

A Programming Languages Course at Web Speed

Art Gittleman
Computer Engineering and Computer Science Department
California State University Long Beach
Long Beach, CA, U.S.A.
artg@csulb.edu

Abstract - The immediate Internet availability of new developments in programming languages allows the core survey of programming languages course to illustrate language concepts with examples that bring students to the forefront of current practice. Ruby and C# each solve problems of current interest in an innovative way that will influence future languages. They are different enough to provide good contrasts and yet their commonalities may show the future direction of productive languages. We describe and provide materials for a survey of programming languages course with examples from Ruby and C#.

Keywords: Programming languages, Ruby, C#, course

1. Introduction

We have all heard that software developers now work at "Internet speed" where world-wide connectivity pushes the pace of new releases. The Web has transformed the lives not only of developers but of the many millions of computer and device users. The growth of Google highlights the value of information and the speed with which it can be disseminated. Unlike Mathematics which changes very little at the undergraduate level, Computer Science is continually evolving. The core Computer Science concepts may be fairly stable, but the environment changes rapidly. Computer Science textbooks often refer to outdated examples even when these examples illustrate the concepts we want to teach. Computer Science enrollments have been declining dramatically. We need to ensure that our students are ready for the future with its fast Web pace.

The comparative programming languages course has been a mainstay of the core curriculum. This paper will show how to adapt the programming languages course to prepare students for the speedier Web world while teaching the concepts of programming languages. The first part will show why Ruby and C# are good choices to be the main languages compared. The second part will look at how to teach such a course, including resources, teaching materials, and assignments.

2. Why Ruby and C#

Java and C++ are the two main languages taught today in the CS I and CS II courses. Students then use these languages throughout the curriculum. The high-school advanced placement exam uses Java. In industry server-side Java has been particularly popular. In *Beyond Java* [1], Bruce Tate argues that Java is catering to the needs of large enterprise applications whose requirements are extremely heavy. The many developers who write applications that interact with a database via a user interface do not need the complexity of enterprise Java. They do not need to struggle with development frameworks that have become very complex.

Tate presents Ruby as a language that combines power and clarity of expression to simply the developer's task. The article "Ruby the Rival" [2] makes some of the same points. Paul Graham, developer and entrepreneur, has written many interesting essays. [3]. He and his two associates programmed Viaweb (later sold to Yahoo) in Lisp. In "Revenge of the Nerds," [4], Graham says that the trend in programming languages is to be more like Lisp. He illustrated with the progression from Java to Perl to Python. Ruby follows this trend. It is dynamically typed with powerful features such as function closures and continuations. It facilitates metaprogramming where one can manipulate code at runtime. David Heinemeier Hansson created the Ruby on Rails web development framework [5] which leverages the power and flexibility of Ruby to enable developers to be vastly more productive. He lists metaprogramming as one of the most important of Ruby's capabilities, calling it a "framework builder's dream." [1, p. 105].

Ruby, a fully object-oriented language, was created in 1993 in Japan by Yukihiro Matsumoto. It is freely available available for Unix, Linux, Mac, and Windows platforms [6]. *Programming Ruby*, [18], by Dave Thomas is the best known of the current Ruby books. The first edition is available online [7]. Among Ruby's interesting features are blocks, iterators, closures, continuations, duck typing, and metaprogramming.

Continuations can serve to build continuation Web servers which are ideal to manage the state of complex Web applications. HTTP is stateless so Web applications have to solve the problem of maintaining the state between requests of the same client. Complicated Web frameworks address this problem. The controller must jump among the various pages. A change of mind by the user or the starting of a second client by the user create additional difficulties. A continuation is a construct that saves the state and location where a computation should continue. With a continuation based Web server one can write a Web application like an ordinary method. Continuations will manage the jumps from page to page without the developer needing to deal with them explicitly. The best known continuation web server, Seaside, is written in Smalltalk, but there are some ports to Ruby, including Borges [8].

Ruby is rapidly increasing in use with a number of significant deployed applications. It is an interesting language that Matz (Matsumoto's nickname) created to minimize his effort in programming [9]. The Rails framework uses Ruby's capabilities to make Web development vastly more productive. Ruby can be one source of examples for a programming languages course that gets students on the fast track to success.

Another other language that solves a real problem for developers is C#. Version 2.0 added some interesting features to the original 1.0, but the 3.0 version previewed last September at the Professional Developers Conference really shows the influence of the programming language researchers that Microsoft has hired, many from academia. The new features in C# 3.0 [10] are designed to support the creation and use of higher order, functional style class libraries. In particular C# 3.0 supports LINQ, Language Integrated Query. Databases store data in relational tables while programming languages use objects. Using a programming language to access a database causes complex conversion problems. Similarly the XML format does not match programming languages types very well. LINQ allows query of in-memory collections as well as databases and XML files with a facility and clarity not heretofore available [11].

LINQ provides standard query operators with a friendly syntax supported in C# by the new version 3.0 features. Lambda expressions provide a compact syntax and using expression trees they can be compiled to data as well as code. These expressions can be processed at runtime, for example to generate SQL to access a database. The standard query operators are defined as extension methods, another C# 3.0 feature, which allow

third parties to augment the contract of a type. Other C# 3.0 additions used in LINQ include implicitly typed variables, object initializers, and anonymous types.

Web frameworks effectively use the expressive Ruby language. Language integrated query integrates XML and database querying into the C# language, reducing the complexity of accessing information. The next section outlines a programming languages course which takes its examples from Ruby and C# so that students learn programming language concepts in a context that will guide them to future success. While C# was chosen for this course, Visual Basic .NET would also have been an excellent choice. A Visual Basic 9.0 preview is available and contains the language extensions necessary for LINQ (see [14]) as does C# 3.0.

3. Getting Up to Speed with Ruby and C#

The site for the Spring 2006 programming languages course [12] at California State University Long Beach (CSULB) has links to resources for Ruby and C#. Installing Ruby is very easy, requiring just one click for Windows for example. Visual C# Express 2005 is a free download as is the LINQ preview. The LINQ site [14] provides downloads of the C# 3.0 preview, LINQ Hands-on Labs, 101 Samples using LINQ, and many other resources. No textbook is required because the Web provides an abundance of tutorials, code samples, specifications, and articles. The entire C# specification, versions 1.0, 2.0, and 3.0, is online, as is Ruby documentation. Students are not held back by dated materials, but rather can look forward to the frontiers of programming languages in use.

The senior-level programming languages course at CSULB meets for 15 weeks with two lecture hours and three lab hours each week. In many CSULB courses with labs students have the option of coming to the lab or doing the projects on their own time. Projects may extend over several weeks. Students have good intentions but can easily get behind and have to drop the course. Because the goal of assignments in this course is to compare languages and not to develop systems the programs in this course can be brief. Thus the Spring 2006 programming languages course had 22 lab assignments, comprising almost one for each one and one-half hour lab period. Students all come to the lab where they are allowed to discuss and collaborate to facilitate learning. These lab assignments and sample Ruby and C# programs are posted on the course site [12].

This is the first time the programming languages course has been taught in this format so there will doubtless be some revisions based on experience. When feasible an assignment requires comparing C# and Ruby. When that would be too complex for one assignment, separate Ruby and C# assignments cover a concept and the comparisons are made in class. Working with students in labs adds aids in learning because the instructor is at hand to answer questions, but this approach could work well in lecture-only formats too. For example weekly assignments could group two of these lab assignments. Some of the extra lecture hours available in a lecture-only format could be used to discuss the objectives of the labs and to interpret the results.

Both C# and Ruby are new to students in this course. The core of C# is similar to C++ and Java so advanced students can easily handle the basic constructs. The course includes some C# 1.0 features such as properties, indexers, delegates, events, and attributes for which students may not have seen a close analog in languages they know. The mostly unfamiliar C# 2.0 additions, namely anonymous methods, iterators, generics, and nullable types need to be introduced as do the 3.0 features previously mentioned. Finally the goal is to present language integrated query illustrated with in-memory, database, and XML examples.

Students have not used a language as flexible as Ruby. Early examples show that an object's type is defined by what it can do. Ruby modules provide namespaces and implement the mixin capability where the module methods are mixed into the class. The Comparable and Enumerable modules from the Ruby core provide great examples of how to define core functionality and reuse it often. The C# contrast comprises namespaces and interfaces.

Ruby blocks, iterators, and procedure objects provide a concise way to process collections. The iterator concentrates on generating the values while the block holds the code to process the values. Control transfers between the iterator and the block by means of a yield statement. C# iterators provide a nice comparison. In C# 1.0 one must explicitly implement the IEnumerable interface, but C# 2.0 adds the *yield return* statement which produces the next value of the iteration. Both Ruby and C# are able to create function closures. One assignment requires the conversion to C# of a call-by-need Ruby example implemented with a function closure.

Ruby continuations take some effort to grasp but the course site has 19 examples to assist students. These include factorial, binary search, producer-consumer, coroutines, and a poor man's Seaside program which, using the Webrick server bundled with Ruby, illustrates how a continuation server works. Rails, although not a continuation server, is garnering increasing interest because of its ability to simplify web development. Fortunately two long online articles give an excellent introduction to its use [13]. Underneath, it is the dynamic nature of Ruby that makes Rails possible. As an example of what can be done with Ruby the course includes one Rails project.

After assignments that introduce C# 2.0 and 3.0, the C# study culminates with query expressions, the DLINQ extension for databases, and the XLINQ extension to query XML. The LINQ resources provide excellent hands-on labs and literally hundreds of examples [14]. Here students have an opportunity to learn language concepts that started out in research languages such as Omega [15] or have been adapted from functional languages like Haskell, as for example the generic operations on collections based on monads [16], [17]. But they learn them in C#, a production language that they might be able to use in industry.

The C# 2.0 assignments cover anonymous methods, iterators, and generics. The language additions in C# 3.0 support language integrated query. Three assignments illustrate these C# 3.0 features and the final three demonstrate language query expressions with application to databases and XML. The query operators are almost all C# 3.0 extension methods of the type *IEnumerable<T>*. C# 3.0 lambda expressions can be passed as arguments to query operators. Lambda expressions can be compiled to expression trees which may then be processed like any other data structure. This helps in generating efficient database queries. C# 3.0 object initializers allow the construction of new structured values in contexts where only expressions are allowed. C# 3.0 anonymous types are often used in the *select* clause of a query. The C# 3.0 Language Enhancements Hands-on Labs and the LINQ Hands-on Labs, [14], provide excellent examples to cover both C# 3.0 and its use in language integrated query.

4. Conclusion

Many of our students seek employment after receiving the B.S. if not already employed in the profession. Students completing this programming languages course have worked

very hard to complete 22 assignments but have deepened their knowledge of programming languages with forward looking examples that solve pressing current problems. At this writing the course is in progress, So far students are outperforming students from past semesters. Both interest and effort have increased. From the Web we learn of new language developments as they appear and we can obtain teaching resources to keep our courses current so students will be up to (Web) speed.

5. References

1. *Beyond Java*, Bruce A. Tate, O'Reilly, 2005.
2. "Ruby the Rival," Chris Adamson, <http://www.onjava.com/pub/a/onjava/2005/11/16/ruby-the-rival.html?page=1>, 11/16/2005
3. <http://www.paulgraham.com/articles.html>
4. "Revenge of the Nerds, " Paul Graham, <http://www.paulgraham.com/icad.html>, May, 2002
5. <http://www.rubyonrails.org/>
6. <http://www.ruby-lang.org/en/20020102.html>
7. <http://www.rubycentral.com/book/>
8. <http://borges.rubyforge.org/>
9. See the four interviews with Yukihiro Matsumoto found at <http://www.artima.com/intv/ruby.html>
10. <http://msdn.microsoft.com/vcsharp/future/default.aspx>
11. "Comparing LINQ and Its Contemporaries," Ted Neward, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/linqcomparisons.asp>, December, 2005
12. <http://www.cecs.csulb.edu/~artg/424/index.html>
13. "Ruby on Rails, Part 2," Curt Hibbs, <http://www.onlamp.com/pub/a/onlamp/2005/03/03/rails.html>, 3/03/2005.
14. <http://msdn.microsoft.com/netframework/future/linq/>
15. <http://research.microsoft.com/Comega/>
16. "XLinQ: XML Programming Refactored (The Return Of The Monoids)," Erik Meijer and Brian Beckman, <http://research.microsoft.com/~emeijer/Papers/XMLRefactored.html>
17. "Confessions Of A Used Programming Language Salesman: Getting the Masses Hooked on Haskell," Erik Meijer, <http://research.microsoft.com/~emeijer/Papers/ICFP06.pdf>
18. *Programming in Ruby*, Dave Thomas with Chad Fowler and Any Hunt, The Pragmatic Programmeers, 2005.