# Statement of Teaching

Aniruddha S. Gokhale (a.gokhale@vanderbilt.edu)
Associate Professor, Dept. of EECS
Vanderbilt University, Nashville, TN 37235

July 29, 2017

## 1 Organization of the Teaching Statement

This document provides my statement of teaching. I start by describing my teaching philosophy. Next, I outline the courses I have developed, refined and taught at Vanderbilt University, first as a Research Scientist in the Institute for Software Integrated Systems (ISIS), then as a Tenure-track Assistant Professor (Fall 2003–Summer 2010), and subsequently as an Associate Professor (Fall 2010–present). I provide a summary of teaching evaluations and comments I received from students. I describe how I made the necessary adjustments to incorporate student feedback.

## 2 Teaching Philosophy

**General Principles.** My experience teaching at Vanderbilt University since Fall 2002 coupled with my experience taking undergraduate classes in India, graduate classes in the United States, and attending many tutorials at conferences has enabled me to obtain a good perspective on effective teaching skills. In particular, I was fortunate that the professors at Washington University, St. Louis took great pride in the quality of their graduate and undergraduate teaching. Additionally, to improve my understanding of interdisciplinary topics, I have also made it a point to sit in different classes taught at Vanderbilt, such as the MIC course (Prof. Nordstrom and Prof. Davis), Hybrid Systems (Prof. Xenofon Koutsoukos), Computer Systems Analysis (Prof. Larry Dowdy), Discrete Event Systems (Prof. Abdelwahed), and Optimization (Prof. McDonald, Civil). Additionally, in recent years I have also completed or in the process of completing some Massive Open Online Courses (MOOCs) in topics, such as Machine Learning, Scala Programming, and Cloud Computing, that are taught on the Coursera platform. These experiences and teaching styles of the professors I have had has enabled me to incorporate proven patterns in the courses I have been teaching at Vanderbilt University.

I firmly believe that students should be inquisitive when learning a new topic. To foster this trait, I try to add a research dimension to my courses and base them on the principles of *active learning*. I feel this makes the subject matter interesting to the students and also provides a deeper understanding of the field. Specifically, I usually require team-based class projects and presentations on a new and emerging topic in the field both of which compel students to immerse themselves in literature search and trying out new things. My teaching philosophy and course requirements have served to be useful in addressing some of the ABET requirements and outcomes (e.g., project needs, communication skills, life long skills), when Vanderbilt's Computer Science program decided to reacquire ABET accreditation.

**Post-tenure Focus.** After tenure, my aim as a teacher was to develop key courses in keeping with new technological trends. For instance, I have developed the Cloud Computing course, which after three years of being offered as a special topics course, became a regular course in the catalog. It is now offered as CS4287/5287 so that both undergraduate and graduate students can take the course. At the same time, I have also made myself available to teach a course to meet departmental needs. For instance, I taught an overflow section of CS2231 (Computer Organization). Moreover, after the retirement of Prof. Larry Dowdy, I took over the responsibilities of teaching the graduate level CS6381 (Distributed Systems), which I have been refining over the past four years to keep up with recent advances.

**Drawing Analogies.** Drawing analogies during lecturing is crucial to understanding core ideas. I have frequently made analogies to discuss key ideas. As an example, during a guest lecture I was asked to discuss the concept of software design patterns [1]. I asked the students to think of the core idea behind the use of seat belts in cars, in aircrafts, and even in roller coasters. I then drew their attention to the fact that though the context in each case was different and so was the solution, the purpose behind a seat belt is always to minimize injury in an accident. This analogy was useful in demonstrating the notion of software patterns.

In another instance, while describing the notion of a deadlock in operating systems, I referred students to the well known story of Alladin and his wicked uncle who had dropped Alladin inside a cave to fetch the magic lamp. On retrieving the lamp, Alladin asks his uncle to pull him out of the cave. The uncle insists on the lamp first and then offers to pull Alladin out. However, Alladin insists to be pulled out first and then hand over the lamp. This story describes the classic condition that results in a deadlock. Any one person relenting from their demand would indicate an approach to recover from a deadlock.

In yet another instance, I used the example of synchronized swimming and relay races to explain concepts like synchronization and rendezvous, which are important operating systems concepts.

**Using visual aids.** In the heavily systems oriented topics that my research is involved with and the corresponding courses I teach, I strongly believe in using visual aids to strengthen the understanding of the material. Tools such as ns-2 or tcpdump in the past have had a great impact in the understanding of network behavior. With increasing complexities, it is necessary to use visually appealing mechanisms. With the advent of model-driven tools that allow composing systems via visual building blocks, and also allowing visualization of the dynamics, the learning experience is substantially enhanced. Several such tools exist, such as OMNeT++ and WireShark, which I have consistently used in my courses.

**Active learning and hands-on projects.** I believe that the scientific- and concept-driven approach to teaching can be complemented with *active learning* wherein students are provided research papers and asked to critique them leading to in-class discussions on various alternatives. In-class group presentations on new research ideas pertaining to the course topic or a capstone design are other effective techniques, which I have used in my undergraduate and graduate courses.

It is hard to fully understand certain subjects by reading textbooks or research papers alone. Examples of such subjects include software engineering and design patterns, operating systems techniques, compilers, real-time systems, computer networks and protocol design, distributed systems, and cloud computing. I therefore firmly believe that hands-on experience is necessary to complement solid conceptual and theoretical coverage of these topics. I address this requirement through programming assignments and team projects, since they provide a practical and pragmatic view of the system.

I also believe that class projects must involve some research challenges to ensure students learn something new and to solve challenging problems by themselves and in small groups. Starting Spring '03, I have been providing students with sample programs I have developed for all my classes that use the tools we use for the class. This enables the students to speed up their understanding of the tools and appreciate good software design principles.

**Biweekly quizzes and problem solving for undergraduate courses.** Instead of midterm or final exams, I believe in biweekly quizzes, which helps student keep up with the reading assignments and other parts of the course. Additionally, homeworks are used as a means for problem solving, where students apply the material they have learned to answer questions on specific problems drawn from contemporary issues. Instead of a final exam, most of my courses have end-of-semester team project and in-class presentation

**Online course content distribution.** I have consistently used Vanderbilt's Blackboard online courseware system for delivering material for the courses. I also had a chance to attend the VaNTH workshop, which demonstrated the use of their course portal. This course portal allows creating reusable course modules, which can be made part of different courses. This idea is akin to building a middleware infrastructure from the desired features. Recently, I have also started using the Piazza platform for course communication, however, I have not used this platform to the fullest extent yet.

**Accessibility to Students.** On many occasions, particularly before examinations or project deadlines, students need better access to the instructor beyond the normal office hours. I have been using advanced web-enabled collaboration technologies, such as class-specific newsgroups and Vanderbilt's Blackboard system, to facilitate better accessibility to students. I have always made myself available to students to answer any of their questions. Moreover, I often address student concerns by being more accessible via other means, such as email or phone.

## 3 Courses Offered

I have had the opportunity to teach several courses both at the undergraduate and graduate level. In the following I list the courses I have offered with the recent ones listed first. Wherever applicable, I mention the motivation for offering the course.

1. **CS4287/5287, Principles of Cloud Computing:** CS4287/5287 is available to both undergraduate and graduate students. The course deals with the principles and technologies that makes the Cloud work. This course was initially a special topics course that I developed and has since become a catalog course. I have or will be offering this course in the following semesters:
   - *Spring 2011:* Enrollment 19 (1 ugrad, 18 grad)
   - *Spring 2013:* Enrollment 36 (18 ugrad, 18 grad)
   - *Fall 2014:* Enrollment 22 (9 ugrad, 13 grad)
   - *Fall 2015:* Enrollment 29 (13 ugrad, 16 grad)
   - *Fall 2016:* Enrollment 29 (16 ugrad, 13 grad)
2. **CS6381, Distributed Systems Principles:** CS6381 is a graduate-level core course for the Distributed and Networked Systems area of specialization, which deals with the challenges and solutions that make distributed systems work. The course is also available to undergraduates with advanced standing. I offered this course in the following semesters:
   - *Fall 2012:* Enrollment 22 (all grad)
   - *Spring 2014:* Enrollment 13 (all grad)
   - *Spring 2015:* Enrollment 21 (all grad)
   - *Spring 2016:* Enrollment 13 (all grad)
   - *Spring 2017:* Enrollment 24 (all grad)

3. **CS2231 Computer Organization:** I offered an overflow section of Computer Organization, which is a required course for undergraduates in sophomore standing. I offered this course in the following semesters:
   - *Spring 2015:* Enrollment 31 (all ugrad)
4. **CS292 Special Topics: Web Programming for Distributed Systems:** I offered a special topics course that explored the use of contemporary web technologies, such as JavaScript-based frameworks, to build distributed systems. I offered this course in the following semesters:
   - *Spring 2015:* Enrollment 38 (23 ugrad, 15 grad)
5. **CS6387, Topics in Software Engineering:** CS6387 is a graduate-level core course that explores different and contemporary topics in Software Engineering. The course is also available to undergraduate students in advanced standing. I offered this course in the following semesters:
   - *Spring 2011:* Enrollment 16 (3 ugrad, 13 grad)
   - *Spring 2012:* Enrollment 16 (2 ugrad, 14 grad)
6. **CS3860, Undergraduate Research,** where undergraduate students work on their undergraduate research topics under the supervision of a faculty. I have consistently had undergraduates work under my supervision on undergraduate research.
7. **CS6390, Independent Study,** where graduate students pursue independent study under a faculty's supervision. I have consistently had graduate students do independent study under my supervision over the past 14 years.
8. **CS 3281, Principles of Operating Systems I:** CS3281 is a mandatory course for junior and higher level students in Computer Science and Engineering, which deals with the principles of operating system design and implementation. I offered this course in the following semesters:
   - *Spring 2003:* Enrollment 19 (19 ugrad, 0 grad)
   - *Fall 2003:* Enrollment 46 (43 ugrad, 3 grad)
   - *Fall 2004:* Enrollment 33 (29 ugrad, 4 grad)
   - *Fall 2006:* Enrollment 16 (16 ugrad, 0 grad)
   - *Fall 2007:* Enrollment 15 (12 ugrad, 3 grad)
   - *Fall 2008:* Enrollment 24 (22 ugrad, 2 grad)
   - *Fall 2009:* Enrollment 21 (19 ugrad, 2 grad)
   - *Fall 2010:* Enrollment 19 (17 ugrad, 2 grad)
   - *Spring 2010:* Enrollment 30 (27 ugrad, 3 grad)
   - *Fall 2013:* Enrollment 21 (19 ugrad, 2 grad)
9. **CS 3283, Computer Networks:** CS3283 is a broadening elective course in the systems area that focuses on the basics of computer networks, in particular the design and operation of protocols and the architecture of the Internet. I offered this course in the following semesters:
   - *Fall 2005:* Enrollment 18 (16 ugrad, 2 grad)
   - *Spring 2007:* Enrollment 19 (15 ugrad, 4 grad)
   - *Spring 2008:* Enrollment 15 (13 ugrad, 2 grad)
   - *Spring 2009:* Enrollment 22 (19 ugrad, 3 grad)
10. **Special Topics Courses related to Model-driven Engineering:** Throughout my stint as an assistant professor, I have offered a special topics course almost every spring semester that addressed different issues pertaining to model-driven engineering and middleware, which are my research interests. I offered the following special topics courses:

- *Spring 2004: CS 292-03/396-04, Model-driven Middleware.* Enrollment 18 (3 ugrad, 15 grad). This course was the first in a series of special topics courses related to my research. This course was available to graduate students and undergraduates in advanced standing. This course identified opportunities to bring together the power of modeling and generative technologies to maximize the benefits of next-generation middleware. Student projects contributed to the CoSMIC open source model-driven middleware tool suite that I am the research lead for.
- *Spring 2005: CS 292-01/396-02, Model Driven Networked Systems Analysis and Simulation* Enrollment 25 (11 ugrad, 14 grad). This course focused on learning and applying tools for performance analysis, simulations and empirical benchmarking. The undergraduates also participated in a Microsoft-sponsored competition on embedded systems. They utilized the embedded Windows CE-based devices supplied by Microsoft to build their projects. Students came up with innovative ideas that included using sensor technology for on-the-fly fingerprinting for forensics, emergency response for a fire evacuation for tall buildings, and rescue missions for spotting lost skiers.
- *Spring 2006: CS 396-03, QoS-Enabled Middleware* Enrollment 9 (1 ugrad, 8 grad). In this course special emphasis was laid on quality of service issues and how these can be realized in contemporary middleware systems using model-driven engineering.
- *Spring 2007: CS 396-04, Model-driven QoS Engineering* Enrollment 8 (0 ugrad, 8 grad). This course was team taught by Doug Schmidt and me. This course focused on literature survey of QoS issues in distributed systems.
- *Spring 2008: CS396-02, Automated QoS Provisioning* Enrollment 8 (0 ugrad, 8 grad). Course focused on critical review of several seminal papers on QoS provisioning in real-time systems.
- *Spring 2010: CS396-03, Real-time Systems* Enrollment 6 (1 ugrad, 5 grad). Course focuses on concepts of real-time systems, middleware for real-time programming, and reviewing seminal papers.

11. **ECE 279-01 + 353-01, Real-time Systems:** ECE 279/353 was a combined offering of a real-time systems course that introduced the students to the principles of real-time systems and emerging middleware solutions in the real-time realm, such as real-time CORBA and real-time Java. I offered this course in the following semesters:
    - *Spring 2002:* Enrollment 22 (4 ugrad for ECE292, 18 grad for ECE353)
12. **EECE203, Undergraduate Independent Study,** where Computer Engineering undergraduates do their independent study. I supervised an independent study EECE203-02 in Summer 06 (1 ugrad).

## 4 Teaching Evaluations and Refinements Made

I have made several adjustments to my teaching style and course format, based on student feedback I have been receiving over the years. During my tenure track years, I had already shown steady improvements in my teaching evaluations. My teaching evaluations post tenure have continued to be on par with departmental and school averages.

Several undergraduate students have shown faith in my teaching abilities by taking more than one course with me. Many of them also continue to request me to write them letters of recommendations for their graduate studies or serve as their reference for jobs. Many undergraduates over the years have also worked on their undergraduate research under my supervision.

During my tenure-track years, I made use of the *Small Group Analysis* offered by Vanderbilt University's Center for Teaching, which helped me significantly in my formative years as an assistant professor. A Small Group Analysis is conducted by a staff member from the Center for Teaching at around the mid point of the semester without the faculty member's presence. The staff member would then aggregate the collected data and have a one-on-one meeting with me making recommendations on areas that needed improvement, and possible ways in which these could be accomplished. I learned a great deal from these analyses and incorporated changes in my course content and logistics.

Some notable and positive comments as well as useful feedback that I have received on my evaluations of courses post-tenure are below:

- For Computer Organization: "Professor Gokhale is a great lecturer. He made Computer Organization interesting and intellectually stimulating."
- Feedback on course title: "If this class was named Web Development it would be correctly named and a fine course."
- General comment: "Dr. Gokhale is great. He is constantly sending emails to answer questions from students, really wants to make sure everyone understands what is being taught, and obviously has a great understanding of the material."
- General comment: "Really great at explaining materials to student. Very dedicated to teach student about the subject."
- General comment: "Gokhale is one of the most fair and caring professors I've had at Vanderbilt. He taught the material clearly. I got a lot out of this course."
- A feedback on assigned work: "Also, the amount of outside work in the course should be spaced out more evenly throughout the semester rather than all at the end."
- A feedback on quiz material: "The quizzes were solely a measure of how well we could memorize a topic and only infrequently actually challenged us to use our brains to do something other than regurgitate some information about a topic."

I am diligently incorporating feedback from students and making improvements every year. I also made new adjustments as described below. Some were specific to the course itself. Below I document various ideas I have pursued, which have helped students and helped improve my evaluations.

- **Bridging Real World and Modern Advances with Seminal Work from the Past:** In courses such as CS6381 (Distributed Systems), many of the seminal papers from the 80s discuss the fundamental issues in Distributed Systems. However, these papers were written in a age when today's technologies and advances did not exist and hence many of these papers are too abstract to the students. To motivate the students, therefore, I use recent papers and events as driving examples to convey the core problems to the students. Once the problem is well-understood by students, I then delve into one or more seminal papers that discuss the key ideas. Examples of such issues are consensus and state machine replication, eventual versus strong consistency. Before introducing seminal papers such as Lamport's Logical Clocks and Paxos, I have used Google's Spanner, Amazon's Dynamo and Yahoo's PNUTS as recent advances and tie them back to the seminal contributions of the past.
- **Focus on contemporary issues:** Although it is necessary to teach the fundamental concepts, it is important to illustrate these ideas in the context of contemporary issues. In my undergraduate courses, towards the end of the semester, a few lectures are devoted to student presentations wherein a team of two students researches a topic of interest, makes a 15 min presentation in class, and submits a technical report. Students have been very excited by this approach.
- **Material from MOOCS:** Since I have registered for and completed some courses on platforms such as Coursera, I also leverage some of that material as additional material for students to study. In

the overflow section of Computer Organization that I taught, I used the tools from a course called Nand2Tetris that work with a simple 8-bit machine. Using a simpler machine to write assembly code and understand organizational features was very helpful for students to get up to speed before getting into the more complex 16 bit RISC architecture described by the text book.

- **Scaffolding code for programming assignments:** To better prepare students to understand the basics, over the years I have developed a large set of sample code sets that touch upon the many different aspects of systems programming for all the courses I have taught. These sample code sets are well documented which helps the student understand the material better.
- **Use of tools:** Since most of my courses are systems-related courses, I use some of the latest tools and open source packages in class for students to experiment with. These also include simulators that help students visualize the progress of various algorithms.
- **Online notes:** For students to participate in class discussions and not be distracted due to note taking, I have maintained all my class notes and lecture material online in electronic form in Vanderbilt's Blackboard system. This makes it easier for students to prepare before class, as well as to have archival access to previous lectures and handouts. In many cases, I produce hand-written notes using my tablet, which then I make available as additional notes. Hand-written notes are essentially a rendering of ideas and concepts that I would typically show on the whiteboard.
- **Quiz solutions:** I make my quiz solutions available online within the Blackboard system so that students can refer to them when they have lost points.
- **Feedback on programming assignments:** Learning from techniques used by my adviser Dr. Doug Schmidt, my TAs and I decided to provide students with individual feedback on their assignments giving them an opportunity to resubmit the assignments after making the necessary corrections.

## 5 Post Tenure Course Development and Future Teaching Plans

Since my tenure, I have created the Cloud Computing course and significantly revamped the Distributed Systems course. I also created the Web Programming course and experimented a different style for the Computer Organization course.

### 5.1 Courses Developed/Revamped

I predominantly teach systems-level courses. Since the realm of Computer Science is always undergoing change, I am required to constantly keep myself informed of recent advances and accordingly tweak the material that I plan to teach. Accordingly, some of the assessments also need to change. Consequently, parts of my courses change from one offering to the next.

**Cloud Computing.** My goal for the Cloud Computing course is to focus on the principles that make the Cloud work. To that end, I focus more on the issues at the infrastructure level instead of application level. For instance, I cover topics like virtualization and technologies that enable autoscaling and elasticity.

In the past, I applied for grants from Amazon and Microsoft Azure, which were awarded, so that students of the course could use production Cloud environment for their hands-on experiments leading to a better understanding of the Cloud. More recently, to avoid issues with students running out of credit, I have switched to using the Horizon cloud hosted by my institute. I am also exploring the use of the NSF-funded Chameleon cloud for teaching.

**Distributed Systems.** Since this course is about algorithms for distributed systems, I want to keep students engaged in the learning of the distributed algorithms. So I have used games as an approach to teach these algorithms where students could enact in class the steps of the algorithms (e.g., token passing) and thereby observe how the algorithms actually work. Naturally, this is a very crude process and human actions do not particularly illustrate the asynchronous nature of distributed systems. This led me to develop new research ideas that will make the course more engaging to students. In this context, I have used my expertise in Model-driven Engineering to develop a modeling and automation environment for students to use as learning aid for Distributed Systems. We already have one publication on this topic. However, a lot more improvements are needed to use this platform.

**Computer Organization.** The CS faculty were contemplating removing the ECE1116 (Digital Logic Design) as a requirement. At the same time, however, we wanted to experiment covering some of the important ideas from ECE 1116 in Computer Organization. I tried out this experiment in the overflow section I taught. To that end, I used material and tools from the Nand2Tetris Coursera course to blend the topics from ECE1116 into the Computer Organization course. Student had many positive comments in their evaluation.

**Web Programming for Distributed Systems.** This course used modern web technologies, such as those based off JavaScript, to teach web programming and build distributed systems. I covered the REST architecture and both the server- and client-side frameworks.

### 5.2 Plans for the Future

**Incorporating research insights into courses and exploring new courses:** With my research now spanning the realm of Software Defined Networking (SDN), Network Function Virtualization (NFV) and Internet of Things (IoT), there is a need to develop and offer special topic courses in topics like SDN and Edge Computing for IoT. I am looking forward to offering such a course either in Spring 2018 or Fall 2018. Security is a substantial challenge in these areas, and there is a need for an advanced course that looks into the security issues in these areas. Programming and engineering these systems is yet another challenge. Moreover, increasingly systems are being realized as microservices. I surmise that I can cover some of the software engineering issues through a course like CS6387 (Topics in Software Engineering).
**Preparing for the Coursera Distributed Systems Specialization MOOC:** Recently I led the effort to submit a proposal to co-teach a Distributed Systems specialization MOOC on the Coursera platform. Our proposal, jointly co-authored with Profs. Abhishek Dubey and Taylor Johnson, was favorably reviewed and the 4-course specialization is planned to go live in Fall 2018. Thus, one of my key focus during the 2017-18 academic year will be on developing the material for the MOOC. The CS6381 experience will guide me in developing the MOOC material.

## References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995. 2