Teaching Beyond Instruction

Sridhar Chimalakonda
Guest Faculty, ***IIT Tirupati, India***
Visiting Faculty, ***IIIT Sri City, India***
Chief Research Advisor, ***FortunaPIX, India***
(+91) 9490705384
ch@iittp.ac.in
researchweb.iiit.ac.in/ sridhar_ch

The landscape of education has changed drastically in the past decade or so. Students today have access to most of the courses online taught from renowned professors and universities around the world. They also have access to websites like *wikipedia*, *quora*, *stackoverflow* for Q&A. *What does teaching mean in this context?*

I believe that teaching should inspire students beyond grades and jobs, instill foundations for life and push boundaries! *I made several experiments in the last two years of my teaching, some worked and some did not! It is exciting and challenging to be a faculty during this emerging times!*

Some Thoughts on What Makes Teaching Interesting

Goal *Can we motivate the best undergrads of IIT to solve societal challenges of India and deliver results?*

- ➳ It has the power to profoundly influence people's lives
- ➳ Probably, the best way to align with state-of-the-art and state-of-the-practice of a subject
- ➳ An opportunity to address several great challenges in collaboration with young minds, which otherwise is difficult to work alone
- ➳ I primarily work in the research area of educational technologies, computer science education and see a big opportunity in the context of improving quality in Indian education

My Thoughts on Teaching

✺ *Interact* Informal at the outside – let students play, have fun, explore but be formal inside, meaning no compromise on quality! – clear learning goals, challenging exercises & *high quality benchmarks!*

✺ *Innovate* Have fixed and flexible components for every course and adapt it as per specific needs. Use innovative teaching practices to facilitate deeper learning of the subject

✺ *Inspire* Push their boundaries and provoke them to think, question and inquire beyond conventions

✺ *Continuous* *Theory* ⇝ *Practice* ⇝ *Theory* ✺ *Practice* ⇝ *Theory* ⇝ *Practice*

Academic Experience - 2+ years of *Innovative Experiments*

Jan 2017 - Present *Guest Faculty*, **IIT Tirupati, India**
Paradigms of Programming - Spring 2017

- ➳ A work-in-progress book titled *The Lasting Contributions of Computing - Past, Present & Future* is an attempt to summarize ACM Turing Award lectures related to programming languages being co-written with 33 students - Did it help in learning seminal work?

Research [Term] Papers
- ➳ *How can we make students understand principles of programming languages in relation to emerging programming languages such as Scala, Rust, Swift?* - Empirical studies on different programming language constructs through analysis of top projects in *Github* repositories
- ➳ On the use and usability of functional programming languages [*Clojure* and *Rust*]
- ➳ A Survey of End User Programming Languages

Novel Methods *A book written with students as an outcome of the course*, interactive lectures, research-driven term papers, seminal and state-of-the-art papers, open-ended subject, challenging projects

August 2015 - Present
Visiting Faculty, **Indian Institute of Information Technology - Sri City**
Software Engineering (Foundations & Practice) - Monsoon 2016, Monsoon 2017
Programming Languages (Foundations & Practice) - Spring 2016, Spring 2017

➳ *How to teach software engineering (generally perceived as theory and boring) as an interesting and challenging course?*

➳ *How to teach principles for design of programming languages, not just their use?*

Novel Methods
interactive lectures, design studio exercises, 24 hour exams, surprise tests, hackathon, seminal and state-of-the-art papers, challenging projects

April 2011 - Sep 2016
Guest Instructor/Teaching Assistant, **IIIT Hyderabad, India**
10+ courses - course design, tutorials, material, grading, labs

Informally, I was a **Teaching Assistant** for *Prof. Kesav V. Nori*'s courses at **IIT Hyderabad**

Courses
Structured Systems Analysis and Design, Software Engineering and Topics, Process Engineering, Principles of Programming Languages, Compiler Design, Operating Systems, Data Structures

Critical Feedback on Teaching

Positive
I tried several experiments in teaching, some worked and some did not! It looks a bit bragging if I say anything about the course feedback I have got, especially for software engineering. Sharing few anecdotes here:

Overall rating
for instructor [Sridhar Chimalakonda] - 🏆 "+100000... (zeroes go on forever :D :P)"

"strength of the course is pushing students to their limits"

"Loved it :)"
"...How to grab attention without boring...instructor's commitment, way of explaining things, classroom activities..."

"faculty support from bottom of the heart, no compromisation in quality of the course"

"This is the course which made us to realize what we are capable of. Its not because of the course, its because of instructor and his way of teaching"

"No improvement is required, this course is already better than the best... This course has created hope in me that even I can solve complex problems

"About the course it is marvellous that so many things & new languages... were all learnt in a single semester for a single course... The other important thing I want to give about Sridhar sir. He is an awesome man with desired ability with lots of passion...he want to teach to every student in the class like through his interaction not only inside but outside the class. One last thing, regarding last one week that was one of the whole workable week till now in my life"

Improvements
When I interacted with students after getting an unusually extreme positive feedback for software engineering course, I realized that some students felt that the course is too rigourous [because I give examples from *Stanford*, *CMU*, *MIT* and discuss how we can be beyond that!]. Reduced the load a bit [not quality!] for Programming Languages course but the feedback was similar. I hope to figure out the right balance in the next version of course offerings!

Feedback link
Complete course feedback is available at http://bit.ly/2m3ycuv

The state-of-the-art of Computing Education in India ✺ ACM iSIGCSE

An Empirical Study
I am interested in the long run to contribute to the future of computing curriculum and education in India. With *code.org*, *cs4all* and several initiatives across the globe on computing education, what is the current state in India?

Solution Approach
Conduct (i) survey with all IITs [and then extend to other institutes] (ii) mine data of courses through LMS, course websites and so on (iii) in-depth case studies of exemplar courses (iv) make recommendations ✺ *What is being taught? How is it being taught? How the same is done across the globe? What's in for future?*

Software Engineering (Foundations & Practice)

Monsoon 2015
Monsoon 2016
IIIT Sri City

This course provides foundations for design of quality software and at the same time application of these foundations to develop reasonably complex software systems. Focus is on providing foundations like abstraction, [de]composition... and different aspects like processes, requirements engineering, architecture, design, testing with rigorous practical exercises. More details about the course including learning objectives, lectures, exercises and so on are available at http://researchweb.iiit.ac.in/~sridhar_ch/courses/sefp/home.html.

Interactive Lectures

➳ Core content of the subject is introduced through seminal and state-of-art articles with discussions around *How the topics would evolve in the next 5, 10, 20 years?* For example, in the first lecture, the entire class was divided into two groups each figuring out *What to develop?* (requirements) and *How to design?* towards "Design of Candy Crush Saga" game

Provocative Articles

➳ Classic papers and interesting notes were used instead of textbooks. For e.g., NATO conference reports and Dijkstra's EWD's were used to emphasize concepts like abstraction, components, proofs, design and so on

Innovative Strategies

➳ Weekly learning reports in course repositories (via *Github*), video assignments, 5 minute in-class assignments, 3 minute narratives, a lecture of student's choice (startups), exit slips (a note of attendance with learning), passion projects, a marathon lecture of 7 hours

Thinking and Reasoning

➳ *Design thinking exams.* Students are asked to argue for and against this statement *"Software lifecyles are a myth. Despite so many of them, people never follow them and they still deliver software"* in an end exam question

Hackathon

➳ A *crowdsourcing* experiment - developing 60 games in 24 hours as part of mid-exam

Continuous Evaluation

➳ *A time-boxing approach (review every 24 hours) to design of software in a 12-day rigorous exercise that yielded good-quality software projects.* This is a spin-off of *Scrum*, an agile software development process model.

Course Feedback

From anonymous feedback, *the course was extremely successful* but on the flip side, some students felt the course was heavy and demanded a lot of effort.

Programming Languages (Foundations & Practice)

Spring 2017
IIT Tirupati

Spring 2016
Spring 2017
IIIT Sri City

This course is an attempt to provide foundations for understanding the anatomy of programming languages whilst dissecting and analyzing these foundations in the context of several programming languages spanning across imperative, declarative paradigms. I briefly introduced topics such as correctness and axiomatic semantics through a series of lectures on ACM Turing Award articles. I used interesting languages such as Erlang (WhatsApp), Scala (Twitter), Clojure, Rust unlike conventional approaches. Specifications of programming languages were discussed from a design perspective.

Proposed Courses

The courses I propose follow an interactive style driven by challenging exercises both inside and outside class, design studio, narratives, exit slips towards instilling foundations and rigorous practice.

Foundations of Programming [Primer]

Programming exercises in imperative, object-oriented and functional paradigms

Today programming is done by people from all branches of engineering including end users. Are they doing good programming? What does good mean? Is programming same for computers, mobiles? This course is an attempt to teach foundations of programming as a three-part approach. *Specification*, *Implementation* and *Verification* of programs, supported with rigorous practice. In this course, programming will be introduced as a *problem solving* and an intellectual discipline. How does writing large programs differ? What does a program for *Twitter* look like? How do you write one? How to write programs with desirable qualities? Is it possible to learn programming without a programming language? Why is it critical to think of design before programming? and how to do co-design?

Foundations of Software Engineering [Primer]

In my view, this is a core course for every student

Do you design software using an ad-hoc approach? How is design of software similar or different from design of electronic circuits or construction? This course proposes to teach basic foundations and essential constructs for design of software. The key goal of this course is to instill the notion of systematic software development and the necessary knowledge and skills early on to develop software. Students will learn systematic ways of developing software and apply these concepts towards developing reasonably complex software in a challenging project. Specifically, they will learn basics of requirements engineering, design methods, coding and testing techniques. The students will also learn and use version control, diagramming languages like UML, debugging and several tools to support their project.

Possible Courses

Operating Systems - My B.Tech thesis is on designing an interface between hardware and operating systems that introduced the idea of software for rent and a powerful pre-boot interface. I can revise and probably make an attempt to teach this course.

Data Structures - I can make an attempt to teach this course owing to my interests in computer science education. An experiment along with my BTP student is to gamify and use music to teach data structures. *Can we teach linked lists through a snake like game?*

Electives

These advanced courses (I call them as *Research & Practice* courses) are primarily targeted at senior undergrads, masters and doctoral students. Students will work on interesting research projects throughout the semester and submit a good paper at quality venues.

Topics in Software Engineering

The core idea is to provide an overview of critical topics like software architecture, design patterns, software product lines, reverse engineering, process and product quality engineering, refactoring, requirements engineering.

Topics in Educational Technologies

To provide basic foundations for design of educational technologies, instructional design, learning environments, personalized learning. Primary focus will be on doing it in the Indian context for the scale and variety of all subjects, classes, boards, languages, skills, learners.

Thanking all my teachers, students and universities for their wonderful support!