Nancy C. Wilson. The Benefits of Intersecting Foreign and Programming Language Acquisition Pedagogical Methods. A Master's Paper for the M.S. in I.S. degree. April, 2006. 37 pages. Advisor: Stephanie W. Haas

This study describes an interview survey of college instructors of foreign and programming languages as well as a content analysis of textbooks from these fields. Seven interviews were conducted with instructors in Romance Languages and Computer Science at five colleges in central North Carolina. The purpose of the interviews was to determine how instructors of foreign and programming languages view their teaching methodology and how this relates to the textbooks they choose.

Based on the information gathered at the interviews and a subsequent content analysis of six textbooks, this study explored the possibilities that exchanging teaching ideas between foreign and programming language texts might afford if these were applied creatively across these two curricula.

Headings:

Programming Languages – Teaching

Foreign Language – Teaching and Learning

Education Literature – Evaluation / Content Analysis

THE BENEFITS OF INTERSECTING FOREIGN AND PROGRAMMING LANGUAGE ACQUISITION PEDAGOGICAL METHODS

by
Nancy C. Wilson

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2006

Approved by

_____

Stephanie W. Haas

Table of Contents

I. Introduction

While some may argue that programming and foreign languages have little in common given the logical, procedural nature of the former and the emotive, creative quality of the latter, there are natural grammatical and syntactical similarities difficult to deny. As forms of communication, programming and foreign languages provide a means of shaping or translating ideas into some form of expression that will be understood by the reader or listener. Besides their shared communicative purposes, both language domains also share culturally-defined notions of correct and incorrect (or less accepted) forms of expression. For example, some cultural dialects of French, such as the Cajun French used in Louisiana, accept different vocabulary than their Parisian counterpart; as for programming "dialects," some "best practices" for labeling variables may vary depending on the programming environment and language used. For example, if global variables are turned on or off in the programming environment, this will affect the accepted forms of expression used when defining variables. Finally, the process of learning more than one programming or foreign language is similar, since one must first master a "native" language and then apply known concepts to learning a new form of expression. Given their potentially similar learning processes, we might consider how the teaching methodologies of foreign languages and programming languages overlap. The techniques and methods used by foreign languages (which have a much longer history) potentially could be applied to their programming counterparts, or vice versa.

As foreign language teaching methodologies have evolved and divided into several "camps" or ideologies about what is best for learning, theories on how programming concepts and syntax should be taught are less solidified. Programming textbooks are often aimed at readers with a certain level of programming experience in another language, but they are not generally self-described as being centered on a particular teaching philosophy. Foreign language textbooks are often created based on a particular methodology or pedagogy. Despite these differences, are there similarities in the goals of instruction in these two domains? Are there similarities between the current practices and teaching methodologies used? If so, what can programming language and foreign language teaching methodologies offer each other?

II. Literature Review

In their 1995 article entitled, “The Application of Second Language Acquisition Pedagogy to the teaching of Programming Languages – A Research Agenda,” Robertson and Lee discuss the historical developments of second natural language teaching and programming language teaching, concluding that more research is needed to test how successful teaching methods for foreign languages might be applied to the teaching of programming languages. As Robertson and Lee note, there has been a shift in natural language teaching methods away from the traditional grammatical approach toward greater attention on communication. Language learning is a journey that we begin early and continue throughout life. In order to build new vocabulary and domain knowledge, children acquire their mother tongue by using it and hearing it used in various situations. This is the philosophy driving what is known as the “communicative approach” in second-language learning. The communicative approach method involves teaching in

context using only the target language and therefore imitating the child's first-language learning experience. This method assumes that one can learn a language through visual cues that indicate the meanings of words. Because the translation of a word does not explain its context, more contextual information is needed to effectively instill the meaning of new vocabulary. Saussure's concept of structuralism, as defined by Steve Hoenisch, explains this need more fully:

> The central tenet of structuralism is that the phenomena of human life, whether language or media, are not intelligible except through their network of relationships, making the sign and the system (or structure) in which the sign is embedded primary concepts.
> As such, a sign -- for instance, a word -- gets its meaning only in relation to or in contrast with other signs in a system of signs. (Hoenisch, http://www.criticism.com/md/the_sign.html)

The communicative approach's focus on context is therefore a reaction against the traditional grammar-translation and audio-linguistic methods; it encourages the appropriate *use* of the language in certain situations[1]. Children learn to build grammatical rules in their first language by hearing others provide corrective speech when they make errors and by patterning based on what they hear. This fundamental ability of the brain to write its own language rules based on its auditory and situational experiences is what the communicative method tries to exploit for second-language learning. Depending on their design, textbooks can create opportunities for students to have responsive experiences and to learn from their errors. However, many older (and some current) texts do not capitalize on the students' ability to use context to create their own internal grammar rules and to be able to use the language in everyday situations. For example, rather than provide students with model real-life sentences or conversations

[1] For more explanation of the history of language-learning methodologies, see http://simsim.rug.ac.be/staff/elke/recpast/recpast.html

to build vocabulary and comprehension of grammatical concepts, many older textbooks simply encouraged verb conjugation and translation into English, expecting students to make the jump on their own from the task of memorization or translation to forming natural phrases used to communicate. Here is an example of this type of grammar-translation activity:

Traduisez par écrit ces phrases en français. [Write the translation of these phrases in French]:

1. He would like to get rid of his car.
2. I will never get used to those iced drinks.
3. We are getting used to American cooking.
4. She cannot get along well without her household appliances.
   (Prévost and Dunlop[2] 11)

These sentences are somewhat loosely related to the chapter's story, but their central shared component is the presence of expressions related to the grammar point just introduced before this exercise (vocabulary with the expression "to get"). Rather than eliciting student response and opinions in the target language about the story through interactive (group or pair) activities, this activity focuses on the grammar and translation of set phrases. Like learning to play the piano on a silent practice keyboard, the methodology of training students through rote grammar rules and vocabulary memorization or translation without real-life contextualization muffles a foreign language's real-world significance and usefulness.

The pedagogical theories in programming have had less time to develop than their foreign language counterparts. Niklaus Wirth developed Pascal with the goal of teaching programming, and others have examined the use of programming diagrams and "visual

[2] Their French textbook, *La France en Marche*, is designed for third-year high school students. (Translation mine).

programming" in order to enhance learning and problem-solving skills (Blackwell, Whitley, Good, and Petre). But as Flood and Lockhart show, languages like Pascal "bring a considerable learning overhead to the students and come with a fixed syntax that has to be mastered." (321). Using programming diagrams and metaphors, either with words or pictures, one can visualize the syntax rules of a programming language. We might compare programming language diagrams to the sentence diagramming process (labeling each word in a sentence according to its group and syntax) used in linguistics. The Unified Modeling Language (UML) diagrams have become the most popular diagrams to visualize relationships between programming classes and objects. "In a UML diagram, each class is represented as a rectangle, possibly containing three sections to show the class name, its attributes (data) and its operations (methods) (Lewis and Loftus 162)". In her study of the effects on user performance of a command line language's word order (verb-object or object-verb), Cherry highlights the obvious syntactical relationships and challenges that natural language and programming languages share[3]. Robertson and Lee argue that there are pedagogical concepts in the foreign language realm that could be applied to the teaching of programming languages, but their 1995 article only introduces the need for research in this area in order to determine what theories might best be practically applied to improve teaching and learning of programming languages.

Several pedagogical approaches to teaching programming have emerged in recent years. Interestingly, there has been a shift in programming teaching pedagogy not unlike

[3] For example, if a command language uses the order verb-object, as in the English language ("Print IM"), Cherry hypothesized that that language would be easier for English speakers to learn. Speakers of other languages where the natural language order is object-verb ("IM Print") might find a command language that uses that word order easier to learn.

that in foreign languages. As foreign language teaching has moved from teaching grammar and syntax to an emphasis on the skill of communicating (the reason one learns a language), programming pedagogy has shifted from focusing on learning the syntax of a specific programming language to an emphasis on the skill of problem-solving (the reason one learns to program). In their 2005 article "Teaching Programming," Dasso et al. explore the questions of "why, what, how, and even when to teach programming," and they conclude that there are generally two reasons programming is taught: "(a) because [it] is needed in some professional context; (b) because [it] is considered a good problem solving methodology" (183). They claim that the first reason is debatable when we consider all the professional fields of computing disciplines (programming skills are not needed for some computing fields, such as systems analysis). If learning programming can train students to be better problem-solvers, then programming teaching methods should reflect that goal. Their article also highlights the debate over whether programming should be taught using examples or by teaching theoretical fundamentals before having students use a particular language. In her 1999 article "What are We Doing When We Teach Programming?," Fincher explores several approaches to teaching programming that "[teach] programming as a skill separate from coding": the "Syntax-free" approach, the "Literacy" approach, the "Problem-solving" approach, and "Computation as Interaction" (1-3). The "Syntax-free" approach involves using pseudocode, which is defined by the Wikipedia as "a description of a computer programming algorithm that uses the structural conventions of programming languages, but omits detailed subroutines or language-specific syntax. Flowcharts can be thought of as a graphical form of pseudocode.[4]" In Richard Bornat's book advocating the "Syntax-

[4] http://en.wikipedia.org/wiki/Pseudocode

free" approach, a pseudocode is used so that students don't get bogged down in the syntax of a particular programming language, and "all the exercises in the book can be done with pencil and paper; to use them in an electronic environment requires translation into a programming language" (Fincher 2). The "Literacy" approach focuses on the ability of students to read much more complex programs than they are able to create. Unlike the "Syntax-free" approach, this approach uses a real programming language and encourages interaction with more complex programs written in that language. As Fincher notes, "Just as the majority of children are not motivated by learning rules of grammar, so the majority of students are not motivated by the construction and manipulation of complex internal data structures that involve little or no user interaction" (2). The "Literacy" approach motivates users to interact with the programs they write and experience the computer program actually *doing* something. As the name describes, the "Problem-solving" approach focuses on the "central concept that problem-solving is a transferable skill" that could be used independently of the domain (Fincher, 3). The "Computation as Interaction" approach described by Lynn Andrea Stein emphasizes "a 'real-world' contextualization of skills" and is in response to the popularity of object-orientation in programming teaching. In her book, *Interactive Programming in Java*[5], Stein describes how traditional teaching of programming as a linear series of steps that the programmer controls must be rethought in order for today's programmers to conceptualize how programs are made up of concurrently interacting components. She implies that in order to teach effective problem solving, we must teach programming students to understand the interactions happening in a program (and not just how to write code that will cover all the steps). The ACM's "Computing Curricula 2001" echoes

[5] http://www.cs101.org/ipij/overview.html

many of the same issues related to learning coding first instead of problem solving, and several recent studies have proposed solutions to focusing on teaching problem solving instead of programming coding. Interestingly, some of the solutions for introductory programming courses include linguistic (communicative) ideas such as pseudocode or having students create a "dialect" programming language. For example, Olsen's study using pseudocode at Winthrop University found that "by designing good solutions to problems first, then creating the code, the focus of the course changes from programming to problem solving" (232). Flood and Lockhart developed a tool called Phoenix that allows students to determine the basic grammatical or syntactical concepts of what they call "*dialects* of a low-level language" (322). Students begin with a discussion of issues about a given programming language, such as "should the language be case sensitive?" Flood and Lockhart explain: "Rather than simply presenting the student with a language in which these issues have been permanently decided, we present the class with a debate – they resolve the issue, and the instructor generates a new Phoenix dialect implementing their decision –one whose consequences are thus directly observable" (322). The recent research in programming pedagogy suggests that there is a shift in focus from coding to the transferable skill of problem solving, or communicating with and through the programming language.

Just as the communicative method in teaching foreign languages emphasizes the real-world significance of the language, the new approaches to teaching programming allow students not only to transfer their knowledge of a specific language to real-world use but also to develop problem-solving skills. Given this brief review of the current literature on pedagogy in foreign languages and programming languages, it is evident that

Robertson and Lee's findings still hold true: there is a need for more examination of how the history and studies performed to develop foreign language learning methods might be applied to the teaching of programming languages, and it will be equally interesting to consider how the recent innovations in programming teaching methods might be applied in the foreign language classroom. In this paper, I hope to continue Robertson and Lee's initial research and explore how the pedagogies of these two disciplines might intersect. For example, are teachers in both disciplines aware of current teaching philosophies and trends? Do they see their teaching as following a certain pedagogical methodology? Do their textbook selections reflect that teaching philosophy? What other issues influence how instructors teach and how they use certain textbooks in the classroom? According to today's teachers, what changes need to be made to the textbooks currently available?

III. Procedures

To analyze and compare the pedagogical methodologies of programming and foreign languages, I interviewed instructors of both disciplines regarding their teaching methods and textbook selection. In order to gather information about individual views and practices, I developed interview questions for 3-5 foreign language college instructors and 3-5 programming language college instructors. The purpose of these interviews was to gather information from instructors regarding the kinds of textbooks they select, which sections of those textbooks they actually use when teaching a class, and what they look for when selecting textbooks for their students. Another goal of the questions was to learn how instructors in the two domains identified their personal teaching pedagogies and how well the textbooks they chose reflected those pedagogies.

The interview also included questions about their teaching experiences and personal teaching styles. These are the interview questions that were asked:

1. What levels do you have experience teaching? (Beginning, intermediate, advanced?) Which level do you prefer to teach?
2. How would you describe your teaching style or method?
3. What qualities do you look for in a textbook? Why are these important to you?
4. Which textbooks do you currently use? Did you select these? Which one(s) would you recommend? (Do you have an example of one of these texts where you could show me which sections you use most when teaching a course?)
5. How do you think the textbooks you use reflect your teaching style?
6. What would you change about them?
7. Would you be willing to participate in a follow-up study?

Note that the interview questions were open-ended questions intended to gather insight into how instructors of programming and foreign language classes select texts. These questions were designed to focus on the instructor's textbook selection as it relates to their teaching style and methods as well. The final question was included in the event that this research continues and follow-up is needed.

Since one of my research questions is related to the instructor awareness of teaching methodologies, I interviewed college instructors of foreign and programming languages. I selected college instructors because, although most will not write and publish textbooks, they are the ones teaching within the current curriculum in their respective fields and possibly doing research that will affect the direction of those fields and their future textbooks. In order to make reasonable comparisons, I selected instructors with similar experiences teaching since these instructors were likely to have seen similar problems in their introductory courses. I interviewed instructors with both basic and intermediate (3rd semester minimum) level language teaching experience. The graduate instructors included in the interviews had taught the same levels as the other

faculty interviewed, and there was an equal ratio of faculty to graduate instructors participating in the interviews from the foreign language and programming departments. I selected to contact only faculty from local 4-year public and private colleges, in order to control for anomalies in class size and regional differences.

In order to reduce bias that might be present if all the interviewees were from the same department and knew each other, I recruited instructors from some of the surrounding 4-year colleges and universities (two public research institutions, one private university, and two all-female universities) in Romance Languages and Computer Science departments. After finding instructors' contact information from their department's website, I send them an email requesting participation[6]. Respondents were allowed to respond to the interview questions via email, on the phone, or in person. All participants participated either via the phone, or via email; for the phone interviews, I took handwritten notes.

Following the interviews, I performed a detailed content analysis on the responses gathered during these faculty interviews with the goal of finding common themes related to what instructors look for in a textbook. I also wanted to gauge whether instructors were aware of current pedagogy trends in their fields and discover how they viewed their personal teaching style and the textbook engaging student interest in the classroom. During the analysis of the interview data, I took into consideration any biases (such as situations where the professor may be the author of a text used!) and tried to balance these with other faculty responses. Another situation of bias I had to consider was the fact that most of the foreign language faculty that I interviewed had known each other or worked together previously at the same public institution, even though they were

[6] This study was approved by the AAIRB of the University of North Carolina at Chapel Hill.

currently at different institutions. Since I did not disclose to the interviewees who the other participants were, this helped to reduce that information potentially influencing the respondents' answers. In order to reduce linguistic differences, I selected to interview instructors of foreign languages and programming languages that share similar syntax. All of the foreign language instructors interviewed taught either French or Spanish, and most of the programming language instructors interviewed taught using Java.

Finally, after the interviews were completed, I examined the textbooks mentioned by faculty during their interviews. I performed a detailed content analysis in order to observe pedagogical themes and methods that might crossover from one domain to the other. This content analysis was done by systematically examining the text's introduction and stated goals, table of contents, presentation of key concepts, and exercises. I recorded my findings by taking notes in on an Excel grid with color-coding to indicate similarities in purpose, style, or goals.

Once all the themes from the interview data and textbooks were collected and analyzed, I examined how the resulting differences might be applied to the teaching philosophies of the other language domain and how these might manifest themselves in their respective textbooks. I explored the possibilities that exchanging teaching ideas between foreign and programming language texts might afford if these were applied creatively across these two curricula.

IV. Results

I completed seven interviews, four of which were with foreign language instructors and three of which were with programming instructors. Three of the four

foreign language instructors interviewed taught French (one was a native French speaker), and the fourth instructor taught Spanish. The foreign language instructors had taught introductory, intermediate, and advanced courses; two of them had experience mentoring student teachers or teaching assistants in their field. The programming language instructors had also taught all levels: beginning, intermediate, and advanced. One instructor had experience teaching programming in a corporate setting (as a refresher course for professionals) as well as in the university setting. Most instructors chose to respond via email, however I did have the opportunity to do the interview on the phone with two instructors (one from each group). These phone interviews lasted an average of approximately 30 minutes. Participants' responses varied in length and amount of detail provided.

The interviews brought forth several themes related to the challenges of finding textbooks that promote an ideal teaching methodology. Following is a summary of the most important themes or attributes that the instructors (both foreign language and programming language instructors) said were important to them when selecting a textbook:

IV a. *The textbook should proceed in an orderly fashion*

Faculty in both groups (two programming instructors and three foreign language instructors) mentioned the importance of "a good progression" and a logical approach: activities should move from simple to advanced. While opinions vary on what constitutes a "good progression" (particularly in the programming pedagogy domain where there is some debate on when to teach programming at all in a curriculum), all the

programming and language instructors interviewed agreed that texts should move logically and gradually from simpler exercises to more advanced ones. For example, one programming instructor mentioned that introductory programming textbooks should not include code that is clearly beyond a student's level and dismiss it by saying "you may ignore the following lines"; he implied that the reason for this sort of illogical ordering (including advanced material too early) was to make things easier on the textbook's author instead of focusing on students' needs. A foreign language instructor (L1) explained the importance of presenting activities in the proper sequence so that students have opportunities to move from "controlled or mechanical activities to more open-ended, communicative [or creative] ones." One example of this in a foreign language text would be having students complete vocabulary exercises at home that had answer keys (so they could self-check) then doing a more meaningful, yet still "easy answers," activity in class, and finally finishing in class with a more communicative and creative activity, such as writing a script for a scene using the vocabulary studied. One instructor cited the importance of "incremented examples," that is, examples that incorporated a balance of providing theory and practice exercises. Another foreign language instructor (L2) described this idea by saying that texts should include "clear guidance and explanations." Both groups of instructors mentioned the importance of logical progression, from simple to more advanced concepts and examples with equilibrium of theory and exercises.

IV b. *The textbook should include plenty of real-world examples.*

Related to their interest in the progression of exercises and theory, instructors also emphasized the importance of the quantity and types of exercises that should be included.

Instructors spoke often about the importance of real-world exercises and examples. Faculty used words such as “real-life questions,” “real-world examples,” “lively cultural examples,” and “exercises with real-life situations” to describe this desirable attribute. The foreign language instructors voiced their concern that the exercises be based on a communicative approach and therefore useful and believable for real-world situations. One programming language instructor provided his own examples in his online lecture notes for students, since he felt these better reflected his teaching methodology than the textbooks available (he claimed the textbook was used as a secondary reinforcement). Figure 1 is an example of some of his online materials.

Figure 1: Lecture notes for explaining Java concepts of Class, Objects, and Methods.

Lecture Notes

1. What is a class?
    1. A class is general template for something you are trying to represent
    2. Examples:
        - Automobile
        - Soda Machine
        - Person
2. What is an object?
    1. An object is a single copy of a class
    2. If the class is Person an object might be Tom or Sally
    3. Every object is of a class
    4. The class an object belongs to defines its attributes
3. Parts defined by a class
    1. A class defines the actions that objects of its type may take with the methods
    2. Methods may be though of as the verbs of an object
    3. Examples for a Car:
        - drive
        - reverse
        - honk
    4. A class defines the knowledge and characteristics of an object with the instance variables
    5. Instance variables may be thought of as the nouns of an object

6. Examples for a Car:
    - color
    - speed
    - range

4. How methods and instance variables apply to an object
    1. An object is one instance of a class
    2. Each object will have its own instance variables as defined in the class
    3. Example:
        - Each Car may be a different color
        - Each Car may be going a different speed
        - Each Car may have a different range
        - They are all still Cars, but each are a DIFFERENT Car
    4. Each object of a particular class will offer the SAME instance variable names, but each will have DIFFERENT data stored in the variable
    5. All objects of a class have the same methods, but each method acts on the particular object it is called upon.

After his rather "linguistic" real-world explanations and analogies of the concepts covered, this instructor's lecture notes continue with a related and heavily-commented example in Java for students to study and try. As both groups mentioned, textbooks should have many sample exercises that provide students the chance to see the topic applied in the real world.

IV c. *Texts should be " interactive."*

If the purpose of a (programming or foreign) language is to communicate something to someone, then textbooks in these domains should facilitate that communication exchange through hands-on practice. Professors from both domains described this "interactive" quality of their ideal textbook as something that would engage students with the topic and allow them to communicate or practice using what they were learning. For example, one programming instructor (P1) suggested that, in an interactive textbook, "a topic is presented, the students have an exercise to practice and expand on the lecture material,

and then the students are able to ask questions during the next session about the topic." For the foreign language instructors, an interactive approach meant a communicative approach; that is, the textbook should facilitate students interacting and communicating with one another. They cited the importance of practical self-teaching exercises, immersion in the language and contextualization[7]. Several of the programming language instructors noted that texts for introductory programming courses tend toward two ends of the spectrum: either too heavy on theory and not enough examples for the students to interact with or too many examples and not enough theory. As one programming professor (P2) stated, we "need some good wikis on programming and [to] ditch the books […], [since] they cost too much and there's no interaction." Despite his negative attitude toward textbooks in general, P2 praised the "conversational tone" and simple drawings that could be done on a blackboard from one of his favorite texts, *Java Structures*, by Duane A Bailey (2nd edition), since these aspects of the text facilitate communication and interactivity with the book's content. His comments echoed the desire expressed by other foreign language and programming professors that students need to interact in (or practically apply) a language if they are to learn it. P2 summarized this point: "Learning to program from a book is like learning to paint from a book; you might get some tips, but mostly what you need to do is close the book and do it." Instructors from both groups emphasized what they call interactivity as indispensable; the

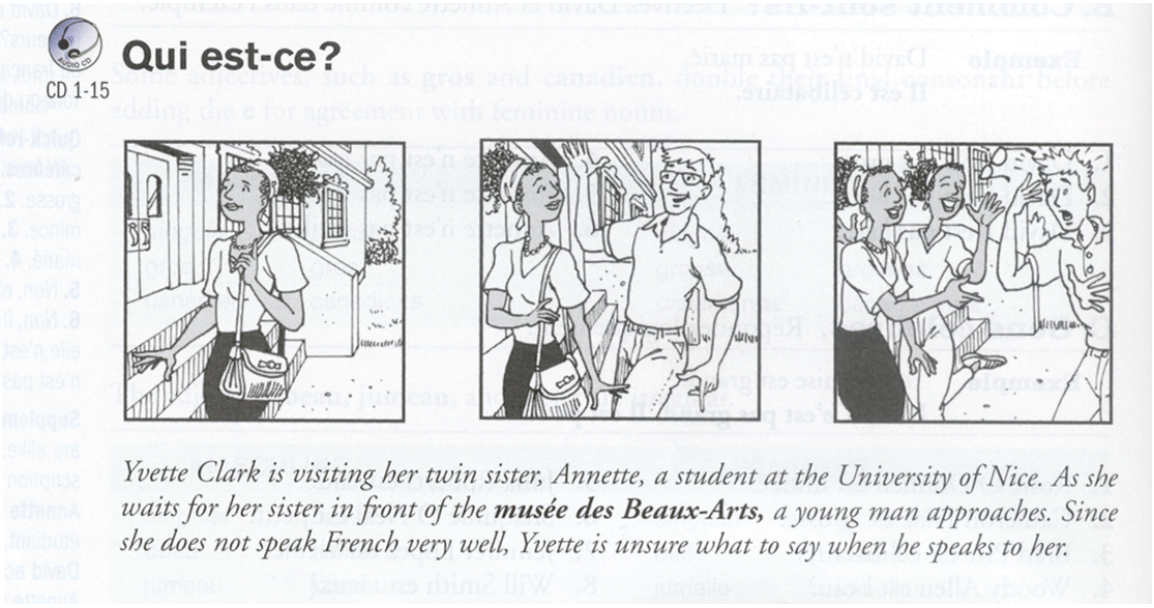
[7] Alice Omaggio Hadley's book *Teaching Language in Context* expands the idea of contextualization. This method of foreign language teaching emphasizes using the target language within contexts, or real life situations in which a person uses language, so that students must imagine themselves using the language to interact and communicate in a real circumstance or setting. For example, rather than merely asking students to memorize and write verb conjugations on an exam, instructors can create a context for a conversation in which the verbs have been left blank and students must select the appropriate verb and verb form to complete the conversation so that it makes sense.

consensus was that texts should push students toward interaction in the (programming or foreign) language.

IV d. *Textbooks should be well illustrated.*

Faculty pointed to the importance of illustrations in communicating ideas in the textbooks they selected. They prefer texts to be visually appealing with useful graphics and visualizations of concepts studied. For example, the French introductory textbook *Horizons* uses comic strips without word balloons, such as the one in figure two below, to help students picture a situation and explore the conversation that the picture suggests:

Figure 2: From Manley, Smith et al. 2006 36



Another foreign language instructor noted how incorporating appropriate cultural artwork at various points in the text was useful to contextualizing the lesson. One programming instructor discussed his desire for drawings that were simple enough to be replicated on a chalkboard for use in class. While the example he gave of this type of illustration is from a textbook used in a third-semester programming course (Duane A Bailey's *Java*

Structures (2nd edition)), it is representative of the kind of illustrations that instructors seek in texts:

Figure 3: From Bailey 1999 84

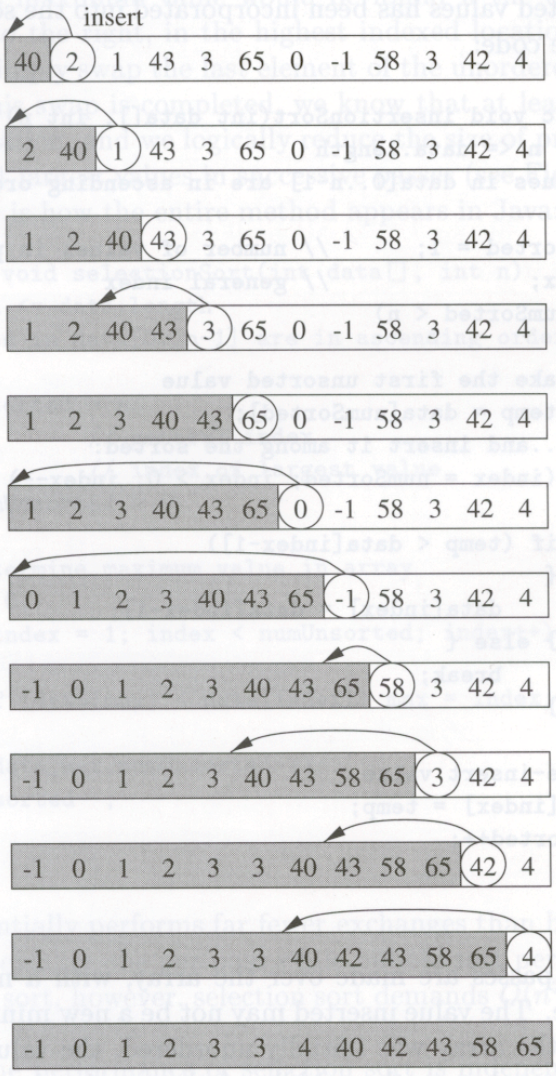


Figure 5.4 Profile of the passes of insertion sort: shaded values form a "hand" of sorted values. Circled values are successively inserted into the hand.

Faculty in both fields noted the usefulness of illustrations, visuals, and graphics in the texts they selected.

IV e. *Textbooks should reflect my teaching style*.

Instructors also saw their personal teaching style as something that came through in their textbook selection. Several programming instructors described their pedagogical style as "hands-on," "visual," and "graphics/example-driven." P2 said he looked for books that could come up with succinct questions that might launch students into solving a problem (as opposed to providing a list of instructions for them to follow). He gave the example of one of his lab assignments, which consisted of just one question: "How fast is your computer?" and very minimal instructions. He indicated that this style was better for teaching students to understand the conceptual core of programming and problem solving techniques. One of the foreign language instructors described her personal teaching methodology with very similar language, explaining that she stresses the "meaning of utterances and not just the grammatical accuracy of a statement." She echoes the comments of P2: "I limit the amount of explanation in class…in favor of lots of practice…class time is spent in using the material studied and verifying hypotheses developed in preparation." Practical application of the languages was part of the personal teaching methodology of all the instructors in both groups.

IV f. *Foreign-language specific themes*

In general, the foreign language instructors had more specific ideas about what they looked for in a textbook than their programming language counterparts. For example, foreign language instructors mentioned immersion as an important attribute. That is, the text should provide the experience of being in a land where the target language is used

and where one would be immersed in its expression, vocabulary, and culture. Related to this, they also mentioned the importance of a text having up-to-date cultural information.

Foreign language instructors also noted that they desired textbooks that incorporated the National Standards for Foreign Language Learning. Updated in 1999, the document "Standards for Foreign Language Learning: Preparing for the 21st century" showcases the importance of teaching students through context and communication. It defines five goal areas: Communication, Cultures, Connections, Comparisons, and Communities—the five C's of Foreign Language Education (that is, "Knowing how, when, and why to say what to whom").

Finally, one foreign language instructor mentioned that she looked for textbooks that had a lot of marginal annotations in the instructors' annotated edition in order to facilitate use by inexperienced or relatively inexperienced instructors.

IV g. *Programming-language specific themes*

Programming language instructors did not highlight the importance of a text being up-to-date, stating that most of the necessary examples and information were available and up-to-date online anyway. The programming language instructors interviewed did not mention any national standards of pedagogy in relation to their textbook selection. In fact, one programming professor was completely anti-textbook altogether.

They did emphasize their use of audio/visual ancillaries (more than their foreign language counterparts, who, surprisingly, did not emphasize this aspect of a textbook). The programming instructors interviewed used the slides that come with the textbook and

also developed their own online lecture notes or online example exercises to supplement the textbooks.

IV. h. Textbook comparison and analysis.

I compared the contents of three programming textbooks and three foreign language (French) textbooks that were mentioned by the interviewed instructors. Figure 4 provides the bibliographical information on these texts:

Figure 4:

Manley, Joan H., Stuart Smith, John T. McMinn, Marc A. Prévost. Horizons. Third edition. Boston: Heinle, 2006.

This is an introductory-level French language text.

Muyskens, Judith A., Alice Omaggio Hadley, Claudine Convert-Chalmers, and Michel Sarner, contributor. Rendez-vous: an Invitation to French. Sixth edition. New York: McGraw-Hill Publishing Company, 2002.

This is an introductory-level French language text.

Jarausch, Hannelore and Claire Tufts. Sur le Vif. Fourth edition. Boston: Heinle and Heinle, 2006.

This is an intermediate-level French language text.

Lewis, John and William Loftus. Java Software Solutions. New York: Pearson Education, Inc., 2005.

This is an introductory-level programming text.

Deitel, H.M. and P.J. Deitel. Java: How to Program. Fourth edition. New Jersey: Prentice Hall, 2002.

This is an introductory-level programming text.

Bailey, Duane. Java Structures. New York: WCB McGraw-Hill, 1999.

This is an intermediate-level programming text.

To see what qualities these textbooks shared, I analyzed the texts' introductions and stated goals, tables of contents, presentations of key concepts, and exercises. Here are some of the themes that resulted from the analysis of these texts:

1. *The existence of pedagogical standards and a common goal*

The introductory-level texts in both domains recognized that there are accepted "standards" for what is good pedagogy in their domains. For example, the foreign language texts emphasized national standards, the communicative approach, context-based activities, and cultural material. The introductions to these texts used a shared vocabulary to talk about their pedagogy in relation to these areas. Similarly, the programming language texts seemed to agree on the ultimate goal for instruction (to teach students to write good software with clarity and to understand the principles of software engineering), although their methods for arriving at that goal were slightly different. Some books focused on having plenty of examples (Deitel and Deitel) while others try to blend together theory with practical examples (Lewis and Loftus). For example, Deitel and Deitel's introduction talks about the "natural progression toward object-oriented programming" yet jumps right into examples of rather complex Java structures as early as chapter two, while the Lewis and Loftus introductory-level text and Bailey's intermediate-level text seem to build up more gradually from simple to more complex levels of object-oriented programming (XLI). In the introduction to the Deitel and Deitel text, its authors describe themselves as "educators who teach edge-of-the-practice topics in industry classrooms worldwide". They even admit that the way the book jumps right to using the Swing-style GUI components as early as chapter two has

been seen as a “gutsy” move, but that they believe students of Java want to “cut to the chase” and get started programming early on in the book (XLI). By contrast, Lewis and Loftus boast in their more gradual build toward helping students write good software, claiming that their text “uses a natural progression that culminates in the ability to design real object-oriented solutions…this text integrates practices that serve as the foundation of good programming skills…students learn how to solve problems as well as how to implement solutions” (VIII).

2. *Texts in both domains addressed the uneven skillset of students in intermediate level courses*

The two intermediate texts analyzed acknowledged that students often come to the intermediate-level courses with different levels of skills, and their authors suggested that students who feel behind might seek out a simpler review text. In *Java Solutions,* the author chose to include a brief overview of Java in the appendix to help students with a less solid background in Java to catch up with their classmates. All programming texts also incorporated online examples to help students understand the concepts presented and get more practice if needed

3. *Texts in both domains used graphics to accent key concepts in the text*

Graphics help alert students to key concepts as they read. For example, in the Deitel and Deitel text, there are various small icons with ants portraying the different concepts, such as an ant who is smiling with the “thumbs up” sign beside a “good programming practice” tip and one who gives the “thumbs down” sign beside the explanation of a “common programming error” (270). The French introductory text *Rendez-vous* claims

in its introduction that "sketches, photographs, realia[8], and varied reinforcement activities enable students to use the new vocabulary without recourse to translations" (xix). All three foreign language textbooks used consistent, colorful graphics in the margins next to important grammar points. In addition to the graphics to accent key concepts, both sets of texts used self-check questions at the end of sections presenting new concepts to help students verify their comprehension of newly presented material.

4. *Differences between the texts in the two domains*

One contrasting attribute between the foreign language and programming textbooks was the types of ancillaries available to instructors. The foreign language textbooks included annotated instructor editions with lesson plan ideas in the margins. In contrast, the programming language texts either did not offer an instructors' annotated edition, or they instead offered a solutions manual/CD and/or PowerPoint slides of the examples and exercises given in the text. The Deitel and Deitel text did offer a website with a "Syllabus Manager" for course planning, as well as additional exercises and reference materials for both students and instructors. The Lewis and Loftus text also offered a free subscription to Addison-Wesley's CodeMate, "an online environment that brings the text to life…[and] allows you to work with many of the texts's Code Listings and Programming Projects" (XI).

Other differences between the two domains were the order of presentation (principles and practical applications) and the use of icons to indicate a progression of ideas. For example, the intermediate programming language textbook used gear-like icons that represented the build from principles to structures to example code. In contrast,

[8] "Realia" can be defined as cultural artifacts that represent a language or culture, such as metro tickets or maps, authentic music, symbols, or costumes.

the foreign language texts varied in their order of presentation of concepts. For example, the introductory texts presented explanations and their associated examples and exercises interwoven throughout the chapters, while the intermediate-level text presented the exercises and practical application section in the first half of the text with the presentation of grammar concepts and principles separated in the second half of the text.

V. Discussion

Based on the analysis of the interview data collected and the textbook content from these two domains, the pedagogies of these two disciplines might intersect in several interesting ways. Based on the interviews and textbook introductions, foreign language teachers use a shared vocabulary (such as “contextualization” and “communicative approach”) to discuss their ideas about their teaching methods and desires for a text, while programming language instructors described several different approaches with varied vocabulary when discussing their methods and goals. Perhaps because of the longer history of pedagogical theories in foreign languages, instructors of foreign languages are more aware of current teaching methodologies in their field. The fact that programming language instructors and texts did not expressly label their methods with similar vocabulary may mean that there is less of a consensus among this group of instructors regarding their pedagogical methods. It is also possible that textbook publishers are trying to meet the needs of a diverse audience of both university and professional students and therefore are less concerned with following any particular teaching methodology in their texts.

Instructors in both domains were aware of their own teaching philosophy, although the programming language instructors did not as readily identify themselves as

following a particular pedagogical methodology. Their responses about the textbooks available indicated that they worked with what was available, adapting where necessary to their particular teaching style (such as the instructor who had written extensive online lecture notes and examples to complement his chosen textbook). According to the interview data, the foreign language instructors were generally satisfied with the textbooks available (they suggested little changes to the texts they currently use), but the programming language instructors had many suggestions to improve the current textbooks available for introductory and intermediate texts, revealing room for improvement in the programming texts currently available. It seems that programming texts need to find a balance between teaching programming theory and providing applicable examples and hands-on experience for students. Dasso et al. (2005) explore several of the reasons why it is difficult for textbooks to answer the needs of instructors and students:

> There was and still is a raging dispute in teaching programming about several issues that have to do with the method, programming language to use, and these have as well [other] considerations…we are confronted by those [instructors] that insist that programming must be taught using examples and particularly using interactive media on a computer on a hands-on approach and those that insist on teaching the theoretical fundamentals of programming, leaving the practice using a particular language for when the students have mastered the theoretical fundamentals of programming. (Dasso et al. 183 and 185)

The debates over "what to teach when and how" in introductory programming courses are still raging, and perhaps as the answers to these questions become more solidified, the programming textbooks available will begin to fulfill more accurately the desires of those using them to teach. In this study, many similar themes came forth in both groups of instructors, suggesting that what instructors are looking for in a classroom textbook in these programming and foreign language is actually quite similar. Both groups desired

an orderly progression, real-world examples, interactivity and good illustrations. As foreign language textbooks have changed dramatically over the past 100 years, we should expect that programming language textbooks will undergo similar transformations, assuming the desires of the faculty interviewed are representative of the general populace of teachers in these two domains and assuming the needs remain the same. The Computing Curricula of the ACM does not impose answers to the debates over programming teaching methodology, but its existence does encourage discussion on the topic. Perhaps some national standards in programming teaching will emerge and transform the textbooks and methodologies currently in use.

We might also consider how the themes brought forth by foreign language instructors and not mentioned or shared by the programming language instructors might reveal new ideas about how to change the programming textbooks available. The fact that instructors of foreign languages mentioned the National Standards communicated that not only do these exist but that their existence and importance have been transmitted and accepted by the pedagogical community. The other theme mentioned by foreign language instructors but not by programming instructors was immersion. Could this concept be useful to programming textbooks as well? One professor suggested putting the “textbooks” on the web as wikis; this would certainly immerse students in a real-world situation, putting them in contact with other students learning the same language.

With the advantage of their unique development and varied inceptions, programming texts could also contribute new ideas about how to present grammatical concepts in foreign language textbooks. For example, the use of icons to identify a progression of ideas in the programming language texts (such as the compass and the

gear icons used in *Java Solutions*) might be applied to foreign language texts as activities move from being merely "manipulative" to gradually having a more open-ended structure, allowing students to practice and be creative. Foreign language texts could also benefit from the example of their programming counterparts' online ancillaries. While most up-to-date foreign language texts do have associated websites with activities, it is unclear how much these are used in and outside the classroom (none of the instructors interviewed mentioned using these).

VI. Conclusion

Based on the texts analyzed, foreign language texts are generally built intentionally around a teaching philosophy. The introductions of the programming language texts analyzed suggested that engineers who may base their textbook design on something other than a specific teaching philosophy often write these. For example, some texts attempt to meet the needs of varied audiences (professionals and students), and therefore their methods may need to be adapted to meet the needs of different instructors. Given this difference in the creation process for programming textbooks, we can see that there may be some benefit to developing a pedagogical framework within which to write effective programming textbooks that allow students to grasp concepts quickly and comprehensively. One technique that may be developed from foreign language teaching is applying the "communicative method" of language learning to programming. By examining sample programs and their results (as is advocated by Owen Astrachan's "Literacy" approach, used at Duke University), students can grasp general syntax rules and explain them in their own words. Similarly, students in a foreign language class taught only in the target language learn grammatical concepts by

reading and hearing sample phrases and expressions. Holmboe, in his article "The Linguistics of Object-Oriented Design: Implications for Teaching," describes the difference between how children develop concepts and how scientific learning takes place. He explains that children develop everyday language and ideas "from the concrete and specific to the general and abstract" while scientific learning methods often proceed the other way around, with scientific concepts and definitions being delivered from a teacher or text in a highly generalized and abstract form, then later being repeatedly applied to concrete phenomena (189). Perhaps creating programming textbooks with a blend of theory, practical examples, and more samples of more complex programs for students to read (as are used by the Literacy Approach) might be an acceptable compromise between the two camps of "programming with examples first, then theory" (concrete to abstract) versus "theory, then practice" (abstract to concrete). Finding this balance can be challenging for foreign language texts as well, as students in that discipline need clear conceptual and grammar explanations interwoven with practical application and interaction with others in the target language. As Sally Fincher concludes her examination of several different programming language teaching methodologies, "changing an approach to teaching requires first the knowledge that other approaches are possible; secondly it requires reflective practitioners…it also requires evaluation and evidences of the success of any given approach" (5). This study comparing foreign and programming language teaching has found some areas where other approaches might be applied, yet several questions still remain: how will programming teaching methods develop and solidify in the years to come? How can we encourage a move toward evaluating the success of various approaches? There is a need for more study of how

students perceive and influence the teaching of programming languages as well: how might students' needs influence the order that concepts are taught and which approaches are tested? Through this study, I hope to have contributed to awareness of teaching methodologies used in foreign and programming language textbooks. Perhaps as more research is done in this area, these two domains of language learning might impact each other's development of textbooks, pedagogies, and national standards.

Selected Bibliography

Astrachan, Owen and Reed, David. "AAA and CS1: The Applied Apprenticeship Approach to CS1." Proceedings ACM SIGCSE Symposium, 1995.

Barreau, Deborah K. " 'Making Do': Adapting transaction Systems to Organizational Needs." Library and Information Science Research. vol 23: 1, Spring 2001, 27-43.

Blackwell, A.F. (1998). Metaphor in Diagrams. Unpublished PhD Thesis, University of Cambridge. Ch2. http://www.cl.cam.ac.uk/~afb21/publications/thesis/chapter2.html (accessed April 6, 2006)

Blackwell, Alan F., Kirsten N. Whitley, Judith Good, and Marian Petre. "Cognitive Factors in Programming with Diagrams." Artificial Intelligence Review. Kluwer Academic Publishers: Netherlands. 15: 95-114, 2001.

Cherry, Joan Margaret. "Command languages: effects of word order on user performance." Microfiche of dissertation typescript. Ann Arbor, Mich.: University Microfilms International: University of Pittsburgh, 1983.

Cox, Richard, Jean McKendree, Richard Tobin, John Lee, and Terry Mayes. "Vicarious Learning From Dialogue and Discourse." Instructional Science. Kluwer Academic Publishers: Netherlands. 27: 431-458, 1999.

Dasso, Aristides, Ana Funes, Daniel Riesco, Germán Montejano, Mario Peralta, and Carlos Salgado. "Teaching Programming." JEITICS 2005: Primeras Jornadas de Educatión en Informática y TICS en Argentina. 2005, 183-186.

Elke De Man "An overview of methodologies for language teaching" http://simsim.rug.ac.be/staff/elke/recpast/recpast.html (accessed December 1, 2003).

Fincher, Sally. "What are We Doing When We Teach Programming?" 29th ASEE/IEEE Frontiers in Education Conference, November 10-13, 1999, San Juan, Puerto Rico.

Flick, William C. and Philip H. Isensee. "The New User: Teaching 'Computerese' as a Second Language." ACM. 212-219. 1982.

Flood, Raymond and Bob Lockhart. "Teaching Programming Collaboratively." ITiCSE'05: Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, June 27-29, 2005, Monte de Caparica, Portugal. 321-324.

Haynes, Christopher T. "Experience with an Analytic Approach to Teaching Programming Languages." SIGSCE Bulletin. vol 30:3, 9-20, December 1998.

Hoenisch, Steve. "The Sign, the Signifier, and the Signified." http://www.criticism.com/md/the_sign.html (accessed March 27, 2006).

Holmboe, Christian. "The Linguistics of Object-Oriented Design: Implications for Teaching." ITiCSE'05: Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, June 27-29, 2005, Monte de Caparica, Portugal. 188-192.

Hood, Cynthia s. and Dennis J Hood. "Teaching Programming and Language Concepts using LEGOs." ITiCSE'05: Annual Joint Conference Integrating Technology into Computer Science Education, Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, June 27-29, 2005, Monte de Caparica, Portugal. 19-23.

The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, "Computing Curricula 2001, Computer Science". Final Report, (December 15, 2001).

The Joint Task Force on Computing Curricula 2004. "Computing Curricula 2004, Overview Report including A Guide to Undergraduate Degree Programs in Computing". The Association for Computing Machinery (ACM), The Association for Information Systems (AIS), The Computer Society (IEEE-CS), November 22, 2004.

The Joint Task Force on National Standards in Foreign Language Education, "Standards for Foreign language Learning: Preparing for the 21st Century". http://www.actfl.org/i4a/pages/index.cfm?pageid=3392 ACTFL, AATF, AATG, AATSP, ACL/APA, ACTR, CLASS/CLTA, and NCSTJ/ATJ. 1996; updated 1999.

Muller, Orna, Bruria Haberman and Haim Averbuch. "(An Almost) Pedagogical Pattern for Pattern-based Problem-solving Instruction." Annual Joint Conference Integrating Technology into Computer Science Education Proceedings of the 9th Annual SIGCSE conference on Innovation and Technology in Computer Science Education. 102-106.

Olsen, Anne. "Using Pseudocode to Teach Problem Solving." JCSC. vol. 21:2 (December, 2005).

Omaggio-Hadley, Alice. Teaching Language in Context. 3rd edition. Boston: Heinle & Heinle, 2001.

Prévost, Geneviève and Karen Dunlop. La France en Marche. Columbus, OH: Charles E Merrill Publishing Co., 1971.

Robertson, Stephanie A. and Martin P. Lee. "The Acquisition of Second Natural Language Pedagogy to the Teaching of Programming Languages." SIGCSE Bulletin. vol. 27:4, 9-20, December 1995.

Schwarz, Jason. Course notes. http://courses.ncsu.edu/csc116/lec/002/lecture_03/ (Accessed April 10, 2006).

Stein, Lynn Andrea. Interactive Programming in Java. 2003. http://www.cs101.org/ipij/ (accessed April 6, 2006).

Wirth, Niklaus. Algorithms + Data Structures=Programs. Englewood Cliffs, N.J.: Prentice-Hall, 1976.