Software Specification and Verification I

`CS 410SV1, 555`, Winter 96
(Revision 1 December 1995)

Instructor:	Sergio Antoy
Office:	PCAT 115, ph. 725-3009, `antoy@cs.pdx.edu`
Office Hours:	Tue & Thu 9:20-10:20
Time:	Tue & Thu 8-9:15
Location:	PCAT 28

This is a graduate course on theoretical and formal aspects of the software lifecycle. Its goals are to provide an overview of the foundations of the field and an appreciation of both the elegance of its underlying ideas and the difficulty of their practical application. Some landmark techniques and methods for the specification of software and the verification of its correctness are analyzed in depth.

A considerable independence and maturity is expected from the students. There is no textbook. The reading material consists of a dozen articles available from Copyman, 1915 SW 6th Ave, 227-6137. Students read on their own about one paper per week. Homework on simple aspects of a paper is due before the paper is discussed in class. The discussion in class is normally intended to show some applications of the concepts and techniques of a paper and to answer questions posed by the students on the more difficult issues. Detailed explanations of the underlying theory are generally motivated only by direct questions of a student. The instructor may ask a student to try to answer the question of another student. Students receive substantial credit when they actively participate in this activity.

To do well in this course a student must: 1) understand the overall motivation of each paper, 2) know the techniques presented in the paper, and 3) be able to apply these techniques to practical situations. A typical question in each category is, for example: 1) Why program correctness with Dijkstra's method is called *total*, 2) Explain point number 3 of Gries' 5-point checklist of a loop, and 3) Prove with Hoare's method the correctness of a piece of code.

The homework is an essential component of the educational process. The instructor provides a complete detailed solution of each assigned problem and discusses it in class on the day the homework is due. No late homework is consequently accepted. There are a midterm and a final exams. Depending on the class size and the students' participation, a portion of the midterm and/or final exams might be taken at home and the use of some automated tools for software specification and verification will contribute to the final grade.

The most appropriate form of interaction with the instructor is in person or by phone during office hours. The proposed time of office hours will be re-negotiated with the students on the first day of class.

pto →

Reading List

1. S. Hantler and J. King, An Introduction to Proving the Correctness of Programs, *Computing Surveys*, **8**–3:332–353, Sept. 1976.

2. C. A. R. Hoare, An Axiomatic Basis for Computer Programming, *Comm. of the ACM*, **12**–10:576–580,583, Oct. 1969.

3. E. W. Dijkstra, Guarded Commands, Nondeterminacy and Formal Derivation of Programs, *Comm. of the ACM*, **18**–8:453–457, Aug. 1975.

4. C. A. R. Hoare, Proof of Correctness of Data Representation, *Acta Informatica*, **1**:271-281, 1972.

5. D. Gries, An Illustration of Current Ideas on the Derivation of Correctness Proofs and Correct Programs, *IEEE Trans. on Soft. Eng.*, **2**–4:238-244, Dec. 1976.

6. J. V. Guttag, E. Horowitz, and D. R. Musser, Abstract Data Types and Software Validation, *Comm. of the ACM*, **21**–12:1048–1064, Dec. 1978.

7. R. S. Boyer and S. Moore, Proving Theorems About LISP Functions, *Journal of the ACM*, **22**–1:129–144, Jan. 1975.

8. S. Antoy and J. Gannon, Using Term Rewriting to Verify Software, *IEEE Transactions on Software Engineering*, **20**-4:259–274, 1994.

9. S. J. Garland and J. V. Guttag, A Guide to LP, The Larch Prover, Nov. 20, 1991.