New Course Proposal Questionnaire

Bridging the Introductory Course Sequence (221/222/321) and 560

Draft: Thursday, January 11, 2007
Author: Paul Sivilotti

What is the nature of the proposed course?

1. *What is the overall nature of the course briefly stated?*

The course is intended as a first course in Java. In addition to language syntax and mechanics, the course will cover a variety of tools used in various stages of software development (eg testing, documentation, version control). Finally, the course will present several best practices, explaining them as manifestations of component-based software principles seen in the introductory course sequence.

2. *What is the scope of the course in terms of expected changes to the current CSE curriculum?*

This course will have significant impact on the core CSE curriculum. We are proposing that this new course be a required core class. The intellectual content covered in this new class would have some overlap with the existing 459.23 (Java language mechanics), 560 (tools for software development, testing, documentation), and 321 (application of principles of component-based software design and development).

3. *What existing CSE courses or courses are related to the proposed course by similarity, prerequisites, etc.?*

The proposed course would be a prerequisite to 560. As a prerequisite, it would require 222. Thus, it would occupy a slot concurrent to 321, although we do not anticipate making these courses co-requisites.

4. *What other departments will be concerned with the proposed course and how?*

Any stakeholders in 560 will be directly affected since the prerequisite chain for 560 is larger (in breadth, not length).

What problems does the proposed course solve and/or create?

1. *Why is there a need for this course?*

(a) Tools are helpful in 560. In 560, we want group projects to begin almost immediately: The bigger the lab (and the longer the students have to work on it) the better. But students have not yet learned many tools that are useful for “programming in the large” (CVS, Make, testing harnesses, debugger) until (midway through) 560. As a result, this information comes too late to be helpful in the early lab(s).

(b) Students lack appreciation for real-world applicability of RESOLVE foundations. In 560 (and beyond?), students working in languages other than RESOLVE seem to have difficulty mapping RESOLVE concepts into their group's implementation language. For example, they don't see the relevance of defining a mathematical model, or of distinguishing between concrete and abstract state.

(c) Java is a complex language. It is hard to do Java justice in a 1-credit S/U class. In 459.23 (Programming in Java), it is practically the end of the quarter before students can do anything interesting in Java. In order to get to the Collections Framework, one must first cover interfaces, inheritance, generics, and exceptions as well as the OO basics (objects, classes, encapsulation, syntax).

2. What could not be accomplished if this course were not created?
The current curriculum does not allow for the integration of the presentation of a programming language and the application of component-based software design principles (221/222/321) in the context of that language. The current curriculum structure also requires that students begin their project work in 560 with little knowledge of the tools that can help.

3. Who is demanding the course or the product of the course?
The 560 instructors and some students, although student interest should really be gauged more carefully and methodically.

4. Who is the intended/expected audience for the course?
Undergraduate CSE majors and minors.

5. How many students would be involved in the course?
Annual enrollment would be similar to the steady-state annual enrollment of 560 and 321. I would guess that number to be roughly around 180, but this guess should be confirmed with a more informed opinion.

6. How is the course related to national movements or trends?
It is similar to many introductory programming courses in CS curricula across the country in that it is a first-course presentation of a real programming language. It is also similar to the (inter)national trend in its choice of Java. In my opinion, Java is a terrible choice (from a pedagogical perspective) for CS 1/2. The complexities and idiosyncrasies of the language present the novice programmer with a significant barrier to understanding many core CS concepts. Nevertheless, it has been a popular choice for the last 10 years.

The proposed course differs significantly from the (international) trend in that it is a first-course in Java without being a first-course in programming. Students arrive with a good background in principles of component-based software design. This course can then focus on the manifestation of these principles in the context of Java, including all of the languages complexities and idiosyncracies.

7. How is the course related to the GRE advanced test in CS?

The GRE advanced test in CS lists its subject focuses on abstract concepts rather than a particular language's syntax or semantics. Part 1 is "Software Systems and Methodology" and accounts for 40% of the total score. Within this part, there are 3 subsections that are potentially relevant: Data Structures, Program Control and Structure, and Programming Languages. None of these mention Java (or any other particular programming language) explicitly.

The GRE practice test currently available from ETS includes several questions that have pseudocode written in a C-like notation. These code snippets do not include pointers. They tend to focus on procedural control structures (functions, while loops, etc) and simple variables or arrays.

Although Java mechanics, as such, are not part of the GRE test, data structures, control structures (eg exceptions) and languages (eg scoping, parameter passing) are. These concepts will be part of the new course.

What is the proposed course's detailed structure?

1. What draft sample course descriptions (objectives, prerequisites, syllabi, texts, grading schemes, ...) are available?
A draft official syllabus has been prepared and submitted via the CSE web portal.

2. What draft sample homework problems, lab assignments, and exam questions are available?
Four labs that were developed as part of a revamping of 459.23 could be used in this new course. These labs can be found at:

http://www.cse.ohio-state.edu/~paolo/teaching/45923/

The lab topics are:
a) Implementing a class representing an arbitrarily large natural number
b) Implementing a personnel database
c) Implementing the Random Writer from "nifty assignments"

3. What is the history and previous experience with this course (pilot sections, other universities, ...)?
Many universities have a "programming in the small" course that is similar to the one being proposed here. The focus is on language facility and tools, as well as a small amount of OO design principles.

What resources are needed to implement and conduct the course?

1. What faculty are available/will be required to teach the course?
The software engineering faculty would be the best choices to teach the course, especially if the manifestation of component-based software design principles is to be a serious component of this course. Faculty from other areas but with practical expertise in Java development could also teach the course. Clinical faculty could draw on their practical experience to teach the course.

2. What computer resources (hardware, software, staff time, etc.) will be required to implement this course? What's the initial, one-time cost and what is the yearly maintenance, upgrade, replacement cost?
The main student labs will need Eclipse (already installed for 201) and JUnit. CVS will also be required (although Eclipse comes with a CVS client, it does not come with a CVS server).

3. What other materials or resources will be required to implement this course?
None.

4. What kind of grading or lab assistant support will this course need?
JUnit test cases would be used for all of the acceptance testing of student submissions, so that part of the grading should be light. The grader will also have to examine Javadoc documentation, however, and provide feedback on design. This will be a lab-intensive course and so will likely be a 10 hr/week grading load (for a 30 student section).

How will the course be implemented?

1. If applicable, how will the course's curriculum be phased in?
A new core package will have to be created. Students can either take: 321 + 560 + 459, or 321(new) + new course + 560(new). After modifying 321, the old version of 560 could be offered for a full calendar, while introducing the occasional offering of the revised 560.

Open question: is it possible to pilot the proposed course without committing to this change in core curriculum?

2. What are the "fall-back" positions if the changes cannot be completed as originally planned?
Status quo.

How will "success" of the course be gauged?

1. What are the criteria to be evaluated?
(a) Student performance in 560. Do students arrive ready to undertake a large programming project armed with the appropriate tools for testing, documentation, and code versioning.
(b) Student perception of RESOLVE. Can students see how industry best practices are approximations of sound principles driven by component-based approaches to software design?
(c) Student proficiency in Java.

2. What provisions are there to conduct the evaluation?
(a) Feedback from 560 instructors, likely as part of the “software spine” course group.
(b) Exit and alumni surveys.
(c) None.