3 Teaching Summary

3.1 Teaching Overview

My teaching experience includes undergraduate and graduate courses taught both at the Technical University of Munich, Germany, and the State University of New York at Stony Brook, USA. At Stony Brook I have taught the following core undergraduate courses (CSE 3xx level):

- CSE 304: Compiler Design, in Fall 2005–2001
- CSE 305: Principles of Database Systems, in Fall 2000
- CSE 307: Principles of Programming Languages, in Spring 2002
- CSE 315: Database Transaction Processing Systems, in Spring 2007, 2005
- CSE 350: Theory of Computation: Honors, in Spring 2008

the following core graduate courses (CSE 5xx level) and advanced graduate courses (CSE 6xx level):

- CSE 504: Advanced Compiler Design, in Spring 2007, 2004–2003
- CSE 510: Hybrid Systems, in Spring 2009, 2006 (new course)
- CSE 515: Advanced Database Transaction Processing Systems, in Fall 2008, 2006–2005
- CSE 548: Analysis of Algorithms, Fall 2009
- CSE 549: Introduction to Computational Biology, in Fall 2008
- CSE 625: Computer Aided Verification, in Spring 2003, 2001 (new course)
- CSE 637: Program Semantics and Verification, in Fall 2004

and organized the following weekly graduate seminars (CSE 6xx level):

- CSE 643: Seminar in Concurrency, in Fall 2006, Spring 2002
- CSE 647: Design and Analysis Research Seminar, in Spring 2004, Fall 2002

The (ongoing) CSE 647 seminar features research talks by Stony Brook faculty, and occasionally graduate students, and guest speakers from around the world. At Technical University of Munich I taught, in conjunction with an associate/full professor, the following core undergraduate courses:

- Practical Project Course in Programming, Spring 1998, (with M. Broy, B. Brügge)
- Introduction to Computer Science II, Spring 1997, (with M. Broy)
- Introduction to Computer Science IV, Spring 1995, (with M. Broy)
- Communication for Computer Scientists, Fall 1997, (with Lang, Witmann)

the following core graduate courses:

- Principles of Program and System Development, Fall 1997–1995, (with M. Broy, T. Nipkow)
- Distributed Systems, Spring 1997, (with M. Broy)

and organized the following weekly graduate seminar:

- Object-Oriented Software Development, Spring 1994, (with M. Broy).

I currently advise or co-avise one Post Doctoral student, five Ph.D. students, and four M.S. students. I have graduated five PhD sudents and twenty five Master students. I was consistently ranked among the most popular teachers in the computer science departments of Stony Brook and TU Munich, and have consistently received one of the highest teaching ratings of these departments.

3.2 Teaching Methods

I consider teaching as one of the most exciting aspects of the academic career, and the opportunity to teach, played an important role in my decision to pursue an academic path. The university campus, as a forum for inter-cultural dialog, where knowledge and critical thinking is shared by teachers with their students, and enthusiasm and fresh ideas are shared by students with their teachers, has always fascinated me.

While striving for universality is human, truth, even in science, is a social consensus, which evolves, as societies evolve. My teaching (and research) philosophy is that students are my allies in the quest for scientific truth. The following joke, which I like to share with my students, is a good illustration of this philosophy. One teacher says to another:

> Today I had such a silly student in my class. I explained a new concept, and the student did not understand it. I explained it for the second time, and the student still did not understand it. I explained it for the third time, and I understood it even myself, but the student still did not understand it.

The point I want to make is that scientific truth has various facets, and it is often the case that we miss some of these facets, even after (or precisely because of) years of experience. These ignored facets, which a fresh mind might highlight, enrich our scientific understanding, and may be the seeds for a new discovery.

For all these reasons, I strongly encourage class interaction, no matter how large the size of the class is, and I make a conscious effort to learn the students name. If students do not ask questions during a class, I ask questions myself, to probe their level of understanding, and adapt my teaching pace, accordingly. I believe that courses have to be taught in a clear and captivating manner, such that students acquire the new concepts during the class, and only deepen them at home. I assign hard, but exciting, homework projects, emphasizing either a new facet or the practical application of a theoretical idea, and I encourage students to solve extra credit questions.

My open and friendly attitude, conveys enthusiasm and respect for students. This leads to very high attendance rates, of both my teaching classes and the associated office hours.

3.3 Curriculum Development

CSE 625: Computer Aided Verification. This is a *new course* I have designed to cover advanced topics in computer aided verification, and in particular, the state of the art of model checking. The course draws from conference papers and similar classes offered at Berkeley, CMU, NYU, Rice, Stanford and UPenn.

An important focus of the course is the term project, which can be done individually or in groups of two. Students are encouraged to advance their own research interests in their project by choosing between a research topic, a survey, a tool comparison, a case study or an implementation of a particular algorithm in jMocha, the model checker we jointly developed at Berkeley, UPenn and Stony Brook. Typically, two projects per semester become published papers in some form.

Another important focus of the course are the homeworks, which are offered on a weekly basis. They deepen students understanding of the theory on the one hand and their ability of using a

model checker (usually jMocha) to specify requirements, model and verify concurrent systems on the other hand. The main topics covered in the class include:

- *Computer aided verification:* Overview, requirements, models and model checking.
- *Modeling language:* Reactive modules (RM), operations on RMs, examples.
- *Invariant verification:* Transition graphs, invariants, graph traversal, state explosion and compositional reasoning.
- *Symbolic graph search:* Symbolic invariant verification and binary decision diagrams (BDDs).
- *Graph minimization:* Graph partitions, partition refinement, reachable partition refinement
- *Temporal safety requirements:* State logics, safe temporal logic STL, model checking, distinguishing and expressive power.
- *Automata-theoretic safety verification:* Automata, safe automata logic SAL, operations on automata, model checking.
- *Hierarchical verification:* Trace semantics, compositionality, asssume-guarantee, simulation relations, homomorphisms.
- *Fair modules:* Omega-languages, safety versus liveness, fairness, fair modules, examples.
- *Response verification:* Searching for fair cycles and response verification.
- *Temporal liveness requirements:* CTL, μ-calculus, and symbolic model checking.
- *Automata-theoretic liveness verification:* Theory of omega-regular languages.
- *Linear temporal logic:* Linear temporal logic LTL.
- *ESDA formalisms:* MSCs, hierarchical modules, and UML.

CSE 637: Program Semantics and Verification. Originally designed to cover Hoare logic for structured programs, I have completely restructured the curriculum of this course to cover recent advances in automated software analysis. The course draws mainly from conference papers, but similar courses offered at NYU and Kansas State University are also considered.

Students are encouraged to advance their own research interests by choosing a project addressing a research topic, a survey, a tool comparison, a case study or an implementation of a particular algorithm. Students are also asked to prepare the solution of homeworks, which are offered on a weekly basis. These deepen students understanding of the theory on the one hand and their ability of using a software model checker to specify requirements, and verify concurrent software systems on the other hand. Each class had also associated a scribe, responsible for creating a powerpoint presentation of the slides. The main topics covered in the class include:

- *Computer aided software analysis:* Overview, requirements specification, reactive programs, compilation process, program abstraction, data-flow and control-flow graphs, program analysis, and software model checking.
- *Abstract interpretation:* Syntax and Semantics of programs, static semantics, consistent abstract interpretation, lattice of abstract interpretations, abstract evaluation of programs, Fixpoint approximation methods, widening and narrowing.

- *Predicate abstraction:* parallel systems, predicate transformers, abstract semantics of programs, abstract state lattice, abstract transitions, computing abstract successors, abstract state space exploration, complexity, abstract state graph construction, state graph refinement, implementation, reachability algorithm, applications.
- *Abstraction refinement:* Overview of the SLAM toolkit, the predicate abstraction tool C2BP, the model checking tool for boolean programs BEBOP and the refinement tool NEWTON
- *Automated predicate abstraction of C Programs:* Challenges of predicate abstraction in C, weakest precondition and cubes, pointers and aliasing, predicate abstraction for assignments, gotos and conditionals, and procedure calls, soundness and completeness, the C2BP tool.
- *Model checking boolean programs:* Hierarchic state machines, push-down automata and boolean programs, statements, variables and scope, construction of the control flow graph, trace semantics, reachability via interprocedural dataflow analysis, path edges, summary edges, transfer functions, shortest trajectories, optimizations, the BEBOP tool.
- *Counterexample driven refinement:* Explaining infeasible paths, path simulation via strongest postconditions, abstract explanations, basic path simulation, procedures and pointers, implementation in the NEWTON tool, experiments.
- *Transition predicate abstraction:* Automata-theoretic verification of concurrent programs, liveness and fair termination, abstract transition programs, transition predicates, automated abstraction, just termination, compassionate termination, lexicographic completeness.

CSE 504: Advanced Compiler Design. This course, which I have considerably restructured over the years, covers classic topics in compiler design, with a special emphasis on recent advances in program analysis. The course is based on two books: "Compilers: Principles, Techniques and Tools", by A.V. Aho, R. Sethi and J.D. Ullman, and "Advanced Compiler Design and Implementation", by S.S. Muchnick.

An important focus of the course is the term project, which consists of a complete implementation of a compiler for a Java-like language. The implementation includes 5 or 6 sub-projects, each having allocated between 2 and 3 weeks. The project can be done either individually or in groups of two, and it is an essential tool both for a deep understanding of theory and for a proficient use of compiler-compiler technology. The main topics covered in the class include:

- *Lexical analysis and regular languages:* Regular expressions (RE), finite automata (FA), equivalence of RE and FA, minimization of FA, specifying RE in a compiler-compiler.
- *Syntax analysis and context-free languages:* Context-free grammars (CFG), analysis of CFGs, and transformation of CFGs, push-down automata, top-down parsing, bottom-up parsing, specifying CFGs in a compiler-compiler.
- *Syntax-directed translation:* Syntax directed definitions, construction of syntax trees, translation schemes, top down and bottom up translation.
- *Type checking and and non-context-free languages:* Type systems, specification of a simple type checker, equivalence of type expressions, type conversions, polymorphism.
- *Run-time environments:* Source language issues, storage organization, storage allocation strategies, access to nonlocal names, parameter passing, symbol tables.

- *Intermediate code generation:* Intermediate languages, syntax-directed translation of declarations, assignment statements, boolean expressions, and case statements into three-address code, backpatching, procedure calls.
- *Code generation:* The target machine, run-time storage management, basic blocks and flow graphs, next-use information, register allocation and assignment, peephole optimization, dynamic programming and code-generation, code-generator generators.
- *Abstract interpretation and code optimization:* Sources of optimization, optimization of basic blocks, loops in flow graphs, data flow analysis, code improving transformations, dealing with aliases, data-flow analysis of structured flow graphs, efficient data-flow algorithms, data-flow analysis and abstract interpretation, estimation of types.

CSE 510: Hybrid Systems. This is a *new course* that I have designed to cover the state of the art in modeling, analysis and control of embedded systems. The course is at the intersection between automata theory and control theory and draws from conference papers and similar classes offered at Berkeley, CMU, Stanford and UPenn.

An important focus of the course is the project, which can be done individually or in groups of two. Students are encouraged to advance their own research interests in their project by choosing between a research topic, a survey, a tool comparison, a case study or an implementation of a particular algorithm. Typically, two projects per semester become published papers in some form.

Another important focus of the course are the homeworks, which are offered on a weekly basis. They deepen students understanding of the theory on the one hand and their ability of using MATLAB to specify requirements, model and verify embedded systems on the other hand. The main topics covered in the class include:

- *Signals and systems:* Discrete and continuous signals, discrete and continuous transducive (or combinational) systems, discrete and continuous reactive (or sequential) systems, finite and infinite memory, relation between transducive and reactive systems, types of composition, (well formed) block diagrams, causality and existence of fixpoints,.
- *State machines:* Finite and infinite state machines, nondeterminism, modeling systems with state machines, Mealy and Moore machines, block and transition diagrams, executions.
- *Composing state machines:* Product, cascade, feedback, synchronous and asynchronous composition, deadlock and receptiveness, delay and existence of fixpoints, reachability.
- *Properties of state machines:* Input-output description and equivalence, state-space description and bisimulation, equivalence and bisimulation, minimization, nondeterminism, refinement and simulation, receptiveness, determinization, output deterministic state machines, minimization of nondeterministic state machines.
- *Discrete linear systems:* Difference equations, linear algebra and matrices, existence of solutions, state-space and input-output description, siso and mimo systems, superposition principle, time-invariant systems, input response, transfer function, general solution.
- *Continuous linear systems:* Differential equations, relation to difference equations, higher order differential equations and state-space description, existence of solutions, uniqueness,

input-output description and transfer function, time-invariant systems, siso and mimo systems, eigenvalues and eigenvectors, general solution, similarity transformations, qualitative behavior of 2nd order LTI, classification of equilibrium points.

- *Z transform:* Motivation, definition and inverse, existence of ZT, table of Z-transforms, properties of ZT, partial fractions, poles, convolution, application of ZT, transfer function and similarity transformation.
- *Laplace transform:* Motivation, definition and inverse, existence of LT, table of Laplace transforms, properties of LT, partial fractions, poles, application of LT, transfer function and similarity transformation.
- *Controllability, observability and realizability:* Controllable systems, controllability of LTI systems, controllability matrix and Gramians, minimal controllable LTI, observability, observability matrix and Gramian, observability of LTI, minimal observable LTI, realizability.
- *Nonlinear systems:* Nonlinear models and nonlinear phenomena, phase plane and phase portraits of 2nd order nonlinear systems, finite escape time, multiple isolated equilibria, limit cycles, bifurcations, multiple modes of behavior, piecewise linearization and hybrid automata.

CSE 515: Database Transaction Processing Systems. This course covers classic topics in database transaction processing systems, with a special emphasis on recent advances due to Internet technology. The course is based on the "Database Systems: An Application-Oriented Approach" book by M. Kifer, A. Bernstein and P. Lewis. The slides of the course were however considerably rewritten, to better emphasize the main ideas and to support an interactive presentation.

An important focus of the course is the term project, which consists of 5 or 6 sub-projects, each having allocated between 2 and 3 weeks. The project can be done either individually or in groups of two, and it is an essential tool both for a deep understanding of theory and for a proficient use of DB transaction-processing technology. The main topics covered in the class include:

- *ACID properties of transactions:* Transaction processing (TP), TP manager (TPM), consistency, atomicity, durability, isolation, the ACID properties of transactions.
- *Models of transactions:* Flat transactions, providing structure within a transaction, structuring an application as multiple transactions with chains, sagas, multilevel transactions, workflows and declarative transaction demarcation..
- *Implementing isolation:* Schedules and schedule equivalence, recoverability, cascaded aborts and strictness, models for concurrency control, immediate-update pessimistic controls (IUPC), design of IUPCs, objects and semantic commutativity, compensating operations, isolation in structured transaction models, other concurrency controls.
- *Isolation in relational databases:* Conflicts in a relational DB, locking and the SQL isolation levels, granular locking with intention and index locks, tuning transactions, multiversion concurrency controls.
- *Atomicity and durability:* Crash, abort and media failure, immediate update systems and write-ahead logging, recovery in deferred-update systems, recovery from media failure.

- *Architecture of transaction processing systems:* TP in a centralized system, TP in a distributed system, the TP monitor, global atomicity and the TPM, remote procedure call (RPC), peer-to-peer communication (P2PC), , event communication (EC), storage architectures, TP on the Internet, web application servers.
- *Implementing distributed transactions:* Implementing the ACID properties, atomic termination, transfer of coordination, distributed deadlock, global serialization, replicated DBs, distributed transactions in the real world.
- *Web services:* Web basics, hypertext transfer protocol, SOAP and message passing, WSDL and the specification of services, BPEL and the specification of business processes, UDDI and the publish-discovery of service information, WS-coordination.
- *Security and electronic commerce:* Authentication, authorization and encryption, digital signatures, key distribution and authentication, authorization, authenticated RPC, electronic commerce, the secure sockets layer protocol, passport and single sigh-on, keeping credit card numbers private.

CSE 304: Compiler Design. This course is the undergraduate version of CSE 504, Advanced Compiler Design. Compared to CSE 504, CSE 304 puts much more emphasis on the front-end aspects of a compiler and the term project is modified accordingly.

CSE 305: Principles of Database Systems. This introductory course covers the classic topics in database systems. The course is based on the book "Database Systems: An Application-Oriented Approach" by M. Kifer, A. Bernstein and P. Lewis. The slides of the course were however considerably rewritten, to better emphasize the main ideas and to support an interactive presentation.

An important focus of the course is the term project, which consists of a complete implementation of a database for a particular application. The implementation includes 4 sub-projects, each having allocated between 2 and 3 weeks. The project can be done either individually or in groups of two, and it is an essential tool both for a deep understanding of theory and for a proficient use of DB technology. The main topics covered in the class include:

- *Overview of databases and transactions:* What are databases (DB) and transactions processing systems (TPS), features of modern DB and TPS, major players in DB and TPS, decision support systems.
- *The entity relationship model:* Conceptual ER modeling, entities and types, relationships and relationship types, advanced features in conceptual data modeling, from ER diagrams to relational DB schemas, UML.
- *The relational model:* The data model, the relational model, SQL data definition sublanguage.
- *Relational algebra and SQL:* Relational algebra, the query sublanguage of SQL, modifying relation instances in SQL.
- *Relational normalization theory:* Independence, decompositions, functional decomposition, properties of functional dependencies, normal forms, properties of decompositions, BCNF decomposition, synthesis of 3NF schemas, the fourth normal form.
- *Triggers and active databases:* Triggers, semantic issues in trigger handling, triggers in SQL, avoiding chain reaction.

- *SQL in the real world:* Embedded SQL, more on integrity constraints, dynamic SQL, JDBC and SQLJ, ODBC, comparison.
- *Physical data organization and indexing:* Disk organization, heap files, sorted files, indices, multilevel indexing, hash indexing, special-purpose indices,
- *The basics of query processing:* Overview, external sorting, computing projection union and set difference, computing selection, computing joins, computing aggregate functions.

CSE 307: Principles of Programming Languages. This introductory course covers the most important programming language constructs present in modern languages. For each construct it discusses its merits, theoretical foundation and implementation in a programming language. The course is based on the book "Programming Languages: An Interpreter-Based Approach" by N. Ramsey and S.N. Kamin, an taught at Harvard and University of Illinois at Urbana.

An important focus of the course is the term project, which includes 7 sub-projects, each having allocated 2 weeks. Each sub-project is programming exercise which reinforces the understanding of a particular programming language construct. The main topics covered in the class include:

- *Imperative core:* The imperative core language, abstract syntax, environments, operational semantics, the interpreter.
- *Functional programming:* μScheme, recursive programming with lists, programs as data, first-class functions, continuation passing.
- *Automatic memory management:* Garbage collection (GC) basics, the managed heap in μScheme, mark-and-sweep GC, copying GC, debugging a GC, reference counting.
- *Type systems:* Typed impcore, an interpreter with type-checking, arrays, type soundness, type constructors, polymorphism.
- *CLU and data abstraction:* Data abstraction, the language, implementation, Clu as it really is, program verification.
- *Smalltalk and object-oriented programming:* The initial basis, implementations of predefined objects, interpreter and operational semantics, Smalltalk as it really is.
- *Prolog and logic programming:* Logic as a programming language, the language, implementation, Prolog as it really is, prolog and mathematical logic.

CSE 315: Database Transaction Processing Systems. This course is the undergraduate version of CSE 515, Advanced Database Transaction Processing Systems. Compared to CSE 515, CSE 315 puts much more emphasis on the classic aspects of DB transaction processing systems and the term project is modified accordingly.

CSE 350: Theory of Computation: Honors. This is a *new honors course* that I have designed to cover the state of the art in the theory of computation. The course draws from similar classes offered at MIT, Berkeley, CMU and Stanford.

An important focus of the course are the homeworks, which are offered on a weekly basis. They deepen students understanding of the theory during the class. The main topics covered include:

- *Mathematical Background:* This part includes an introduction to the theory of computation and associated mathematical notions and terminology, such as functions and relations, graphs, strings, languages, and boolean logic, theorems and proofs.

- *Finite Automata and Regular Languages:* This part covers finite automata and their formal definition, regular languages and ondeterministic automata, quivalence of regular expressions and finite automata, and the pumping lemma for regular languages.

- *Pushdown Automata and Context-Free Grammars:* This part covers context-free grammars and languages, properties of context-free grammars, normal forms, pushdown automata, equivalence of pushdown automata with context-free grammars, and the pumping lemma for context-free languages.

- *Turing Machines:* This part covers Turing machines and their variants, the definition of an algorithm, computatbility theory, decidable problems, the halting problem and the diagonalization method, reducibility as a method for proving unsolvability, and recursive functions.

- *Computation Complexity:* This part covers measuring computation complexity, class P, class NP, NP Completness.

CSE 653: Seminar in Concurrency. This seminar features research talks by the instructor and graduate students. In Spring 2002 the seminar was dedicated to the study of embedded systems. The main topics covered included: synchronous languages, hardware/software codesign and hybrid (mixed analog/digital) systems.

CSE 657: Design and Analysis Research Seminar. The (ongoing) CSE 657 seminar features research talks by Stony Brook faculty, and occasionally graduate students, and guest speakers from around the world.

3.4 Supervision of Students

I actively contribute to the educational and research missions of the university through supervision of graduate and undergraduate students. Supervision of students is also personally rewarding. Every week, I meet individually with each of my students, to discuss problems, solutions, future plans, etc. My goals are to make good progress on current projects and to train students to be independent thinkers and researchers. This requires walking a fine line between giving too much and too little guidance. It requires continuous sensitivity to each student's personality, ability, and experience. I continually try to improve my skill at this.

PhD and Post Doctoral Students. I advised five PhD students that have successfuly graduated: Shiyong Lu, with PhD thesis "Semantic Correctness of Transactions and Workflows", Ziyang Duan, with PhD thesis ''Automatic Verification and Synthesis of Web Service-based Workflows", Pei Ye, with PhD thesis "Modeling Excitable Cells Using Hybrid Automata", Ezio Bartocci with PhD thesis "A Formal Framework for Modeling, Simulating and Analyzing Networks of Excitable Cells", and Oliviero Riganelli with PhD thesis "Designing Optimal Controllers for MANETs". Shiyong graduated in 2002, Ziyang in 2007, Pei in 2008, Ezio in 2009 and Oliviero in 2010.

I am currently advising one Post Doctoral sudent, Ezio Bartocci, and five PhD students: Sumit Jain, Xiaowan Huang, Abhishek Murthy, Justin Syester, and Zhichao Li. Sumit started to work with me in Fall 2005 and he is expected to graduate in Fall 2011. His thesis focuses on "Open Source Software Model Checking". Xiaowan started to work with me in Fall 2003 and he is expected to graduate in Fall 2010. His thesis focuses on "Guided Monte Carlo Software Model Checking". Justin and Abhishek started to work with me in Fall 2009, and Zhichao in Spring 2010. All these students have made very good progress and they are all expected to finish their PhD in time. Sumit joined Intel but he works hard towards the completion of his thesis.

MS Students. I have supervised and currently supervise the research projects for a considerable number of M.S. students including international students: Michell Baeten came from the Technical University of Eindhoven to work and complete her thesis with me in 2005-2006; Robert Schmohl from the Technical University of Munich between 2004-2005. The other supervised students are as follows: Stephan Breutel and Andrea Krause at the Technical University of Munich, between 1995-1996; Tobias Mayr at the TU of Munich between 1996–1997; Christian Stieber at the TU of Munich between 1996–1997; Xie Wen, Yingrou Chen and Zhiping Qiu at Stony Brook, between 2000–2001; Divyangi Anchan, Jyoti Wagholikar and Haibo Hu between 2001–2002; Khushru K. Bamji, Harshad Kamat, Delia Paval (Joined Reuters Inc.), Jaspreet Singh, Vaishali R. Wani, Qinghua Zhang and Jing Zhang, between 2002–2003; Wai Chong, Puloma Mukherjee and Krishnakumar N. Nair, between 2003-2004; Ganesh Prabhu, Seyasachi Pradhan and Mike True, between 2004-2005; Densel Santhmayor and Sugabrahmam Giridharan since 2005; Wenkai Tan between 2003–2007.

Undergraduate Students. As the Computer Science Department's Undergraduate Information Systems Program Director, I helped and advised numerous information systems undergraduate students in their academic career.

I prepare carefully for each exam, reading the student's report, thesis proposal, or dissertation, as appropriate. Students deserve a good working environment. Together with Professor Liu and Stoller I have created and equipped a new lab, the Design and Analysis Research (DAR) Lab. This required considerable renovation and is now a good place for students to work and share ideas.

3.5 Course Evaluations

Table 1 and Table 2 summarize the course evaluations I obtained for the courses I taught between Fall 2000 and Spring 2006. As it can be seen, they are much better than the department averages.

3.6 Student Feedback from Course Evaluations

This section includes selected feedback from anonymous course evaluations. All quoted responses are unedited and complete.[1].

[1] I am happy to provide photocopies of all written responses

	WELL PRE-PARED	SUBJ CLEAR	ENTHU-SIAS-TIC	CON-CER-NED	AVAIL-ABLE	OBJECT-IVES CLEAR	WORK CHALL-ENGE	EVALU-ATES FAIR	RECO-MMEND	LEAR-NED MORE
CSE 305 F00	**1.80**	**2.45**	**2.25**	**2.25**	**1.95**	**2.50**	**2.45**	**2.20**	**2.26**	**2.75**
dept average	2.15	2.69	2.56	2.53	2.34	2.68	2.28	2.56	2.76	2.91
univ average	1.77	2.13	2.01	2.01	1.89	2.13	2.04	2.02	2.19	2.47
CSE 635 S01	**1.36**	**1.91**	**1.09**	**1.09**	**1.36**	**1.91**	**2.45**	**1.27**	**1.90**	**2.00**
dept average	1.98	2.39	2.29	2.23	2.02	2.35	2.10	2.28	2.45	2.64
univ average	1.69	2.00	1.91	1.90	1.80	2.02	1.93	1.94	2.09	2.33
CSE 304 F01	**2.00**	**1.89**	**1.33**	**1.67**	**2.22**	**2.78**	**1.56**	**1.89**	**2.37**	**2.22**
dept average	1.77	2.37	2.21	2.10	1.93	2.31	1.93	2.12	2.37	2.50
univ average	1.72	2.07	1.94	1.92	1.81	2.07	1.97	1.95	2.13	2.39
CSE 307 S02	**3.10**	**4.33**	**3.15**	**3.30**	**3.05**	**3.95**	**3.15**	**2.95**	**4.20**	**4.68**
dept average	1.84	2.29	2.15	2.13	1.96	2.24	1.98	2.10	2.31	2.46
univ average	1.67	1.99	1.88	1.89	1.80	2.00	1.93	1.91	2.08	2.30
CSE 304 F02	**1.40**	**1.90**	**1.30**	**1.90**	**1.89**	**2.40**	**1.60**	**1.22**	**1.90**	**1.89**
dept average	1.91	2.31	2.21	2.17	1.99	2.32	2.04	2.14	2.33	2.51
univ average	1.76	2.10	1.97	1.97	1.87	2.10	2.02	1.97	2.18	2.42
CSE 635 S03	**1.64**	**1.71**	**1.36**	**1.50**	**1.57**	**2.00**	**1.71**	**1.64**	**1.50**	**1.77**
dept average	1.88	2.28	2.08	2.03	1.97	2.25	2.02	2.10	2.28	2.39
univ average	1.68	1.95	1.84	1.83	1.76	1.97	1.90	1.86	2.03	2.25
CSE 504 S03	**1.85**	**1.77**	**1.73**	**1.70**	**1.80**	**2.20**	**2.16**	**1.68**	**2.12**	**2.58**
dept average	1.88	2.28	2.08	2.03	1.97	2.25	2.02	2.10	2.28	2.39
univ average	1.68	1.95	1.84	1.83	1.76	1.97	1.90	1.86	2.03	2.25
CSE 304 F03	**2.14**	**2.43**	**2.00**	**2.86**	**2.00**	**2.43**	**2.14**	**1.86**	**3.14**	**2.86**
dept average	1.94	2.33	2.14	2.11	1.97	2.32	2.09	2.14	2.42	2.59
univ average	1.69	2.01	1.90	1.90	1.81	2.02	1.96	1.91	2.08	2.31

Table 1: My teaching evaluations for F2000–F2003. 1=Strongly agree. 7=Strongly disagree.

	WELL PRE-PARED	SUBJ CLEAR	ENTHU-SIAS-TIC	CON-CER-NED	AVAIL-ABLE	OBJECT-IVES CLEAR	WORK CHALL-ENGE	EVALU-ATES FAIR	RECO-MMEND	LEAR-NED MORE
CSE 504 S04	**1.42**	**1.63**	**1.52**	**1.48**	**1.55**	**1.88**	**1.75**	**1.44**	**1.69**	**1.93**
dept average	1.70	2.15	2.05	2.00	1.86	2.20	1.97	2.04	2.21	2.39
univ average	1.65	1.94	1.81	1.82	1.75	1.96	1.90	1.86	2.01	2.22
CSE 637 F04	**1.33**	**2.17**	**1.67**	**1.17**	**1.33**	**1.17**	**1.67**	**1.17**	**1.50**	**2.00**
dept average	1.76	2.20	2.01	1.98	1.86	2.18	2.05	1.99	2.22	2.45
univ average	1.75	2.12	1.97	1.99	1.87	2.12	2.04	2.00	2.21	2.43
CSE 304 F04	**1.75**	**1.89**	**1.50**	**1.38**	**1.80**	**2.13**	**1.63**	**1.50**	**1.88**	**2.00**
dept average	1.76	2.20	2.01	1.98	1.86	2.18	2.05	1.99	2.22	2.45
univ average	1.75	2.12	1.97	1.99	1.87	2.12	2.04	2.00	2.21	2.43
CSE 315 S05	**3.00**	**3.00**	**2.85**	**3.25**	**3.14**	**3.50**	**2.58**	**2.73**	**3.75**	**3.67**
dept average	1.75	2.02	1.84	1.79	1.76	2.09	1.90	1.94	2.04	2.33
univ average	1.68	1.96	1.83	1.83	1.76	1.97	1.92	1.87	2.03	2.24
CSE 515 F05	**1.59**	**1.73**	**1.59**	**1.76**	**2.11**	**1.86**	**2.55**	**1.52**	**1.73**	**2.21**
dept average	1.76	2.20	2.01	1.98	1.86	2.18	2.05	1.99	2.22	2.45
univ average	1.71	2.05	1.90	1.93	1.81	2.06	1.99	1.94	2.13	2.33
CSE 304 F05	**2.33**	**1.33**	**1.50**	**1.17**	**1.40**	**1.83**	**1.33**	**1.00**	**1.17**	**1.50**
dept average	1.71	1.99	1.85	1.73	2.00	1.92	1.75	1.97	2.23	1.89
univ average	1.71	2.05	1.90	1.93	1.81	2.06	1.99	1.94	2.13	2.33
CSE 510 S06	**1.83**	**1.33**	**1.67**	**1.50**	**2.00**	**1.50**	**1.83**	**2.00**	**1.50**	**1.50**
dept average	1.68	1.93	1.79	1.69	1.66	1.96	1.80	1.74	1.89	2.17
univ average	1.69	1.99	1.86	1.88	1.78	2.00	1.93	1.91	2.07	2.26

Table 2: My teaching evaluations for S2004–S2006. 1=Strongly agree. 7=Strongly disagree.

Student Comments on What Instructor Does Particularly Well

The following selected student comments are in response to the question "What does the instructor do particularly well?"

Everything.
Ties together various topics (even from other classes) to provide a deeper understanding of the subject and its relation to other subjects.
Prepares lectures very well.
Explains very well, relates the material and successfully takes the theory of computer science and gives it practical use. The instructor makes sure everyone understands concepts before moving on.
Covers material well, good examples.
Explains the course material well.
Interaction with students by being friendly.
Explains related background of the materials.
Teaching material.
Many good examples.
Encourage class participation.
Explains the concepts well.
Explains everything well, is enthusiastic while teaching subject.
Explains the concepts well.
Interactive question and answering.
Ask students questions frequently.
Well organized, systematic, open to students.
Prof. Grosu teaches well keeping the interest alive in the class. Takes a lot of examples in class which aids in understanding.
Convey enthusiasm for the subject.
Quizzes and tests are good. Examples for explanations are well rehearsed.
Explain the material.
Ask us questions in class, being awake and thinking the entire class.
Involves the class into lecture by calling on them.
He asks students a lot of questions.

Student Comments on Quality of the Course

The following selected student comments are in response to the question "Aside from the quality of the instructor, what is particularly good or bad about the course?"

Everything is excellent.
Helps students understand the inner workings (the Whys? and Hows?) of the languages we use.
One of the better CS classes.
Challenging homework projects that relate the material learned in class.
The course covers areas which are of practical and research interest.
Its a good course.
The course is very good as it covers theory and is practical.
The course is very interesting.
Course is Ok. Teaches a lot about compilers.
The project was a good learning experience.
Liked how he lectured and forced us to answer questions in class.
I learned more about databases..