (or book or class) is to be used. The options in the middle column determine the appearance of the tree. An overall tree expand level can be chosen (none, All, or 1 through 9). Selecting "Hide Questions" hides the descriptions or questions in the outline, labeled in **Figure 3**. Similarly selecting "Hide Keywords" hides the keywords associated with each example. In the right-hand column shown in **Figure 4** we can arrange the results of a keyword search either by "Difficulty" or "Popularity". Difficulty is selected when examples are put into the system, while Popularity is determined over time through use of the system. A user can select whether or not more difficult examples are even shown, as well as choosing whether or not un-moderated results are shown. This is necessary since users can submit examples of their own.

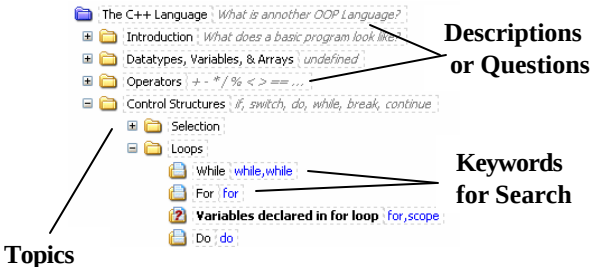


**Figure 3: Topics Display Components**

Choosing to search on the keyword *for* brings up the search results as shown in **Figure 4**. Selecting an example listed as a search result automatically opens and scrolls the topics hierarchy in the left hand panel to the corresponding location and highlights it, so that example can be opened. Scrolling the tree in this way has the advantage of showing related content. The results in the search panel itself can be sorted by clicking on a column heading, showing the results either in alphabetical order, order by popularity (number of hits), in order by moderated/un-moderated, or by difficulty level.

Although screen shots are not shown here, the administrator's interface is used to add examples, modify existing examples and topics, and create entire new hierarchies of topics. Of particular interest is the ability to generate a static version of the interface, as mentioned earlier.

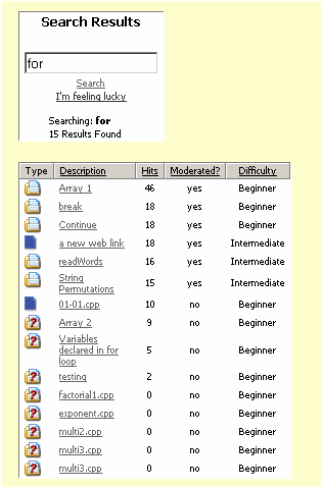


**Figure 4: Keyword Search Results Sorted by Hits**

**Figure 5** shows a small snapshot of the tree-editing interface, highlighting the Export portion that allows generating the static version. Note the interface is built using a client-server system with a database on the back end, however the generated version is then stand-alone on the client, without a server or database required. The static version is identical to the dynamic client-server version, with the exception of the server-side elements (# of hits, submit feedback button, URL textbox). Being able to generate a static version means that it can be listed on textbooks' home pages, as well as included with a CD or diskette that often accompanies introductory programming texts.

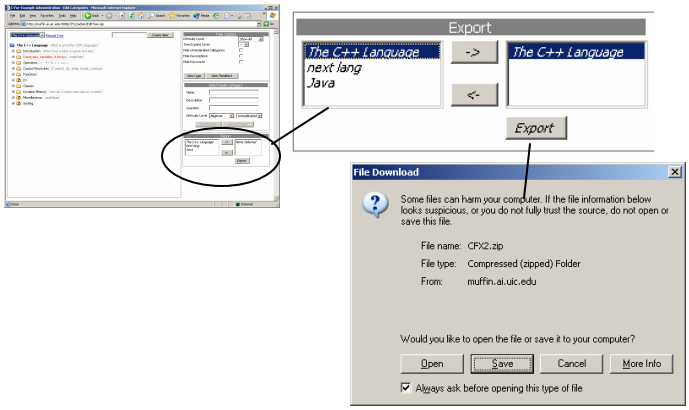


**Figure 5: Exporting a Static Version**

# 4. RESEARCH BASIS: INTELLECTUAL SCAFFOLDING FOR DISCOVERY LEARNING

Rapidly changing technology as well as research in Cognitive Psychology over the last two decades has given impetus to a pedagogical shift away from transmitting a body of expected knowledge to a more process-oriented approach [2]. There has been a move from the behavioral approach of direct instruction and explicit teaching [7] to the cognitive approach of collaborative [1] and discovery learning.

Vygotsky proposed that all learning takes place in the 'zone of proximal development', where this 'zone' is the difference between what a student can do alone and what he/she can do with assistance. By building on the student's experiences and providing moderately challenging tasks, teachers can provide the intellectual scaffolding to help students learn. Students are more to remember and have a deep understanding of concepts they discover on their own.

Discovery learning is "an approach to instruction through which students interact with their environment by exploring and manipulating objects, wrestling with questions and controversies, or performing experiments.[5]" For example, students using this approach to learn computer programming would be given an assignment currently beyond their programming skills (the 'leading question'). Students would then ask questions (e.g. how can I display a two-dimensional board on the screen?) which an expert could answer using carefully constructed examples. These examples give the student the information necessary to construct a mental model leading to the solution to the problem.

The examples give enough information to answer the student's questions, yet don't trivially give the entire solution. Resolving this 'cognitive dissonance' is what leads students to incorporate learned concepts into a contextualized framework in long-term memory. Choosing a set of carefully crafted examples can help overcome the cognitive bottleneck where too few students negotiate the transition from lower-level skills to higher-level skills, from basic rote memorization and re-telling skills to more abstract knowledge synthesizing and transforming skills.

For this to be successful within the constraints of a traditional undergraduate introductory programming course the inquiry must be guided, and expertise must be readily available to answer students' questions by giving appropriate examples. In a large programming class (e.g. over 100 students) it is not feasible for the instructor and the teacher's assistants to provide detailed answers and examples to each students' questions. We can, however, capture this expertise and make it available as an exhaustive set of carefully constructed examples, accessible through the Internet.

Discovery learning is most successful when students have such prerequisite knowledge and undergo some structured experiences. This can come from a textbook as well as from classroom lectures. This baseline of information gives students the vocabulary to ask questions, looking for examples to answer these questions. In even moderately sized classes (40 students) there isn't enough time during class, lab, and office hours for everyone to have their questions answered. Searches on the web often are ineffective for the beginning programmer since they lack the perspective to hone their searches. (e.g. Searching in Google for "C Programming Examples" yields many tutorials, but not necessarily sample programs.)

Another way to think of this is Robert Gagne's hierarchy of intellectual skills [3], where recognizing is a lower-order order skill than is higher-order rule application. The right web-based application can instrument a user's action by giving higher-order performance while using only a lower-order recognition skill.

## 4.1 Advantages of Web-based Examples vs. Print

The *average* student is often frustrated in trying to find answers in the textbook because after one or two basic examples for each concept, the rest of the information in the book is described textually and not given as examples due to limitations of space. (The *above-average* student *can* handle this level of abstraction.) Texts are meant as a jumping-off point. Reference manuals, on the other hand, require a level of mastery of the subject material usually not present until *after* the course is over (or even several courses). Textbooks that try to cover both bases end up being encyclopedic, where (in the author's experience) the size of the book is a disincentive to students to read it. This leaves a gap between basic programming texts and reference manuals, where average students try to get their questions answered by asking other people and looking for appropriate examples.

Presenting a complete set of examples on the web has several advantages over trying to do this in print. On the web any number of example programs can be given, organized into multiple layers of presentation. These examples can be cross-indexed according to concept, difficulty level, and related examples, as well as by keyword. This organization allows a user to get questions answered and explore in a self-paced asynchronous fashion. This can be done anytime & anywhere there is a network connection, exploring concepts to varying degrees of depth.

## 4.2 Comparison of CFX to Textbook Search

A study was done to determine the effectiveness of CFX on students' ability to find examples. There were two methods used by the 9 students (an admittedly small sample), where the students were timed as to how long it would take for them to find examples for 7 different CS1 topics. The 7 topics were The

seven topics were: do, break, eof, function parameter, random, dynamic array, linked-list. The traditional method was to have students user their textbook, accompanied by an on-line version of the textbook examples. Note that publishers organize these examples by chapter, so students must use the book as a starting point unless they want to randomly search. The second method used was CFX. Care was taken to make sure each topic could be readily found in both the CFX C++ example base as well as in the programming textbook index. 7 of the 9 students were CS1 students, the other two were CS2 students.

On average, examples were found in 1 minute using CFX, while it took 1 minute 54 seconds (nearly twice the time) for examples to be found using the textbook. Students were given minimal instructions, and found CFX to be intuitive and readily accessible.

## 5. TECHNICAL DETAILS

The system currently requires a client's use of Internet Explorer (IE) browser for full functionality. On the server, the Tomcat Java servlet engine is used to talk to a mySQL database. The advantage of running with these components is that they are *completely free*! CFX is freely distributed as open source software under the GNU General Public License [4]. We have installed and run CFX on both Windows and Linux platforms, using a wide range of processor speeds and memory configurations. For a windows server a minimum of 384 MB RAM is recommended, with a minimum processor speed of 500 MHz.

## 6. DOWNLOADS AND ONLINE ACCESS

The source code and installation instructions can be downloaded from http://logos.cs.uic.edu/cfx with a site reserved for open-source distribution at http://cfx.sourceforge.net. CFX can be accessed online for Java examples at:

http://synapse.ai.uic.edu:8080/CFX2/MainUI.jsp

and for C++ examples at:

http://muffin.ai.uic.edu:8080/CFX2/ MainUI.jsp

## 7. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation Award #0127299. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The Java examples content development and some interface design was sponsored by McGraw-Hill.

## 8. REFERENCES

[ 1] Arends, R.I. (1994). Learning to Teach. (3rd ed.), New York, NY:McGraw Hill, Inc (ch. 11).

[2] http://copland.udel.edu/~jconway/EDST666.htm, 6/5/2001.

[3] http://education.indiana.edu/~p540/webcourse/gagne.html

[4] http://www.gnu.org/licenses/gpl.txt

[5] Ormrod, J. (1995). Educational Psychology: Principles and Applications. Englewood Cliffs, NJ:Prentice-Hall.

[6] Roblyer, Edwards, and Havriluk, M.D., Edwards, Jack, & Havriluk, Mary Anne (1997). Integrating Educational Technology into Teaching. Merrill, Upper Saddle River, NJ.

[7] Rosenshine, B. (1986). Synthesis of research on explicit teaching, Educational Leadership, April issue, pp. 60 – 69.