On Teaching Programming Languages Using a Wiki

Iliano Cervesato
Carnegie Mellon University – Qatar

iliano@cmu.edu

Exploring ICT in Education, 2010

25 March 2010
CMU-CS 15-212
“Principles of Programming”

- Sophomore-level CS course
  - Advanced programming concepts and skills
  - Introduced in the early 1990s

- Now, little supporting material
  - Notes taken in class
  - Few handouts
  - Code posted on the web page
  - No book!
    - Out-of-print or obsolete

- A challenge for many students

Put the material on a wiki
What’s a wiki, again?

…the stuff of Wikipedia

- Collaborative framework to create (and share) information
  - Simple, transparent editing
    - Supports text, images, math, sounds, …
  - Topic-oriented
    - Short articles (compared to book chapter)
    - Related topics accessible via links
  - Collaborative
    - Everybody can be an author
    - Mechanisms to avoid abuse
The 15-212 Wiki

- Put the whole course on a wiki
  - Categorizes and cross-references material
    - Detailed explanations of material covered in class
    - Lots of examples, exercises
    - Further readings
    - Pointers to advanced material
  - 60% so far (Nov 07 - Mar 10)
  - Built on MediaWiki (same as Wikipedia)

- Semi-open for editing
  - 15-212Q staff and students
    - Create, correct, improve articles
  - Experts, instructors, students elsewhere
    - Upon authorization

Still preliminary
Sample Code

```plaintext
// Another Example: Reversing a List

val reverse : 'a list -> 'a list
  returns a list consisting of the elements of l in reverse order.
  Invariants: none
  Effects: none

fun reverse (nil, 'a list) : 'a list = nil
| reverse (x::l) = reverse l @ [x]

Because the append function recurses on its first argument, the recursive call is extremely inefficient. Observe the following evaluation trace:

reverse [3,4,5] = reverse [[4,5], 3] = reverse [[5], 4] = [[5], 4] = reverse [] = nil

The unbinding of reverse leaves an expression to be evaluate that contains 4 calls to append, such as the list argument of the last Evaluating this expression produces the leftmost element of the resulting list (there 5) over
and over. Instead, it can be shown that the reversal of a list of n elements involves the creation of n+1/2 list
```

Graphics

Prose

Formulas
Wiki-Based Instruction

- Not just a surrogate for a book!
- A comprehensive didactic tool
  - Promotes participatory learning
    - Students are “encouraged” to modify articles
      - For play, for curiosity, or for points
      - Get them to research topics
      - Explain ideas to others (in writing)
    - Active participants in the didactic process
  - Easy monitoring of students’ involvement
    - Every edit is logged
      - We know who did what
    - Every access is logged
      - Make sure students read the material before class
        » More interactive and focused in-class discussion
        » No “guest account”
Actual Experiments

- 1 recitation on wiki editing
- 20% of 1st assignment
  - Objective 1: play with the mechanics of the wiki
  - Objective 2: test research/creativity

Problem 4: The 15-212 Wiki [10 Points] Due by Tuesday, January 22nd

This part (and only this part) will be done with a partner. At recitation 2 on January Sunday 20th, you will be introduced to the 15-212 wiki. As an assignment, you will have to write a wiki page on a given topic. One of these topics will be assigned to you:

- **Strings and Boolean**: You will be to write something about String and Boolean in the same way that it has already done for Integers and Reals.

- **The Standard ML Basis Libraries**: You will be to introduce what are the ML basis libraries and briefly describe the most useful ML libraries.

- **SML Top Level Environment**: You will be to explain what is natively defined in the SML Top Level Environment.

- **ML Compilers**: Your work will be to write a short summary about the other existing ML compilers and their differences with SML/NJ.

8 students

- Group A
- Group B
- Group C
- Group D
Group A

- Students followed models rather accurately
  - Took good advantage of wiki
  - Good starting point

- Some humor

- Little creativity
  - Shy to experiment
Group A (2)

- Same structure
  - Was partially given

- Lot less prose
  - Ran out of time?
    - 2nd week of class
Group B

- These students looked up the material and reported on what they found
  - No elaboration
  - Not integrated within wiki
  - Limited use to other students

- Could have written more
Group C

- Students did research and found several relevant documents
  - Good analysis/synthesis work
  - Combine prose and code as appropriate
  - Reference sources

- Lots of humor and creativity
  - Text at the top of the page
  - Cartoon

- Best result
Group D

➢ Students did a search on the web and dumped what they found
  ▪ Little post-processing
  ▪ No creativity, no fun

➢ I was not too happy about this one

ML compilers

[edit] Standard ML of New Jersey
Also known as SML/NJ®, it is an optimizing native-code compiler for Standard ML that is written in Standard ML. It runs on a wide range of architectures. It was developed jointly by Bell Laboratories and Princeton University.

[edit] Moscow ML
Moscow SML® is particularly suited for teaching and experimentation, where fast compilation and modest storage consumption are more important than fast program execution. Thanks to the efficient run-time system of Caml® Light, Moscow SML compiles fast and uses little memory. Moscow ML was created by Sergey Romanenko at the Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, Claudio Russo® at Cambridge University, UK. Niko Korkman® at the IT University of Copenhagen (Moscow ML for .Net), Kim Friis Larsen® at the IT University of Copenhagen, and Peter Sestoft® at the Royal Veterinary and Agricultural University, Copenhagen, Denmark.

[edit] MLton
MLton® aims to produce fast executables, and to encourage rapid prototyping and modular programming by eliminating performance penalties often associated with the use of high-level language features. It was developed by Sureh Jagannathan®, Stephen Weeks®, Matthew Fluet® and Henry Cejlin®, who is also the first employee of Mathworks®.

[edit] PolyML
PolyML® was originally written by David Mathews® at the Computer Laboratory at Cambridge University.

[edit] TILT
TILT® is a self-checking compiler for Standard ML that uses Typed Intermediate Languages. TILT is under active development as part of the ComCert Project® at Carnegie Mellon. TILT began as a joint effort between researchers at ComCert® and in the Fox Project® at Carnegie Mellon to develop a successor to the TIL Compiler, a self-checking compiler for the Standard ML core language.

[edit] The ML Kit
The ML Kit® compiler covers all of Standard ML, as defined in the 1997 edition of the Definition of Standard ML. The ML Kit implements most of the latest Standard ML Basis Library specification. The people involved in The ML Kit can be found on this page®.

[edit] SML.NET
SML.NET® is a compiler for the functional programming language Standard ML that targets the .NET Common Language Runtime and which supports language interoperability features for easy access to .NET libraries. The people involved in SML.NET can be found on this page®.
Outcome Summary

- **Objective 1: play with mechanics of wiki**
  - All figured out the basics
  - Some did a little extra
  - None did more than expected

- **Objective 2: test research/creativity**
  - Research varied from vigorous web search to minimum needed to adapting example
  - Creativity ranged from the dull to the surprising

- **Altogether**
  - Students found exercise fun
    - Novelty
    - Not usual routine
The Wiki and 212’s Future

- **Categorization helps focus on the big picture**
  - What the course is about
    - Problem solving, not ML

- **Encourages rearranging material and delivery**
  - Experiment with what works best
    - Case-studies, examples, exercises
    - Interplay between problem solving and programming

- **Encourage exploring syllabus improvements**
  - Add/remove topics
  - Change language

- **A more dynamic course**
Future Developments

- Add rest of course material
  - ... my summer project
  - Continuous improvement cycle
- Extend with “try-it” capability
  - Sandbox interpreter within wiki
- Pair-up exercises with e-tutor
  - Intelligent learning system
- Explore opportunistic learning ...
Opportunistic Learning

- Books, notes often modeled after lecture
  - 1+ hour long
    - Long attention span
  - Mostly linear presentation of material
    - This is not necessary and probably not efficient

- Wiki breaks away from this model
  - Brief topic-oriented articles linked together
    - Smaller time granularity for studying/reviewing (5-15 min)
      - Harness “dead times” (commuting, time between classes, …)
    - Focus on actual dependencies
  - Make it mobile
    - Reformat wiki for viewing on PDAs, smartphones
    - Add matching video segments of lecture, slides, …